P E D S Y S

A Pedigree Data Management System

Version 2.0

For Sun Microsystems Unix, MS-Windows and Macintosh OS

USER'S MANUAL

(Draft Version, July 1999)

Bennett Dyke, Ph. D.

PGL Technical Report No. 2

Population Genetics Laboratory
Department of Genetics
Southwest Foundation for Biomedical Research
San Antonio, Texas 78245

Table of Contents

(Note: This is a draft version of the manual; items marked TBA are in preparation.)

Acknowledgments

The concepts underlying PEDSYS were originally developed in 1978-79, while I was on the faculty at the Pennsylvania State University. During that time, I devised a version of the PEDSYS record format and wrote four of the original Data Management Programs (INDEX, LIST, SORT, and SUBSET), which were designed for human population studies.

In 1981, I moved to the Southwest Foundation for Biomedical Research. The basic PEDSYS record structure, as well as these four programs, were modified for use with non-human animal populations, and the linked pointer file structure was developed with the help of Ms. Linda J. Collins. Ms. Collins implemented the file structure, and took major responsibility for PEDSYS programming and development over the next five years with the assistance of Ms. M. Connie Murguia. Ms. Collins' original work underlies much of the subsequent development of PEDSYS. Dr. Jean W. MacCluer has played a major role in the overall planning and development of the package, and has made numerous suggestions for new functions and improvements. A number of programs are based on her original algorithms.

In 1987, Ms. Linda Freeman-Shade became lead programmer for the package. She did most of the conversion of the FORTRAN code first for MS-DOS, and then for Sun Microsystems Unix. In addition to these demanding tasks, Ms. Freeman-Shade has made important progress in algorithm development and implementation for many of the Pedigree Management Programs. Much of the success of the package is the result of her programming skill, understanding of the objectives of pedigree management, and an extraordinary talent for making complex tasks simple to carry out. Ms. Rose H. King contributed substantially to algorithm and program development at many levels, and provided numerous ideas for modification of the package as it has grown in scale. Ms. Denise Wahl assisted in the early stages of conversion to the Unix operating system. Mr. Tom Dyer continued this conversion, and has been responsible for the Unix systems interface (including security) of the package. In the process, he has made a number of important improvements to several of the programs. Mr. Richard Polich has contributed to programming and has been instrumental in implementing the Windows version of the package. Mr. Paul Mamelka has converted the programs for the Apple Macintosh, developed the Mac interface that will serve as the prototype for subsequent graphic versions of PEDSYS, and has taken the lead in organizing and restructuring source code into a modular form that facilitates implementation of the package on any hardware platform or operating system.

It has been a pleasure to work and learn from these individuals, who make up the programming staff of the Population Genetics Laboratory. The fact that we have found PEDSYS to serve our needs so successfully is not only the result of their technical skills, but also of their ability to work with the scientists and technical staff of the Department of Genetics at the Southwest Foundation who have used the system, and who continually make suggestions for its improvement.

We have licensed an extremely efficient sorting routine (Opt-tech Sort) for incorporation into PEDSYS programs. Opt-tech Sort is the copyrighted property of Opt-Tech Data Processing, P.O. Box 678, Zephyr Cove, NV 89448. Terms of the license permit use of the routine in any authorized version of PEDSYS.

Major support for the development of PEDSYS has been provided by NIH Grants RR02229 (Genetics and Inbreeding in Nonhuman Primates and RR). Support for programming personnel also has been provided by NIH grants HL28972 (Diet and Genotype in Primate Atherosclerosis) and GM31575

(Genetic Analysis of Common Diseases: An Evaluation).

Copyright and Limits on Use

The development of PEDSYS represents a substantial and costly effort that has been supported by research grants from the U.S. National Institutes of Health. We feel strongly that it is unethical to sell the efforts of work funded from these sources to individuals or institutions also supported by public money. Although we will provide PEDSYS to NIH grantees and non-profit institutions (or their equivalents in other nations) at a fee covering only the costs of media, shipping, etc., the package is not in the public domain. The programs and manual bear copyright notices dated 1988 or later, and are subject to the following conditions of use:

The programs may not be sold or incorporated into other software packages that are sold for profit, nor may fees (other than standard costs associated with data entry and computer services) be charged for their use without the permission of the Southwest Foundation for Biomedical Research.

The programs and manual may be copied and distributed to users engaged in non-profit research, non-profit management of animal colonies, or non-profit health care delivery. However, we ask that those receiving the package from sources other than the Southwest Foundation notify us so that we may keep our distribution list current. The intent of this request is to avoid problems that occur when the software and manual are out of date or incompatible.

We ask that users of the package report software bugs, confusing instructions, missing functions, etc., to us so that we can continue to make improvements.

I. INTRODUCTION

PEDSYS (Pedigree Database System) is a series of programs written for management and analysis of genealogical and demographic data. The system was designed principally as a research tool for use in genetic analysis of either human or non-human subjects. Although some of the programs are specialized, many are general enough to be used effectively with data of any sort.

General Principles

Several criteria governed the development of PEDSYS:

1. An important requirement was that PEDSYS support the collection, management and analysis of data taken from the same population by investigators in several different laboratories. This meant that it should be possible for anyone using the system to manage and analyze data generated in their own projects, and at the same time to be able to combine these data (a) with a common pool of demographic and genealogical information about the population and (b) with data sets collected in other projects. To accommodate this requirement, PEDSYS databases are organized around a Master File containing basic information on vital events and genealogical relationships for individuals in the population. Any number of independently developed Data Files may be joined as needed to the appropriate Master File. Data Files can be copied or combined in various ways to form entirely new files that maintain their linkage with the Master File.

2. Another critical need was the ability to import foreign data (that is, files developed on other database systems), and to export PEDSYS data to other programs or computers for analysis. This requirement was met by choosing a simple file structure based on formatted records containing nothing but data represented by printable ASCII characters. That is, there are no file headers, record descriptors, or control characters that must be processed (or avoided) before accessing information kept in the database. In addition, several of the PEDSYS General Data Management programs are designed to transform or reconfigure ASCII data into any required format with considerable ease and flexibility.

3. Although we anticipated that some individuals would become highly skilled in the use of the system, we felt that it was important for non-experts to be able to use PEDSYS effectively without extensive training. Experience has shown that a common pattern for many scientists and technicians is to use a set of PEDSYS programs intensively for a matter of days or weeks during the course of a project, after which weeks or months will pass before the system is used heavily again. We therefore paid particular attention to the development of an orderly set of commands, prompts and instructions that would make the programs as self-instructing as possible without irritating the proficient user.

   A major effort has been made to keep the programs consistent and easy to use. We have often chosen for the sake of simplicity to write entirely new programs, rather than add new functions to existing programs. As far as possible we have kept the same conventions for keyboard entry and screen displays in all programs, and we have paid particular attention to our use of the English language.

4. PEDSYS had to accommodate constantly evolving data sets. The continual changes in form and content of research data on growing animal populations required that the user be able to change or experiment with the composition and format of data quickly and without a major restructuring of the database. Several features make it simple to reorganize data: (a) The independence of PEDSYS files means that any single Data File can be created, changed or deleted without affecting another. Master Files provide a stable core of basic information that remains unaffected as Data Files are changed. (b) A number of programs have been designed specifically to automate the task of merging, reformatting, and transforming data.

5. We feel that large-scale software packages must not be dependent on any one type of computer or operating system. Consequently, PEDSYS programs are written principally in ANSI Standard FORTRAN-77 with numerous functions in ANSI Standard C. Machine-dependent code has been isolated and identified in the few cases where it has not been possible to avoid altogether (chiefly file access and screen display routines). Written originally for a time-sharing Concurrent Computer (Perkin-Elmer) 3210, the system has been adapted for MS-DOS, Microsoft Windoes, Sun Microsystems Unix (SunOS 4.6 and Solaris), Linux, and the Apple Macintosh OS (System 6.07 and higher).

PEDSYS has grown and evolved over the years in response to the suggestions of the scientists and technicians who have used the system. New programs are first written as quickly as possible in response to the needs of a specific research question. Previously developed program modules or algorithms are incorporated wherever possible, but refinements are added only as users gain experience with the program. This approach has resulted in programs that meet a broad range of requirements for pedigree data management.

Three features distinguish the PEDSYS database management system from most commercial database systems:

1. PEDSYS Master File records incorporate as data a set of pointers (Sequential IDs) that link each individual with key members of the nuclear family. Thus, the Master File carries its own indexing for family members, making it possible to construct extended families very quickly, and without the need to access additional files.

2. We have spent some effort to make screen displays and listings simple and easy to interpret, but because the system is so frequently used to prepare data for other analysis programs, PEDSYS output is oriented principally toward the generation of files, rather than prpoduction of elaborate reports.

3. PEDSYS programs are compiled, containing no interpreted code. A disadvantage of this approach is that the user cannot add new functions to pre-existing programs, as can be done with many commercial database systems. On the other hand, this program structure makes it easier to control and test the various functions and algorithms used, as well as to assure consistency and simplicity.

II. CREATING A PEDIGREE DATABASE (Under revision)

The simplicity of PEDSYS program operations depends on correct configuration of the file and directory structure. The basic rules are

1. Master Files and their associated Data Files must be in the same directory.

2. Each Master File or Data File must have an associated Code File.

3. A Master File must have a Master Pointer File (**MPOINT.<EXT>**).

4. Data Files are linked to a Master File by a Data Pointer File (**DPOINT.<EXT>**).

5. Records of multiple-entry Data Files must be linked by internal pointers.

Although these principles are simple, their implementation must be precise. Of all file management procedures, perhaps none is as critical as the initial creation of a PEDSYS database. A major effort has gone into developing programs that automate this, and other file management operations. The instructions that follow for creating PEDSYS files assume:

a. That PEDSYS program files, and auxiliary files **CODE.STD** and **DEVICES** have been installed in the PEDSYS Program Directory, and that

The appropriate system files have been modified to include a path to the PEDSYS Program Directory.

b. That a file extension that identifies the population, as well as appropriate labels and file names, have been chosen, and that Master File, Data File, Pointer File, etc. names are in upper case (see Appendix A).

c. That a PEDSYS Data Directory has been created. This directory may initially contain the Directory File **DBFILES**, although this file will be created automatically by programs ALTMASTR or ALTERDAT if it is not found in the directory.

Creating a Master File from the Keyboard

In addition to the initial assumptions above, creation of a Master File by entering data from the keyboard assumes that pedigree data are available in the form of paper records, with a single record for each individual. Each record must contain minimally five of the last six items listed in **CODE.STD** (see The Standard Mnemonics File, Appendix A). These are

Item	No. of Characters	Standard Mnemonic	Data Type	Notes
EGO Permanent ID (PID)	3-9	ID (or EGO)	C	may be numeric
Sire PID	3-9	SIRE (or FA)	C	may be numeric

```
Dam PID                 3-9     DAM (or MO)    C   may be numeric
EGO's sex                 1     SEX            C   coded M,F,U or 1,2,3
EGO's birth date        3-8     BIRTH (or BIR) D
EGO's exit date         3-8     EXIT           D   optional, but useful
```

It is important that these items conform to the specifications of the **CODE.STD**. Order of the items is not important.

Creation of the files necessary for a minimal Pedigree Database is done in four steps:

Step 1. Create a Code File for the new PEDSYS Master File by running program CODE, choosing Create Mode and entering **H** or **N** to specify the human/nonhuman Master Code File option. CODE will automatically enter whatever information it can that is required by the Master File format, and will request entry of optional labels, mnemonics, field lengths and data types. When entry is finished, the output file should be named **CODE.<EXT>**, where **<EXT>** is the file extension identifying the population. If you forget and give the file its default name of **code.tab**, it will be necessary to rename it. Depending on the computer on which PEDSYS is installed, this can be done with one of the following commands:

MS-DOS	`REN CODE.TAB  CODE.<EXT>`
Unix	`mv code.tab  CODE.<EXT>`
Macintosh	Select the file name by clicking once on the name portion of the icon, and replace **code.tab** with **CODE.<EXT>**.

Step 2. Run program ALTMASTR in START Mode, choosing file extension **<EXT>**, that is, the same extension selected when renaming the Code File. Enter data as required. ALTMASTR generates an intermediate file named **update.<EXT>**, and creates a preliminary version of the Directory File DBFILES, if the file does not already exist.

Step 3. Run program SETMASTR, using extension **<EXT>** as the population identifier when prompted to do so. SETMASTR creates the files required, that is, a Master File named **MASTER.<EXT>** and a Master Pointer File, **MPOINT.<EXT>**. SETMASTR also creates the Directory File **DBFILES**, or modifies an existing copy to accommodate the new Master File. Output file **setmastr.err** may reveal inconsistencies that require correcting the original data (see notes for ALTMASTR and SETMASTR in Chapter V).

Step 4. Run a spot check of records in **MASTER.<EXT>** using program BROWSE or SHOW . If data are not correct, it may be necessary to edit records using program ALTMASTR, and then run SETMASTR.

Creating a Master File from Imported Computer-readable Data

Creation of a Master File by importing computer-readable data assumes that pedigree data are

available in the form of a foreign (that is, non-PEDSYS) ASCII text file containing a single record for each individual. Contents and format of these records should be the same as those listed above for keyboard data entry.

Step 1. Procedures for creating a Master File from a foreign data file will vary, depending on the record format of the foreign data file.

Character-delimited format

Data are frequently transferred between computers or application programs using a simple format by which individual items are separated by specially designated field separators, usually the tab character (ASCII code 09), comma (ASCII code 44), or blank (ASCII code 32). If your data are formatted this way, use program TRANSLAT, selecting the **Delimited field to PEDSYS** option in the introductory screen display, choosing the name of the foreign file when prompted, and letting the program create a Code File based on input data field widths. This operation will produce an output Data File **translat.out** in PEDSYS format, and a rudimentary Code File **translat.cde**, in which field widths and data types are correctly specified, but item descriptors and mnemonics are numbered, but otherwise undefined.

Item descriptors and mnemonics in **translat.cde** can then be modified with program CODE to better describe the data in **translat.out**.

Fixed format

The safest way to create a Master File from a foreign file having fixed format records is to run program ITEMIZE using either the **h** (human) or **n** (nonhuman) Master File option that appears in an early screen display. The principal function of ITEMIZE is to allow you to define and examine fields in an experimental fashion, automatically creating a Code File (**itemize.cde**) that defines the PEDSYS structure of a foreign file. ITEMIZE can be run either with or without an input Code File.

If no input Code File is used, you will be prompted for the field width, data type and first mnemonic of each item. In this case, choose Standard Mnemonics when entering descriptors for EGO, Father (or Sire), Mother (or Dam), Sex, Birth Date and Exit Date when prompted.

If a Code File describing the structure of the foreign file is available, it may used as input to ITEMIZE. In this case, Standard Mnemonics should have been used when entering descriptors for EGO, Father (or Sire), Mother (or Dam), Sex, Birth Date and Exit Date at the time the Code File was created.

In either case, ITEMIZE checks to make sure that Standard Mnemonics required in a Master File (EGO or ID, BIRTH, FA or SIRE, MO or DAM, and SEX) appear in the output, and that these items are given the appropriate Data Type and field width. An opportunity to assign Standard Mnemonics is provided if they are not found initially, after which the foreign file is copied (with optional minor alterations) into file **itemize.out**.

Step 2. Run a spot check of records in **translat.out** (or **itemize.out**) using program BROWSE or SHOW (paying particular attention to records at the beginning and the end of the file). If data on the records appear to be correct, go on to Step 3. If data are not correct, it may be necessary to

a. Modify the format of records in the file using program CODE,

b. Force records to conform to the format specified in **translat.cde** (or **itemize.cde**) by running the input file through program REFORMAT with all items selected for output.

c. Edit records using program ALTERDAT, or

d. Re-submit the original data to program ITEMIZE.

Step 3. Run program INDEX, specifying the Master File option (choices 1 or 2), and the **Unlisted Pedigree File** choice. When prompted, enter the name of the correctly formatted version of the foreign file.

Step 4. Rename output files **index.out**, **index.cde** and **index.mpt** as **MASTER.<EXT>**, **CODE.<EXT>** and **MPOINT.<EXT>**, respectively, where **<EXT>** is a file extension identifying the population. Output file **index.err** may reveal inconsistencies that require correcting the original data, and re-running INDEX (see Chapter V).

Step 5. Run program DBFILER to put the new Master File in DBFILES.

Creating a linked Data File from the Keyboard

In addition to the initial assumptions above, creation of a linked Data File by entering data from the keyboard assumes that data are available in the form of paper records. The first item in each record must be a PID identical to one of the PIDs found in the Master File. Creation and linkage of a Data File requires is done in four steps:

Step 1. Create a Code File for the new PEDSYS Data File by running program CODE, choosing Create Mode and skipping the Master Code File options. CODE will request entry of optional labels, mnemonics, field lengths and data types. When entry is finished, the output file should be named **<filename>.CDE**, where **<filename>** is the name chosen to identify the new Data File. If you forget and give the file its default name of **code.tab**, it will be necessary to rename it. Depending on the computer on which PEDSYS is installed, this can be done with one of the following commands:

MS-DOS	`REN CODE.TAB <FILENAME>.CDE`
Unix	`mv code.tab <FILENAME>.CDE`
Macintosh	Select the file name by clicking once on the name portion of the icon, and replace **code.tab** with **<FILENAME>.CDE**

Step 2. Run program ALTERDAT in START Mode, entering the name of the new Data File and the Master File to which it will be linked, when prompted. These should be **<FILENAME>.<EXT>** and **<FILENAME>.CDE**, respectively (that is, the same file name selected when renaming the Code File in Step 1 above). ALTERDAT will modify the Code File as needed by adding an internal pointer (mnemonic PNTR) as the last field if the file is a Multiple-entry Data File. Enter data as required in ADD Mode. ALTERDAT creates a Data File interactively, without an intermediate step.

Step 3. Run a spot check of records in **<FILENAME>.<EXT>** using program BROWSE or SHOW. If data on the records appear to be correct, go on to Step 4. If data are not correct, it may be necessary to edit records using program ALTERDAT.

Step 4. Run program DBFILER in ADD Mode, choosing option **D** to add a linked Data File, and entering the number corresponding to the Master File to which the Data File is to be linked, and the name of the Data File itself, when prompted. DBFILER will modify the Directory File **DBFILES**, and create or modify (as necessary) a Data Pointer File **DPOINT.<EXT>**, so that each is appropriately configured for use as input to program DATALINK.

Step 5. Run program DATALINK, entering the number corresponding to the Master File to which the Data File is to be linked, when prompted. DATALINK produces an updated version of **DPOINT.<EXT>** that contains the appropriate pointers between the Master File and Data Files. DATALINK also generates a new version of **<FILENAME>.<EXT>** with updated internal pointers, if it is a Multiple-entry Data File (see File Types in Appendix A). Output file **datalink.err** may reveal inconsistencies that require correcting the original data (see documentation for program ALTERDAT).

Creating a Linked Data File from Imported Computer-readable Data

Creation of a Linked Data File by importing computer-readable data assumes that data are available in the form of a foreign (that is, non-PEDSYS) ASCII text file containing one or more records for any individual in the Master File. The Permanent ID of the individual must be included in each record. It is important that this PID have exactly the same format as the PID in Master File records. Other content and format requirements of these records are the same as those listed above for keyboard data entry.

Step 1. Procedures for creating a Linked Data from a foreign data file will vary, depending on the record format of the foreign data file.

Character-delimited format

For records character-delimited format, follow the procedure outlined above for creation of a Master File, using program TRANSLAT.

Fixed format

The easiest way to create a PEDSYS Data File from a foreign data file with fixed format records is to run program ITEMIZE without selecting the Master File options:

If no input Code File is used, you will be prompted for the field width, data type and first mnemonic of each item. In the case of a Linked Data File, make sure that the Standard Mnemonic **EGO** (or **ID**) is used to identify the Permanent ID that must appear on each record.

If a Code File describing the structure of the foreign file is available, it may used as input to ITEMIZE. In this case also, make sure that the Standard Mnemonic **EGO** (or **ID**) was used to identify

the Permanent ID that must appear on each record. If this is not the case, change the mnemonic as needed.

Note: If you are sure that the Code File describing the structure of the foreign file is correct, Step 1 may be skipped. This shortcut is not advised, however.

Step 2. (As needed) run file **itemize.out** through program REFORMAT if the PID is not the first item in the records, or if items require removal or rearrangement.

Step 3. Run a spot check of records in **translat.out** (or **itemize.out**) using program BROWSE or SHOW (paying particular attention to records at the beginning and the end of the file). If data on the records appear to be correct, go on to Step 4. If data are not correct, it may be necessary to

a. Modify the format of records in the file using program CODE,

b. Force records to conform to the format specified in **transl.cde** (or **itemize.cde**) by running the input file through program REFORMAT with all items selected for output,

c. Edit records using program ALTERDAT, or

d. Re-submit the original data to program ITEMIZE.

Step 4. Rename output files **itemize.out** (or **reformat.out**) and **itemize.out** (or **reformat.cde**) as **<FILENAME>.<EXT>** and **<FILENAME>.CDE** after archiving or deleting the original files of the same name, and then run program DBFILER to modify the Directory File by adding the name of the new Data File to the Linked Data File list.

Step 5. Run program DATALINK, entering the number corresponding to the Master File to which the Data File is to be linked, when prompted. DATALINK produces an updated version of **DPOINT.<EXT>** that contains the appropriate pointers between the Master File and Data Files. DATALINK also generates a new version of **<FILENAME>.<EXT>** with updated internal pointers, if it is a Multiple-entry Data File (see note below). Output file **datalink.err** may reveal inconsistencies that require correcting the original data (see notes for ALTERDAT in Chapter V).

A note on importing Multiple-entry Data Files: Recording multiple entries for the same individual may be handled in different ways, depending on the foreign database software used. If multiple entries for the same individual are already stored on separate records in the foreign file, all that is required is to make sure that the records are formatted and sorted properly (see the *Multiple-entry Data Files* section in Appendix A for details) prior to submitting them to program DATALINK to generate PEDSYS Multiple-entry Data Files. If, on the other hand, multiple entries for the same individual are stored on a single record, it will be necessary to disassemble the sequence into separate identically formatted records using program REFORMAT

Creating an unlinked Data File from the Keyboard

If a Data File is not to be linked to a Master File, only two steps are required for its creation from the keyboard.

Step 1. Create a Code File by running program CODE, choosing Create Mode and skipping the Master Code File option. CODE will request entry of optional labels, mnemonics, field lengths and data types. When entry is finished, the output file should be named **<filename>.cde**, where <filename> is the name chosen to identify the Data File.

Step 2. Run program ALTERDAT in START Mode, entering the name of the Data File and its associated Code File when prompted. These should be **<filename>.<ext>** and **<filename>.cde**, respectively (that is, the same file name selected when renaming the Code File). One item must be selected as a record identification field. ALTERDAT creates a Data File interactively, without an intermediate step.

Step 3. Run a spot check of records in **<filename>.<ext>** using program BROWSE or SHOW. If data on the records are not correct, it may be necessary to make corrections, as described above for Linked Data Files.

Creating an unlinked Data File from Imported Computer-readable Data

The easiest way to create an PEDSYS unlinked Data File from a foreign data file is to run program ITEMIZE.

Run a spot check of records in **itemize.out** using program BROWSE or SHOW (paying particular attention to records at the beginning and the end of the file). If data on the records are not correct, it may be necessary to make corrections, as described above for Linked Data Files.

Optionally, program DBFILER may be run to modify the Directory File by adding the name of the new Data File to the Unlinked Data File list.

III. PEDSYS PROGRAMS

AGE

Program AGE calculates ages for all individuals in a file by finding the difference between two dates represented in Gregorian form (order year-month-day). One of these dates must be taken from the individual's record (for example Master File BIRTH, ENTRY, or EXIT dates). The other date also may be taken from the record, or it may be entered as a constant from the keyboard either in numerical form or as the character string TODAY (which takes the current date from the computer's hardware clock).

Introductory Display:

```
PEDSYS program AGE - Copyright 1992, 1999 SFBR

AGE calculates ages for all individuals in a file by finding the difference
between two dates represented in Gregorian form (order year-month-day).

  1.  Unlinked File
  2.  reformat.out
  3.  EXAMPLE
  4.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>3
```

After a file has been chosen, the output format of ages to be calculated is selected:

```
Enter the number corresponding to the way in which age is to be output:

    [1]  In years (with fraction)
     2.  In months (with fraction)
     3.  In days
     4.  In years, months, and days
>
```

Once the format has been designated, the following display appears:

```
   1 EGO SEQUENTIAL ID       9 PATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   2 SIRE'S SEQ             10 MATERNAL SIBSHIP SIZ   18 Exit Code
   3 DAM'S SEQ              11 FULL SIBSHIP SIZE      19 Mate's Permanent ID
   4 FIRST OFFSPRING SEQ    12 EGO Permanent ID       20 PEDIGREE NUMBER
   5 NEXT PATERNAL SIB SE   13 Birth Date (YYYYMMDD   21 GENERATION NUMBER
   6 NEXT MATERNAL SIB SE   14 Sire's Permanent ID   *22 EXIT-BIRTH
   7 NEXT FULL SIB SEQ      15 Dam's Permanent ID
   8 NUMBER OF OFFSPRING    16 Sex (M, F or U)
--------------------------------------------------------------------------------
  AGE         Select dates             File - MASTER.XMP
--------------------------------------------------------------------------------
To find the difference between two dates, enter the first date, a minus
sign or blank, and the second date. Values may be specified as a number
corresponding to a date in the list above, a number preceded by a "D", or
the word "TODAY" (examples:  17-13, D19590631-13, TODAY 17). Enter [C] to
continue to next step.
>                  <--  COMMAND
```

Here, age based on the difference between Birth Date and Exit Date in the Example Master File has been selected, and the resulting new value appended as Item 22.

Notes:

- Preferred PEDSYS date format (YYYYMMDD) conforms to International Standard ISO 8601 (see Appendix A, Data Types).

- All dates used in a calculation must have the same format: YYYYMMDD, YYYMMDD, YYYY, or YYY (where Y = year, M = month and D = day). These formats are specified by the length parameter in the Code File as 8, 7, 4, or 3, respectively. A leading zero in three-digit representation of the year implies a third millenium date (i.e., **001** = **2001**). Two-digit representation of year (YYMMDD, or YY alone) is not allowed.

- A date entered from the keyboard must have the same length as is specified in the Code File. For example, when dates are specified as having length 8, it will be necessary to enter D, followed by exactly 8 digits whose order corresponds to YYYYMMDD. An error message is displayed on the screen if the length is not correct.

- At least one of the dates for an age calculation must be taken from the individual's record, that is, it must be selected from the item list display. Items that are not defined as dates in the Code File (that is, data type D) cannot be used in age calculations. An error message will be displayed on the screen if this requirement is not met.

- In contrast to other PEDSYS data types, representation of unknown or missing dates depends to some extent on context. This is because a blank date field may designate not only dates that are unknown or missing in the usual sense, but also anticipated dates of events that have not yet occurred (such as deaths or scheduled examinations, etc.).

  In some cases, events may be known to have occurred, but evidence for associated dates may not be implicit in the database. For example, it may be known that an individual has died or has been examined in clinic, but no information about the event is available or recorded. In this case, the unknown date may be represented by a full field of nines (**99999999** or **9999999**). If dates are represented by year alone, **9999** may be used, *but not 999* (which represents the year 1999 in our abbreviated format). Nines may not be used to represent unknown or missing month or day when year is known.

- The table below lists various configurations of the date field and their interpretations by program AGE:

Year	Month	Day	Interpretation	Note
+	+	+	Full representation	(a)
+	+	-	Day estimated	(b)
+	-	-	Month and day estimated	(c)
-	-	-	Skipped	(d)
+	-	+	Invalid	(e)
-	+	+	"	(e)
-	+	-	"	(e)
-	-	+	"	(e)

  where **+** represents data present, **-** represents missing data.

Interpretation:

(a) Full representation of year, month and day. An error message is written to **age.err** if:

Non-numerical characters (other than blanks) are found in the record, or are entered from the keyboard after the letter D. Embedded blanks are also invalid.

Month falls outside the range 1 - 12, or day falls outside 1 - 31 (with exceptions for missing elements noted below).

Year is present and month and/or day is represented by nines.

(b) When year and month are present, but day is missing (expressed as blanks or zeros), an age calculation is done using day 15 as an estimate of mid-month.

(c) When year is present, but month and day are missing (that is, both fields are filled with blanks or zeros), an age calculation is done using 30 June as an estimated mid-year date.

(d) Unknown or missing dates (expressed as a full field of blanks, zeros or nines) are skipped in age calculations. No error message is generated.

(e) A data error is assumed when lower order elements (day, for example) are present, but higher order elements (i.e, year) are missing in the date. This condition results in the date being skipped and an error message written to **age.err**.

- Differences in length of month and leap years are taken into account.

- Age may be expressed in four different ways:

  As a single item consisting of years (with fraction to two decimal places).
  As a single item consisting of months (with fraction to two decimal places).
  As a single item consisting of days (without fractions).
  As three items consisting of years, months, and days, respectively.

- Ages may be negative.

- A blank field is generated whenever an age cannot be computed.

- The following sequence appears after the confirmation sequence whenever an Exit Date (designated by the standard mnemonic **EXIT** in the Code Field) is used in calculations:

```
Substitute another date for a blank EXIT date? Y/[N]
>y

Enter date (YYYYMMDD), or RETURN alone for today's date (19981230)
>
```

This substitution makes it possible to compute maximum ages achieved by all individuals in the population, whether they are living or dead.

- The command line startup shortcut (program name, followed by a blank and an input file name) may

be used to start this program. Example: **age subset.out**.

Files:

Input: Any Master File or Data File, and an associated Code File. Recursive use of output files is permitted, that is, file **age.out** may be used as input to program AGE.

Output: **age.out** contains all records in the input file in their original order; records contain all items in the input records in their original order, followed by items containing age calculations.

age.cde defines the record structure of **age.out.**

age.err identifies invalid dates and the records on which they appear.

Progam limits:

12 output items. That is, up to 12 age calculations that result in single items (years, month, or days), or up to four age calculations that result in three items (years+months+days), etc.

ALTERDAT

ALTERDAT is a data entry and editing program for Data File records. PEDSYS records are very simply formatted, and can in principle be edited with any general-purpose text editor. However, this is not recommended because of the danger of disrupting the fixed format of PEDSYS record structure.

Introductory Display:

```
PEDSYS program ALTERDAT - Copyright 1992, 1999 SFBR

ALTERDAT is a data entry and editing program used to modify or create
Single- or Multiple-entry Data Files. Two update Modes are available:

Interactive Mode - items are entered or updated directly from the key-
   board.  DATALINK must be run after creating Linked Data Files, DELRECS
   must be run if records are deleted. More than one file may be changed
   or created per run.

Batch Mode - changes entered at the keyboard are stored in one or more
   temporary Update Files used in turn as input to SETDATA and/or
   DELRECS.  Only one file may be changed per run.

Note: an appropriate Data Code File must accompany the Data File used by
   this program.

Enter - [C] to CHANGE a file Interactively
         N  to start a NEW data file Interactively
         B to enter BATCH updates for a single file
         Q  to QUIT program.
>C
```

ALTERDAT has two modes of operation. In **Interactive Mode** (options **C** and **N** above), permanent changes, except for deletions of records, are made directly to Data File records in a single step. In **Batch Mode** (option **B**), the program becomes the first step of a process by which a temporary Update File is created and used in turn as input to program SETDATA which incorporates changes permanently into the Data File selected. When either mode is selected, a Standard File List Display sequence appears:

```
                                                          (Interactive Mode)
------------------------------------------------------------------------------
 1.  Unlinked File
 2.  EXAMPLE

Enter number of file or file group to be altered, or filename.
>2

Specify the Linked Data file to be updated.

   1. Markers + Quant. P'types
   2. Blood Sample Inventory

Enter a file number.
>1
```

LINKED DATA FILES

Linked Single-entry Data Files

A linked Single-entry Data File (**Markers + Quant. P'types**) has been selected in the example above. This is followed by a request for entry of a password if passwords have been included in the Data File Descriptor Record of **DBFILES** (see note **q** in the Notes on DBFILES Record Formats section in Appendix A). After a correct password has been entered (or if no password is required), the following instruction appears:

```
Updating QUANTGEN.XMP                                    (Interactive Mode)
--------------------------------------------------------------------------------

Make a backup copy of QUANTGEN.XMP before making changes?  [Y]/N
>
```

If the Backup option has been chosen, the contents of **<FILENAME>.<EXT>** will be copied and saved as **<FILENAME>.OLD**. In the example above, the Data File selected was **QUANTGEN.XMP**, and the backup file will be named **QUANTGEN.OLD**.

ALTERDAT begins its data entry cycle for modification of a linked Single-entry Data File with the following options display:

```
Updating QUANTGEN.XMP                                    (Interactive Mode)
--------------------------------------------------------------------------------

Enter - [E] to EDIT records
         A  to ADD records
         D  to DELETE records

Edit control

         O  to set Order of access (Order is RANDOM)
         F  to set conFirmation display (Confirmation is ON)
         L  to switch Log setting (LOG is ON)

If finished with QUANTGEN.XMP

         C  to CHANGE an existing file
         N  to start a NEW file.
         Q  to QUIT program.
>
```

The first three commands shown in this display govern the type of modification that will be done to the file selected(editing, adding or deleting records):

Command E -- Edit Linked Single-entry Data File records

Entering **E** (EDIT) results in a Standard Item List as shown below from which items to be modified are selected so that only those that must be changed will appear on the Edit screen:

```
   1 ADA  Phenotype          3 Tot. Ser. Cholestero
   2 APOB Pvu2a Genotype   H 4 APOAI Level
--------------------------------------------------------------------------------
  ALTERDAT    Items to be modified            File - QUANTGEN.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to items to be modified. Precede number with "H"
to carry over entry of item from one record to the next. Enter "A" (or "HA")
to include all items, [C] to continue to next step.
>
```

Preceding numbers with an **H** (for "hold") carries over the previously entered value to the next record to be changed or added. This feature makes it unnecessary to re-enter duplicate values on successive records. The value displayed in the space provided for the item on the data entry screen will be added to the record unless it is changed or erased. The new change or erasure will then be passed to subsequent records selected for alteration or addition. In the example above, all items have been selected for modification, and the APOAI Level phenotype has been marked for hold-over. After the usual confirmation step a request appears for the Permanent ID associated with the record to be changed :

```
Last entry in QUANTGEN.XMP is ID077              (EDIT Record: Random Order)
--------------------------------------------------------------------------------

Enter individual's Permanent ID, or RETURN to change mode of entry.
>ID069<-   EGO Permanent ID
```

Here, PID ID069 has been entered for modification. At this point, ALTERDAT checks to assure first that a record for this individual can be found in the Master File. If EGO is found in the Master File, the program next searches for an appropriate record for EGO in the Data File. If one is found, the display shown below appears. Note that in this case, no entry appears for the APOAI Level Phenotype. This is because this item is marked for hold-over (**H**), which automatically erases any previous entry.

```
 ID: ID069                        Page  1/ 1     (EDIT Record: Random Order)
--------------------------------------------------------------------------------
  1. ADA  Phenotype       >B   <
  2. APOB Pvu2a Genotype  >    <
  3. Tot. Ser. Cholesterol>103<
H 4. APOAI Level          >     <

Enter number of item to be modified, "S" to SAVE data, or "X" to EXIT
without saving data.
>
```

- *Warning: The carry-over feature should be used with care, as it is easy to inadvertently overwrite valuable information.*

When changes have been entered and saved (**S**) for this record, a new request for a PID is displayed:

```
Last entry in QUANTGEN.XMP is ID077              (EDIT Record: Random Order)
Last entry SAVED was ID069
--------------------------------------------------------------------------------

Enter "=" to reenter changes for last entry SAVED, otherwise enter
individual's Permanent ID, or RETURN to change mode of entry.
>     <-   Ego ID
```

Note that as soon as at least one record has been saved, PIDs of both **Last entry in <file>** and **Last entry SAVED** (which need not be the same) are displayed. At this point the PID of a new individual may be entered and another edit cycle begun. Entry of RETURN alone redisplays the options display:

```
Updating QUANTGEN.XMP                                    (Interactive Mode)
--------------------------------------------------------------------------------

Enter - [E] to EDIT records
         A  to ADD records
         R  to RESELECT items to be updated
         D  to DELETE records

Edit control

         O  to set Order of access (Order is RANDOM)
         F  to set conFirmation display (Confirmation is ON)
         L  to switch Log setting (LOG is ON)

If finished with QUANTGEN.XMP

         C  to CHANGE an existing file
         N  to start a NEW file.
         Q  to QUIT program.
>
```

Note here that an additional command (**R**) appears in the first group of options. This command makes it possible to reselect items for modification in subsequent records.

If a record for EGO cannot be found, the following display appears:

```
WARNING.  No record in Data File for ID077.
Add a record for this individual? (Y/N)
>
```

ALTERDAT automatically switches from EDIT to ADD mode if **Y** is selected. Otherwise, a request for a PID reappears, followed by an EDIT display, etc.

Command A -- Add new records to a Linked Single-entry Data File

The ADD procedure is very similar to EDIT, except that ALTERDAT will not permit addition of records with IDs that duplicate those already in a Single-entry Data File (this restriction does not apply to Multiple-entry files).

Command D -- Delete records from a Linked Single-entry Data File

Although records be may selected for deletion while running in Interactive Mode, final deletion is done as a batch process. That is, whenever deletion is specified, a temporary Update File named **<filename>.DEL** is created and used in turn as input to program DELRECS that must be run separately to incorporate these changes permanently into the Data File selected. If the temporary Update File already exists, the following sequence appears:

```
QUANTGEN.DEL already exists; overwrite it?  Y/[N]
>N

Please rename QUANTGEN.DEL before attempting to delete individuals
from QUANTGEN.XMP
```

When an appropriate Update File is available, the following sequence appears:

```
Last entry in QUANTGEN.XMP is ID070                          (DELETE Record)
--------------------------------------------------------------------------------

Enter individual's Permanent ID, or RETURN to change mode of entry.
>ID069<-   Ego ID
```

and

```
 ID: ID069                                                   (DELETE Record)
--------------------------------------------------------------------------------
ADA  Phenotype        - B
APOB Pvu2a Genotype   -
Tot. Ser. Cholesterol - 103
APOAI Level           -  99.2

Enter "D" to DELETE record, or "X" to exit without deleting.
>
```

When all deletions have been entered, ALTERDAT is ended (with the **Q** command), and program DELRECS is run to complete the process.

The next three commands control settings for various editing features of ALTERDAT:

Command O -- Set order of access

Entry of **O** at the command line toggles between random and sequential order of accessing items on records to be modified. Although the default is random order, the most convenient order depends on the number and regularity of changes to be made:

RANDOM Order

When items to be changed occur in a sporadic pattern (for example, when errors in a small number of records require correction), a random order of access may be preferable. Each field to be changed or added is represented as a line on the screen consisting of a numbered item label, followed by data space marked off by angle brackets. Each time an item is to be edited, its number is entered, and the item modified. When editing is finished, a request for another ID is given. Example displays above are all set to Random Order.

SEQUENTIAL Order

On the other hand, when a fixed set of items is to be changed repeatedly (typically when new records are being entered into the file), entry of data items in sequential order is likely to require the fewest keystrokes. Each field to be changed or added is represented as a line on the screen consisting of an numbered item label, followed by data space marked off by angle brackets. The cursor moves automatically to a new line when entry of values for each item is completed. Entry of RETURN as the initial character of any field leaves the current value unchanged (important in the carry-over feature); entry of a single blank as the initial character will erase the entire field. When all items have been completed, or if the character **]** (right square bracket) has been entered in the initial position of any field, either (a) the new values will be redisplayed for confirmation, or (b) a request for another ID is given, depending on the Confirmation setting (see below). An example of a Sequential Edit display follows:

```
 ID: ID069                                                (EDIT Mode: Seq Order)
--------------------------------------------------------------------------------
Enter items below as prompted, one or more spaces to blank an item, RETURN
for next item, or "]" for next step.

  1. ADA  Phenotype       >B   <
  2. APOB Pvu2a Genotype  >    <
  3. Tot. Ser. Cholesterol>103<
  4. APOAI Level          > 99.2<
```

Here, the cursor is positioned at the first character of the first item (ADA Phenotype) selected for modification. Entry of any printable character except **]** will replace the current contents of the item (here, B); entry of RETURN moves the cursor to the next item; entry of **]** or replacement of the last item moves the cursor to the command line, at which point entry of **S** saves the record, entry of **E** switches the program to Random Order for purposes of editing, or entry of **X** results in a request for another record to edit.

Command F -- Set confirmation status (Sequential Order only)

Entry of an **F** at the command line toggles an ON/OFF setting of the confirmation display. With the order of entry set to Sequential and Confirmation ON (the default), a newly edited or added record is redisplayed for inspection, and the opportunity is provided to make additional changes, save the entries as they appear, or to quit without saving and go on to the next record. With confirmation OFF, redisplay is suppressed and RETURN alone saves the record when editing is done (command **S** is required to save the record when Confirmation is ON). The default ON state is renewed each time a new file is selected for modification.

Command L -- Set the Log (Interactive Mode only)

Entry of an **L** at the command line toggles the generation of an optional file that preserves a record of all operations performed during the current interactive ALTERDAT session. This feature is not

needed in Batch Mode, since the update file **<FILENAME>.UPD** serves the same purpose. The default ON/OFF status of this feature is set with the Log variable in the Data File Descriptor Record (see note **p** in the Notes on DBFILES Record Formats section in Appendix A).

The next three commands determine the course of action to be taken once modification of the current file has been completed:

Command C -- Change to another file for modification

Entry of **C** here results in the following display which gives a brief summary of actions taken during the current session:

```
 Update Summary for QUANTGEN.XMP (File  1)

 Records Modified:    1
            Added:    0
          Deleted:    0

Press RETURN to continue.
>
```

This is followed by a return to the file selection display.

Command N -- Create a new Linked Single-entry Data File

Entry of **N** produces the update summary display shown above, followed by the following display:

```
                                                          (New File Creation)
------------------------------------------------------------------------------
 1.  Unlinked File
 2.  EXAMPLE

Enter number of file group to which data file will belong.
>2
```

Here, the Example Directory has been selected. When a directory has been specified at this point, ALTERDAT assumes that the file to be created will be linked to the Master File associated with the directory . It is also assumed that an appropriate Code File has been created prior to running ALTERDAT. In this case the following display appears:

```
Enter the name of the New Data File to start (without extension).
>TEST1

Enter the name of the Code File for TEST1.XMP or RETURN if TEST1.CDE
>

Is TEST1.XMP a Multiple-entry Data File?  [Y]/N
>n
```

Since a linked Data File has been specified, the file extension of the Master File (here, **.XMP**) is automatically provided, although the name of this Data File may or may not have been added to **DBFILES** (see program DBFILER in this Chapter). *Important: the first entry in the Code File must be EGO Permanent ID with Standard Mnemonic* **EGO** *or* **ID**.

A Standard Item List then appears, and items may be chosen for entry with or without carry-over, etc., in the same manner as in ADD mode. Records may not be entered into a Linked Data File for individuals not yet in the Master File. Starting a new Data File may be done only in Interactive Mode, and the Delete function is not an option.

Command Q -- Quit the program

The **Q** command terminates the program, at which time the following summary display appears:

```
                                                              (Interactive Mode)
-------------------------------------------------------------------------------

 Update Summary for QUANTGEN.XMP (File  1)

 Records Modified:    0
            Added:    0
          Deleted:    1

Press RETURN to continue.
>
```

Upon entry of RETURN, the final screen display appears:

```
*******************************************************************************
*                                                                             *
*     ALTERDAT is finished.                                                   *
*     To complete the process of changing QUANTGEN.XMP,                       *
*        it will be necessary to run the program DELRECS.                     *
*     A log of this session's updates can be found in alterdat.log            *
*     Output is in directory /h1/population/bdyke/example32/                  *
*        QUANTGEN.DEL                                                         *
*                                                                             *
*     Be sure to rename output files before rerunning this program.           *
*                                                                             *
*******************************************************************************
```

Here, a record was deleted, so a reminder is given to run program DELRECS to complete the deletion process, however, other notices may be be included in this display: If a new Linked Data File has been created, a reminder to run program DATALINK appears; if ALTERDAT was run in Batch Mode, a reminder is given to run program SETDATA to add changes to the Data File.

Linked Multiple-entry Data Files

If the file to be modified is identified as a linked **Multiple-entry Data File**, commands governing the editing process are slightly different from those shown previously in that they include additional operators which make it easier to make simultaneous changes to all records belonging to a single individual. ALTERDAT begins its interactive data entry cycle for modification of a linked Multiple-entry Data File with the following options display:

```
Updating SAMPLES.XMP                                        (Interactive Mode)
--------------------------------------------------------------------------------

Enter - [E] to EDIT record
        A  to ADD record
        D  to DELETE record

Edit control

        O  to set Order of access (Order is RANDOM)
        F  to set conFirmation display (Confirmation is ON)
        L  to switch Log setting (Log is ON)
        T  to set Template when adding records (Fill from ID's prior record)
        B  to Beep if adding the first record for an individual (Do NOT Beep)

When done with SAMPLES.XMP

        C  to CHANGE an existing file.
        N  to start a NEW file.
        Q  to QUIT program.
>
```

Note that the first three commands shown (**E**, **A** and **D**) are identical to those that appear when a Single-entry file has been selected:

Command E -- Edit Linked Multiple-entry Data File records

As is the case for Single-entry files, entering **E** (EDIT) results in a Standard Item List from which items to be modified are selected so that only those that must be changed will appear on the Edit screen. Note, however, that in this case, the instruction lines contain an additional **D** operator. Changes made to an item marked with the **D** operator on *one* of EGO's multiple records, are duplicated automatically in the same item on *all the other* records associated with that individual. This feature makes it unnecessary to make the same change to the same item in each of a series of multiple records for the same individual.

```
 * 1 Sample Number         * 4 Sample Status         * 7 Comments
 * 2 Date Taken            * 5 Animal Status
 * 3 Vial Type             * 6 Project Code
--------------------------------------------------------------------------------
  ALTERDAT    Items to be modified            File - SAMPLES.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to items to be modified. Precede number with "H"
to carry over entry of item from one record to the next. Enter "A" (or "HA")
to include all items, "D" to duplicate change of same item in all of EGO's
records, [C] to continue.
>
```

The Duplicate and Hold-over functions may be used simultaneously, although the combined effect can be confusing, making it imperative to check that records are configured properly before beginning a long ALTERDAT session in which many records are entered or modified. This is particularly important when running in Interactive Mode where records are changed directly without an intervening step.

Choice of items to be edited is followed by a request for a PID, after which follows an Edit display:

```
 ID: ID069     (Record 3/3)          Page  1/ 1     (EDIT Mode: Random Order)
--------------------------------------------------------------------------------
  1. Sample Number        >W250<
  2. Date Taken           >19980414<
  3. Vial Type            >TYG<
  4. Sample Status        >    <
  5. Animal Status        > <
  6. Project Code         >HLM<
  7. Comments             >New map<

Enter number of item to be modified, "S" to SAVE, "+/-<n>" to advance
through entries, or "X" to EXIT without saving changes for current record.
>
```

This display is slightly different from that associated with the Single-entry file. The most recently entered record is displayed initially (**Record 3 of 3** in the example above). The earlier records for this individual are accessed by entering a **-** (minus sign). If an earlier record is currently being shown on the screen, entry of **+** will display the next most recent record. Entry of **-<n>** or **+<n>**, will result in the display the of record <n> steps prior or subsequent, respectively, to the currently displayed record.

Command A -- Add records to a Linked Multiple-entry Data File

The ADD procedure for Linked Multiple-entry files is similar to that for Linked Single-entry files, except that in this case, ALTERDAT permits addition of records with IDs that duplicate those already in the file, and the Duplicate feature is not provided.

Command D -- Delete records from a Linked Multiple-entry Data File

The delete procedure for Multiple-entry files is similar to that for Single-entry files, except that provision is made to delete all of Ego's multiple records with a single command (the **DA** command):

```
 ID: ID069     Record 3/3                                      (DELETE Record)
--------------------------------------------------------------------------------
Animal ID             - ID069
Sample Number         - W250
Date Taken            - 19980414
Vial Type             - TYG
Sample Status         -
Animal Status         -
Project Code          - HLM
Comments              - New map
INTERNAL POINTER      -    95

Enter "D" to DELETE record, "DA" to DELETE ALL of EGO's records, "+/-<n>"
to advance through multiple records, or "X" to EXIT without deleting.
>
```

Again, when all deletions have been entered, ALTERDAT is ended (with the **Q** command), and program DELRECS is run to complete the deletion process. DELRECS automatically adjusts the pointers to internally linked records when deletions are made in a linked Multiple-entry Data File (see note **o** in the Notes on DBFILES Record Formats section in Appendix A).

Two of the commands controlling editing features of ALTERDAT (commands **O** and **L**) operate identically with Single- and Multiple-entry files. The confirmation command (**F**) is slightly different, in that it is unneccesary to re-enter IDs for each record of a multiple record set when confirmation is off. Two additional commands (**T** and **B**) unique to editing Multiple-entry files are provided:

Command T -- Set template when adding new records

When adding a record to a multiple-entry record set, the default initial state is for all its items to be blank on the new record. The template function makes it possible to change this initial state by filling in items on the new record with data taken from the most recently entered record in the set. Thus, the template function is similar to the hold-over feature (Commands **H** and **HA**) and the Duplicate feature (Command **D**) available when selecting items for modification in that it can eliminate the need to re-enter identical information on more than one record during a session. There are, however, some subtle differences between the three functions:

	Hold-over (Command H)	**Duplicate (Command D) EDIT Mode only**	**Template (Command T)**
File type affected:	Single- and Multiple-entry	Multiple-entry only	Multiple-entry only
Records affected:	Any record saved during session	All records in set	Any new record added to set
Initial content:	Blank	Not predetermined	Data from previous entry in set
Content of subsequent records	Data from previous entry	Data from initial entry	Data from previous record in set

The Hold-over and Template functions may be used simultaneously, although the combined effect can be confusing, making it imperative to check that records are configured properly before beginning a long ALTERDAT session in which many records are entered or modified. This is particularly important when running in Interactive Mode where records are changed directly without an intervening step.

Command B -- Set audible warning when adding first record in new set

Command **B** toggles an auditory signal that notifies the user that there is no previously entered record for the current individual.

Command N -- Create a new Linked Multiple-entry Data File

Creation of a Multiple-entry Data File that will be linked to a Master File proceeds in much the same way as with the Single-entry case described above. Two differences are that more than one record per individual is permitted in the file, and an Internal Pointer (with value **0**) is automatically appended to each record created. ALTERDAT output file **<filename>.<ext>** must then be used as input to program DATALINK to complete the linking process.

UNLINKED DATA FILES

When the **Unlinked** category has been selected from the Initial File List, a list of unlinked data files is displayed. A file is then selected from this list, or alternately, a file name is supplied, and the file is identified as a Single- or Multiple-entry Data File.

Unlinked Single-entry Data Files

Linked Data Files require that EGO PID be the first item on the record. This is not a requirement for unlinked files, which need not contain family data. However, ALTERDAT requires that at least one item on each record contain an identifier, whether or not it is a PID. *The width of the item selected as an identifier must be no greater than 9 characters, however.*

```
   1 EGO Permanent ID        3 Sire's Permanent ID     5 Dam's Permanent ID
   2 Tot. Ser. Cholesterol   4 Tot. Ser. Cholesterol   6 Tot. Ser. Cholesterol
--------------------------------------------------------------------------------
  ALTERDAT    Identification field            File - merge.out
--------------------------------------------------------------------------------
Enter the number corresponding to the item above that uniquely identifies
each record (usually a Permanent ID, Sequential ID, or Sample No., etc.).
>1
```

Here, Ego PID has been selected as the identifier. Although it is in the first position in the example, it need not be. In the case of a Single-entry Data File, the identifier must be unique to each record, and once it has been selected, records are sorted by the identifier so that the file can be searched efficiently.

Commands E, A, and D -- Edit, Add and Delete records in an unlinked Single-entry Data File

Once an identifier has been selected and the file is sorted, ALTERDAT Edit, Add and Delete functions are virtually identical to those described above for Linked files.

Commands O, F, and L -- Edit controls for an unlinked Single-entry Data File

Setting order of access, confirmation status and edit log for unlinked files is the same as that for Linked files.

Command N -- Create a new unlinked Single-entry Data File

Creation of a Single-entry Data File that will not be linked to a Master File proceeds identically to the procedure described above for Linked files, except a full file name (with extension, if there is one) must be supplied before data entry begins.

Unlinked Multiple-entry Data Files

In a Multiple-entry Data File, more than one record carries the same identifier. ALTERDAT preserves the original order of these records.

Commands E, A, and D -- Edit, Add and Delete records in an unlinked Multiple-entry Data File

Once an identifier has been selected and the file is sorted, ALTERDAT Edit, Add and Delete functions are virtually identical to those described above for Linked files.

Commands O, F, L, T and B -- Edit controls for an unlinked Multiple-entry Data File

Setting order of access, confirmation status, edit log, template and audible warning for unlinked files is the same as that for Linked files.

Command N -- Create a new unlinked Multiple-entry Data File

Creation of a Multiple-entry Data File that will not be linked to a Master File proceeds identically to the procedure described above for Linked Multiple-entry files, except that no Internal Pointer is appended to records as they are created.

Notes:

- The number of multiple records for each individual is limited to 500 for the Unix and Macintosh implementations of PEDSYS, 100 for the MS-DOS version.

- Reserved mnemonics for two items control special features (Random Order only):

  An item with the first and second mnemonics **DATE** and **UPDATE**, respectively will automatically be assigned today's date from the system calendar, eliminating the need for keyboard entry of this information. *If these mnemonics are assigned, automatic update occurs even if the item has not been selected for modification.* The item may have any label, although common ones used are simply **Date of Update**, **Edit Date**, etc.

  Editing an item with the single mnemonic **COMMNT** invokes a special instruction:

```
ID - ID069     (Record 2/2)         Page 1/1      (EDIT Mode; Random Order)
--------------------------------------------------------------------------------
1. Sample ID Number      >5582<
2. Sample Type           >R<
3. Date Taken            >820811<
4. Vial Type             >TYG<
5. Sample Status         >   <
6. Animal Status         > <
7. Diet                  >CH<
8. Project               >H  <
9. Comment               >                  <

Enter "R" to REPLACE, or "A" to APPEND to Comment.
>r

Enter item:  (one or more spaces will blank item)
Comment              >                  <
```

The **R** command simply replaces any existing comment with a new entry. The **A** command, on the other hand, makes it possible to append information to a previously entered comment. The item may have any label, although common ones are **Comment**, **Notes**, etc. The Comment entry will be truncated if it extends beyond the item field width boundary. If this feature is unwanted, the reserved mnemonic should be avoided (for example, **NOTES**, or **COMM**, etc. might be substituted).

- When running in Batch Mode, ALTERDAT saves changes and additions in file **<filename>.UPD**, which is used in turn as input to program SETDATA. Because deletion of records may require adjustments of the pointers upon which the PEDSYS file structure depends, IDs of records to be deleted are saved by ALTERDAT in file **<filename>.DEL**, which is used in turn as input to program DELRECS (the same program used to delete records from Master Files). DELRECS must be run after records have been selected for deletion in either Batch or Interactive Mode.

- ALTERDAT usually creates Update Files **<filename>.UPD** and/or **<filename>.DEL** each time it is run with the same Data File. If Update Files with these names already exist at start-up, an opportunity is provided to exit the program and archive them, or to overwrite them.

- If both deletions and batch additions have been made (that is, if ALTERDAT has produced both **<filename>.UPD** and **<filename>.DEL)**, both SETDATA and DELRECS must be run to complete the update process. *SETDATA should be run before DELRECS*.

Output:

File **<filename>.UPD** contains the following information:

```
UPDATE <recno>
<ID> <mne1> <mne2> <mne3> <mne4> <newvalue>
UPDATE <recno>
<ID> <mne1> <mne2> <mne3> <mne4> <newvalue>  ...

NEW ENTRY
<item1><item2>...<itemn>
…
```

Where records containing the words UPDATE and NEW ENTRY are descriptor records that indicate the type of change represented in the record immediately following. Also included in the UPDATE descriptor record is **<recno>**, the number of the Data File record to be changed. The next record begins with **<ID>**, the ID of the Data File record to be changed. This will be a Permanent ID Number if the file is linked, or it will be the item designated as the identifier, if the file is unlinked. The ID is followed by all four mnemonic words identifying the item, and finally the new value to be added or substituted. A NEW ENTRY record contains values of all items that are to be included in the record. If the file is linked, the first item will be the PID of the individual for which the record is to be added. If the file is unlinked, one of the items will have been designated as an identifier.

Records in **<filename>.DEL** contain two IDs of records to be deleted. The first is an **ID** as defined above, and the second a **recno** representing the record's position in the file.

Files:

Input: When CHANGING file contents: Any Data File having an associated Code File, and **CODE.STD**. If the Data File is linked, a Master Pointer File and Data Pointer File are required.

When CREATING new files: An appropriately named Code File and **CODE.STD** are required. If the Data File created is ultimately to be linked to a Master File, a Master Pointer File is also required.

Output: When CHANGING file contents: (Batch Mode): **<FILENAME>.UPD** and/or **<FILENAME>.DEL** (Interactive Mode): A modified input Data File, and (optionally) **ALTERDAT.LOG**.

When CREATING new files: (either Batch or Interactive Mode) A Data File named **<FILENAME>.<EXT>** whose Code File will have been created prior to data entry.

ALTMASTR

Program ALTMASTR is a data entry and editing program for Master File records, and is the first program used in a two-step process required to make changes in Master Files from keyboard data entry.

Introductory Display:

```
PEDSYS program ALTMASTR - Copyright 1992, 1999 SFBR

ALTMASTR is an editing program for Master File records. Changes made with this
program are kept in one or more temporary Update Files used in turn as input
to SETMASTR and/or DELRECS, programs that incorporate them permanently into
the original Master File.

Note:  An appropriate Master Code File must be available to this program.

Enter - [C] to make CHANGES to an existing Master File
         S to START new Master File
         Q to QUIT program
>
```

Changing an existing Master File

When choice **C** for CHANGES is selected, a Standard File List Display showing only Master Files appears:

```
 1.  Example Master File

Enter the number corresponding to the file to change.
>1
```

If passwords have been included in the Master File Descriptor Record of DBFILES (see note **h** in the Notes on DBFILES Record Formats section in Appendix A), a request for entry of a password will appear on the screen. When a correct password is entered (or if no password is required), the following display is shown:

```
Enter - [E] to EDIT Master File records
         A to ADD new records to Master File
         D to DELETE records from Master File
         X to EXIT program
>e
```

Command E -- Edit Master File records

Entering **E** (EDIT) results in a Standard Item List:

```
 * 1 Birth Date (YYYYMMDD  * 4 Sex (M, F or U)       * 7 Mate's Permanent ID
 * 2 Sire's Permanent ID   * 5 Exit Date (YYYYMMDD)    8 PEDIGREE NUMBER
 * 3 Dam's Permanent ID    * 6 Exit Code               9 GENERATION NUMBER
--------------------------------------------------------------------------------
  ALTMASTR    Items to be changed             File - MASTER.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to items shown above that are to be changed.
Precede item number with an "H" to carry over change of item from one record
to the next. Enter "A" (or "HA") to change all items. Enter [C] to continue
to next step.
>
```

Note that a number of items normally found in a Master File are not presented in the display and cannot be modified. These are EGO Permanent ID which is a fixed identifier of each record in the file, as well as Sequential IDs and other items derived when programs INDEX or SETMASTR are run.

Items may be selected in the usual way so that only those that must be changed will appear on the Edit screen. Additionally, preceding numbers with an **H** (for "hold") carries over the previously entered value to the next record to be changed or added. This feature makes it unnecessary to re-enter duplicate values on records of more than one individual. The value displayed in the space provided for the item on the data entry screen will be added to the record unless it is changed or erased. The new change or erasure will then be passed to subsequent records selected for alteration or addition.

- *Warning: The carry-over feature should be used with care, as it is easy to inadvertently over-write valuable information.*

Once items have been selected, ALTMASTR begins its data EDIT cycle. First, a request appears for the PID of the record to be changed:

```
Last entry in file is ID130

Enter Permanent ID of individual for which items are to be changed, or RETURN
to change mode of entry.
>ID069<
```

When this ID is entered the following display appears, with the cursor positioned at the first item to be modified:

```
Enter items below as prompted, one or more spaces to blank an item, RETURN
for next item, or "]" to advance to next step.

  >ID069<-Ego's PID
  >19961013<-Birth Date (YYYYMMDD)
  >ID054<-Sire's Permanent ID
  >ID068<-Dam's Permanent ID
  >M<-Sex (M, F or U)
  >        <-Exit Date (YYYYMMDD)
  >0<-Exit Code
  >     <-Mate's Permanent ID
```

The cursor moves automatically to a new line when entry of values for each item is completed. Entry of RETURN as the initial character of any field leaves the current value unchanged (important in the carryover feature); entry of an initial blank character will erase the entire field. When all items have been completed, or if an initial **]** (square bracket) has been entered in the initial position of any field, the new values will be displayed for confirmation, and the opportunity is provided to save (by entering **S**), edit (by entering **E**), or skip them altogether (by entering **X**). A request for another PID is then displayed, etc.

Note: To change an existing EGO PID, a new record for EGO must be entered and the old record deleted using the ADD and DELETE features of ALTMASTR.

Command A -- Add new records to a Master File

Procedures for adding records are similar to those described above for editing records, except that an error message appears if an attempt is made to enter the PID of an individual already in the Master File.

Command D -- Delete records from a Master File

The **D** command produces a request for the PID of the record to be deleted. When this ID is entered, the contents of the record are displayed for review:

```
ID - ID069
 1.   Birth Date (YYYYMMDD)     19961013
 2.   Sire's Permanent ID       ID054
 3.   Dam's Permanent ID        ID068
 4.   Sex (M, F or U)           M
 5.   Exit Date (YYYYMMDD)
 6.   Exit Code                 0
 7.   Mate's Permanent ID
 8.   PEDIGREE NUMBER               1
 9.   GENERATION NUMBER             3

 Delete this individual?  [Y]/N
>
```

This confirmation is required before the record is finally selected for deletion, and a request for another PID is displayed, etc.

Starting a new Master File:

Before ALTMASTR can be run, a Code File containing all items required for a Master File must have been created (see Chapter 4, CREATING A PEDIGREE DATABASE). This Code File must be named **CODE.<EXT>**, where **<EXT>** is an extension identifying the population for which the Master File is to be created.

The following instruction appears on the screen if START Mode is selected:

```
File UPDATE.<ext> will be produced and used to create a new MASTER.<ext>,
where <ext> is a three-character file extension used as a population iden-
tifier.

>   <- Enter the identifier for this population.
```

If a Code File with the extension entered here does not exist, the following error message appears:

```
ERROR. CODE.DOG does not exist.  Create an appropriately named Master Code
File, or if one exists, use the correct identifier. Enter RETURN to continue,
or "Q" to quit program.
>
```

If a Master File with the same extension already exists, the following message appears:

```
ERROR.  MASTER.XMP already exists.  Delete or rename this file, or use
another identifier.  Press RETURN to continue, or "Q" to quit program.
>
```

A Standard Item List is used, and items may be chosen for entry with or without carry-over in the same manner as in CHANGE Mode (see above).

The following display appears upon exiting the program after all changes are made:

```
*****************************************************************************
*                                                                           *
*            To complete the process of changing MASTER.<ext>,              *
*             it will be necessary to run program SETMASTR.                 *
*                                                                           *
*****************************************************************************
```

and/or

```
*****************************************************************************
*                                                                           *
*            To complete the process of changing MASTER.<ext>,              *
*             it will be necessary to run program DELRECS.                  *
*                                                                           *
*****************************************************************************
```

Notes:

- Parental records must be entered prior to entry of offspring.

- ALTMASTR saves changes and additions in file **UPDATE.<EXT>**, which is used in turn as input to program SETMASTR. Because deletion of records requires extensive adjustments of the pointers upon which the PEDSYS file structure depends, a somewhat different procedure is followed: Deletions are saved by ALTMASTR in file **DELETE.<EXT>**, which is used in turn as input to program DELRECS.

  ALTMASTR creates files **UPDATE.<EXT>** and/or **DELETE.<EXT>**, but the program will not run if files with these names are present at start-up. This is because the existence of these files implies that SETMASTR or DELRECS, the ensuing programs in the two-step Master File update process, have not been run successfully: Normally, SETMASTR and DELRECS eliminate files **UPDATE.<EXT>** and **DELETE.<EXT>**, respectively, by converting them to backup files **UPDATE.OLD** and **DELETE.OLD**, after permanent changes have been made to **MASTER.<EXT>**. Presence of original Update Files results in the following error message on the screen after entry of the password:

```
WARNING.  UPDATE.<ext> (or DELETE.<ext>) already exists.  Program SETMASTR (or
DELRECS) has not been run successfully since the last time ALTMASTR was used.
If you continue without running SETMASTR (or DELRECS) first, the contents of
UPDATE.<ext> (or DELETE.<ext>) will be destroyed.

Enter "C" to CONTINUE or "X" to EXIT program.
>
```

- SETMASTR should be run before running DELRECS.

Output: File **UPDATE.<EXT>** contains the following information:

```
UPDATE
<PID> <mne1> <mne2> <mne3> <mne4> <newvalue>
UPDATE
<PID> <mne1> <mne2> <mne3> <mne4> <newvalue>
...

NEW ENTRY
<PID><item1><item2>...<itemn>
...
```

where records containing the words UPDATE and NEW ENTRY indicate the type of change represented in the record immediately following. An UPDATE record begins with the PID of the Master File record to be changed. This is followed by all four mnemonic words identifying the item, and the new value to be added or substituted. A NEW ENTRY record begins with a PID, followed by values of all items in the order they are to be included in the record.

File **DELETE.<EXT>** contains only PIDs of records to be deleted.

Files:

Input: In CHANGE Mode: Any Master File having an associated Code File and Master Pointer File, and **CODE.STD**.

In START Mode: An appropriately named Code File and **CODE.STD**.

Output: **UPDATE.<EXT>** and/or **DELETE.<EXT>**.

Program Limits:

A maximum of 500 individuals may be ADDED to the Master File during any one session.

ANCESTOR

Program ANCESTOR identifies the closest common ancestors of all or a subset of probands, and constructs the minimal pedigree linking ancestors and their proband descendants.

Two sources of input are required. One is a file from which pedigrees are generated, which may be either a Master File or an indexed pedigree file consisting of records containing IDs for Ego and Ego's parents, Sex, etc., along with an associated Master Pointer File. The other is a list of 2 to 200 probands entered from either the keyboard, or from a Proband Input File (a file consisting of a list of either Sequential or Permanent IDs). Output consists of files containing (a) a list of ancestors common to a specified number of probands, (b) a single pedigree linking all common ancestors with their proband descendants, and (c) a pedigree for each ancestor and his or her proband descendants.

Introductory Display:

```
PEDSYS program ANCESTOR - Copyright 1992, 1999 SFBR

ANCESTOR identifies the closest common ancestors of all or a subset of
probands, and constructs the minimal pedigree linking ancestors to their
proband descendants.

IMPORTANT: The file from which the pedigree is generated must be an indexed
           pedigree file with an associated Master Pointer File.

 1.  Unlinked file
 2.  age.out
 3.  Example Master File

Enter the number of the pedigree file to use, or "Q" to quit.
>1
```

ANCESTOR automatically identifies pedigree files by searching for standard mnemonics identifying Ego, Ego's parents, and sex in the Code File of the Data File selected. For example, the Code File associated with **age.out** (choice **2** above) has been found to contain these mnemonics, and the Example Master File (choice **3**) is by definition a pedigree file.

```
Use of an unlinked pedigree file requires that the file be in Master File
format; that is, it must be indexed and have associated with it a Code File
and a Master Pointer File (such as INDEX.MPT).

Enter the name of the pedigree file to use.
>age.out

Enter name of the Code File for index.out, or RETURN for age.cde.
>

Enter the name of the Master Pointer File.
>MPOINT.XMP
```

Here, the unlinked file category (choice **1**) and file **age.out** have been selected (this file could have been selected directly as choice **2**). If a Master File has been selected its associated pointer file is automatically used. For an unlinked file to run successfully, it too must contain these standard pedigree items. The program also searches for a Master Pointer File (identified by a matching file name and the extension **.mpt**). In the example above, **age.out** was created directly from the Example Master File so that pointer file **MPOINT.XMP** could be used. If no Master Pointer File is available, it may be necessary to use the unlinked pedigree file as input to program INDEX with the option to create a Master Pointer File selected. Failure to find a Master Pointer File results in a request for another file name.

When a Master File or indexed pedigree file has been selected, an Item List Display appears, from which items may be selected for inclusion in the output files generated by the program:

```
   1 EGO SEQUENTIAL ID       9 PATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   2 SIRE'S SEQ             10 MATERNAL SIBSHIP SIZ   18 Exit Code
   3 DAM'S SEQ              11 FULL SIBSHIP SIZE      19 Mate's Permanent ID
   4 FIRST OFFSPRING SEQ    12 EGO Permanent ID       20 PEDIGREE NUMBER
   5 NEXT PATERNAL SIB SE   13 Birth Date (YYYYMMDD   21 GENERATION NUMBER
   6 NEXT MATERNAL SIB SE   14 Sire's Permanent ID    22 EXIT-BIRTH
   7 NEXT FULL SIB SEQ      15 Dam's Permanent ID
   8 NUMBER OF OFFSPRING    16 Sex (M, F or U)
--------------------------------------------------------------------------------
  ANCESTOR    Items to be included            File - age.out
--------------------------------------------------------------------------------
Enter numbers corresponding to items shown above in the order they are to be
included. Enter "A" to include all items, or [C] to continue.
>
```

Next, the source of proband IDs must be entered from the following display:

```
Pedigrees may be constructed for 2-200 probands entered from

  1.  Keyboard
  2.  Unlinked File
  3.  age.out
  4.  Example
  5.    LOCUS Sub-directory

Enter number for the appropriate source of probands.
>1
```

Choice 1: Proband IDs entered from the keyboard.

Choice 1 (shown above) specifies that proband IDs are to be entered from the keyboard. Before keyboard entry begins, the form of the IDs (Permanent or Sequential) must be specified. ANCESTOR identifies pedigrees by starting with a proband and accumulating his or her progenitors in ascending generations. The upper generation must also be specified here:

```
Enter - [P] if probands are identified by Permanent ID
         S  if by Sequential ID
>p


Enter the maximum number of ascending generations (1-20) relative to each
proband to be included in the pedigree.
>3
```

In this example, Permanent ID has been selected, and maximum pedigree depth for each proband has been set at three generations, which is sufficient to include ancestors in the earliest generation in the Example pedigree. Next, keyboard entry proceeds as follows:

```
Enter proband's Permanent ID.  RETURN alone terminates entry.

>ID067< ID
>ID071< ID
>ID049< ID
>ID069< ID
>     < ID
```

Ancestors of all probands selected initially have been found, but it sometimes may be useful to specify individuals that are ancestors of subsets of the probands selected. This is controlled in the display that appears next:

```
A list of individuals that are ancestral to at least <n> probands will be
generated.  Enter <n>, the minimum number of proband descendants that each
ancestor must have in order to be included in the list (2-3).
>3
```

The lower number of probands is always two, while the upper bound is determined by the program's initial analysis. Here it can be seen that the maximum number of probands sharing a common ancestor is three. Choosing this number results in the following display:

```
 2 ancestors have been found that are common to at least 3 probands:

    Ancestor       No.Pro-   Av.No.
   SEQ    PID       bands    Steps

    12   ID065         3      2.00
     4   ID042         3      3.00

Enter "D" to display a list of proband descendants of an ancestor above, "N"
to generate a list of ancestors for a new subset of probands, "S" to save
only proband and ancestor files, "P" to create files containing PEDIGREES
connecting ancestors to their descendants. Enter "Q" to quit without saving.
>d
```

Command D -- Display a list of proband descendants of an ancestor

Entry of this command produces the following display from which a single ancestor Sequential ID (as specified earlier) is selected:

```
 2 ancestors have been found that are common to at least 3 probands:

    Ancestor       No.Pro-   Av.No.
   SEQ    PID       bands    Steps

    12   ID065         3      2.00
     4   ID042         3      3.00

Enter Ancestor SEQ ID from list above or press RETURN to continue.
>12
```

Here, ancestor **12** (PID **ID065**) has been selected, producing the following display:

```
Ancestor 12 (PID ID065), is common to the following probands:

    SEQ  PID

     20  ID071
     21  ID049
     22  ID069

Enter [C] to continue.
>
```

Entry of **C** or RETURN re-displays the ancestral list display previously generated.

Command N -- Generate a list of ancestors for a new subset of probands

This command produces a re-display of the following screen, from which ancestors of a new subset of probands may be chosen:

```
A list of individuals that are ancestral to at least <n> probands will be
generated.  Enter <n>, the minimum number of proband descendants that each
ancestor must have in order to be included in the list (2-3).
>2
```

Here, ancestors common to any two or more of these individuals has been selected, which then produces the following display:

```
 6 ancestors have been found that are common to at least 2 probands:

    Ancestor       No.Pro-   Av.No.
   SEQ    PID       bands    Steps

    12   ID065         3      2.00
     4   ID042         3      3.00
     8   ID045         2      1.00
    13   ID034         2      1.00
    11   ID047         2      2.00
     7   ID062         2      2.00

Enter "D" to display a list of proband descendants of an ancestor above, "N"
to generate a list of ancestors for a new subset of probands, "S" to save
only proband and ancestor files, "P" to create files containing PEDIGREES
connecting ancestors to their descendants. Enter "Q" to quit without saving.
>p
```

Command S -- Save only proband and ancestor files

This command is given if it is sufficient at this point to generate only lists of probands and ancestors, in which case the program creates the appropriate files, and terminates. Output consists of a table of ancestor IDs, numbers of proband descendants and the mean number of generational steps separating ancestors and probands (file **ancestor.tab**), and separate files containing records of all probands (file **ancesego.out**) and ancestors (file **ancesanc.out**).

Command P -- Save pedigrees connecting ancestors to their descendants

This command re-displays the list of ancestors selected, along with options for creating and saving files containing selected pedigrees connecting ancestors to their proband descendants:

```
 6 ancestors have been found that are common to at least 2 probands:

    Ancestor       No.Pro-   Av.No.
   SEQ    PID       bands    Steps

   *12   ID065         3      2.00
     4   ID042         3      3.00
     8   ID045         2      1.00
    13   ID034         2      1.00
    11   ID047         2      2.00
     7   ID062         2      2.00

Enter SEQ IDs from the list above to create files containing pedigrees that
connect a single ancestor to its descendant probands. Enter "A" to create a
single file that contains one or more pedigrees connecting all ancestors and
probands. Enter "S" to save files and exit, "Q" to quit without saving.
>a
```

For each ancestor selected, a file is created that contains records for that ancestor, all descendants among the set of probands chosen initially, and all individuals that connect the ancestor and these des-

cendants. These files are automatically named **anc<seq>.out**, where <seq> is the Sequential ID of the ancestor selected. Here, Sequential ID **12** has been selected, which specifies that the pedigree connecting this individual (PID ID065) with two descendant probands will be constructed and saved in output file **anc12.out**. If **a** is entered, as many pedigrees as necessary will be constructed to connected all six individuals listed as ancestors with their respective descendants. Entry of **S** saves the separate files containing pedigrees, plus the table, proband and ancestor files as described above. The final output screen message for the example session shown here is as follows:

```
*****************************************************************************
*                                                                           *
*     ANCESTOR is finished.                                                 *
*     Output is in directory /h1/population/bdyke/example/                  *
*        ancesall.out                                                       *
*        anc12.out                                                          *
*     ancesego.out contains probands for whom pedigrees were constructed.   *
*     ancesanc.out contains all ancestors identified by the program.        *
*     ancestor.tab contains table summarizing output.                       *
*                                                                           *
*     Be sure to rename output files before rerunning this program.         *
*                                                                           *
*****************************************************************************
```

Choice 2, 3 or 4: Proband IDs read from a file.

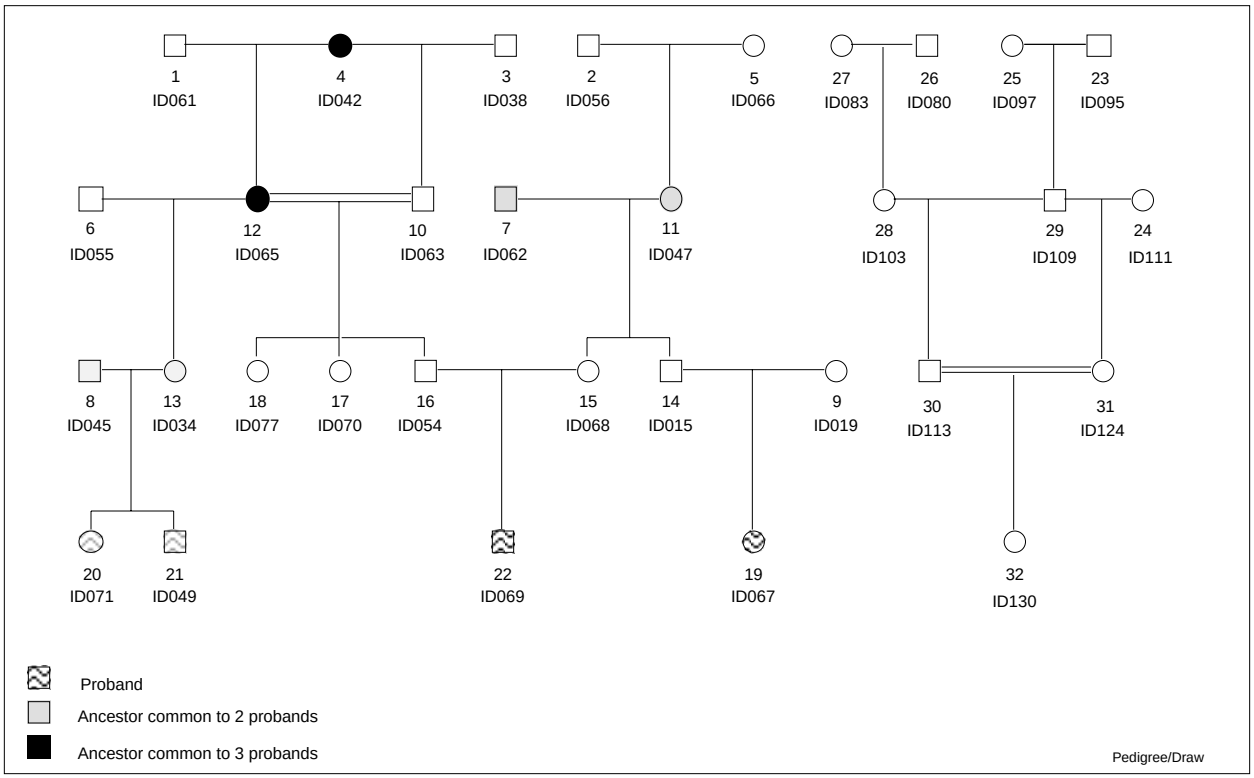
A proband file is often a subset of the input pedigree file (typically a Master File), but may be any Data File whose records contain IDs in the appropriate form (that is, Permanent or Sequential). Selection of a single item containing the proband ID is made from a Standard Item List Display:

```
   1 EGO SEQUENTIAL ID       9 PATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   2 SIRE'S SEQ             10 MATERNAL SIBSHIP SIZ   18 Exit Code
   3 DAM'S SEQ              11 FULL SIBSHIP SIZE      19 Mate's Permanent ID
   4 FIRST OFFSPRING SEQ    12 EGO Permanent ID       20 PEDIGREE NUMBER
   5 NEXT PATERNAL SIB SE   13 Birth Date (YYYYMMDD   21 GENERATION NUMBER
   6 NEXT MATERNAL SIB SE   14 Sire's Permanent ID    22 EXIT-BIRTH
   7 NEXT FULL SIB SEQ      15 Dam's Permanent ID
   8 NUMBER OF OFFSPRING    16 Sex (M, F or U)
-------------------------------------------------------------------------------
ANCESTOR     Identify PROBAND                File - MASTER.XMP
-------------------------------------------------------------------------------
Enter a number above corresponding to Proband ID.  Enter "C" to continue.
>14
```

Here, common ancestors of all sires in the Master File have been specified, and the program proceeds directly to the queries described above in **Choice 1** for keyboard entry of probands.

Notes:

- The consequences of including ancestors for different numbers of probands can be seen from the pedigree diagram below. Here, the four probands selected are in the last generation, ancestors common to only two probands are identified by shaded symbols, and ancestors common to three probands are identified by solid symbols.



- The notion of *the closest common ancestor* can be seen in the pedigree diagram, where neither individual ID056 nor ID066 are included as ancestors of probands ID069 and ID067. This is because any genetic contribution they make to the pedigree must be passed on in its entirety through their offspring ID047, who *is* designated as a common ancestor of the probands.

Files:

Input: Any Master File or indexed pedigree file and Master Pointer File.
Input Data File or console keyboard.

Output: **ancestor.tab** contains a list of ancestors, numbers of proband descendants and the mean number of generational steps separating ancestors and probands. The table in the file will be identical to the one most recently displayed on the screen.

ancesego.out contains records for all probands.

ancesanc.out contains records for all ancestors identified by the program.

anc<n>.out contains records for individuals making up a pedigree connecting a single ancestor to its proband descendants, where <n> is the ancestor Sequential ID.

ancesall.out contains records for individuals making up one or more pedigrees connecting all ancestors to their proband descendants.

ancestor.err contains proband IDs entered from an input file that are not found in the input pedigree file.

APPEND

Program APPEND joins two files whose records share the same format.

Introductory Display:

```
PEDSYS program APPEND - Copyright 1992, 1999 SFBR

APPEND joins two or more files whose records share the same format. Code
Files are compared for length and format type of each item. If discrepancies
are found, error messages are displayed, and joining of files is not per-
mitted.

1. Unlisted file
2. EXAMPLE

Enter number of the file or directory to use, or "Q" to quit.
>
```

A second standard File List is displayed next, from which the second of the two files to be joined is selected. Standard Item List Displays are not used.

After two files have been selected and joined, the following display appears:

```
<filename1>.<ext>
<fliename2>.<ext>

have been joined and written to file append.out.

Enter -    A to append an additional file
           X to exit program
>
```

Note:

- The command line shortcut with *two* input files (program name, followed by a blank, one input filename, another blank and a second input filename) may be used to start this program. Example: **append sort.out subset.out**.

Files:

Input: Two or more Master Files or Data Files (**<filename1>.<ext>** and **<filename2>.<ext>**), each having the same format. Corresponding Item Descriptor Records of all Code Files must specify identical width and data type; mnemonic and description fields need not be the same. Recursive use of output files is permitted, that is, file **append.out** may be used as input to program APPEND.

Output: **append.out** contains the joined files.

append.cde will be identical to the Code File of **<filename1>.<ext>**, the first of the input files, except that its label will read as much as will fit of**<filename1>.<ext> + <filename2>.<ext> + <filename3>.<ext> + ...**

where **<filename2>.<ext>** is the second of the two input files, etc.

BROWSE

Program BROWSE gives a direct display of records, a screen page at a time, from any PEDSYS file having an accompanying Code File. BROWSE is in some ways similar to SHOWDATA, but unlike SHOWDATA, it is *record* oriented, rather than *individual* oriented. That is, BROWSE displays records of more than one individual in a file, whereas the multiple records displayed by SHOWDATA all belong to a single individual.

Introductory Display:

```
PEDSYS program BROWSE - Copyright 1992, 1999 SFBR

BROWSE gives a direct display of records from a PEDSYS file, a screen page
       at a time.

  1.  Unlinked File
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>
```

If the **Unlinked file** category is chosen (choice **1** in the example above), a standard unlinked file list appears. However, if a **directory** containing linked files has been selected (choice **2** in the example above), an alternate Secondary File List follows. This looks much like the standard Secondary File List, except that it permits selection of up to five linked files for display:

```
Linked File Group for Extension XMP

*   1.  Example Master File
*   2.  Markers + Quant. P'types
    3.  Blood Sample Inventory
    4.  Hemoglobin Levels

More than one LINKED file may be selected for repeated sequential display.
Enter numbers corresponding to file(s) above, in the order they are to be
shown. Enter RETURN alone to continue to next step.
```

In the example above, the Example Master File and the **Markers + Quant. P`'types** file have been marked with an asterisk, indicating that they have been selected for display. The order of selection and display is usually not critical, since switching between files is cyclical. However, the Master File is frequently chosen first.

Initial Screen Page:

The display below shows the first screen page of the Example Master File. BROWSE starts initially with the first item of the first record in the upper left corner of the data display window. The display window extends to the right for as many additional complete items as will fit on the screen (typically 80 characters wide), and downward as many records as will fill the screen (see Notes below). Columns are unlocked.

```
MASTER.XMP                              Items: 21       Top record: 1/32
 (1)   (2)   (3)   (4)   (5)   (6)   (7)  (8)   (9)    (10)   (11)   (12)
 SEQ   SSEQ  DSEQ  KID1  PSIB  MSIB  FSIB NKID PSIBSZ MSIBSZ FSIBSZ  ID

   1     0     0    12     0     0     0   1     0      0      0   ID061
   2     0     0    11     0     0     0   1     0      0      0   ID056
   3     0     0    10     0     0     0   1     0      0      0   ID038
   4     0     0    10     0     0     0   2     0      0      0   ID042
   5     0     0    11     0     0     0   1     0      0      0   ID066
   6     0     0    13     0     0     0   1     0      0      0   ID055
   7     0     0    14     0     0     0   2     0      0      0   ID062
   8     0     0    20     0     0     0   2     0      0      0   ID045
   9     0     0    19     0     0     0   1     0      0      0   ID019
  10     3     4    16     0    12     0   3     1      2      1   ID063
  11     2     5    14     0     0     0   2     1      1      1   ID047
  12     1     4    16     0    10     0   4     1      2      1   ID065
  13     6    12    20     0    17     0   2     1      4      1   ID034
  14     7    11    19    15    15    15   1     2      2      2   ID015
  15     7    11    22    14    14    14   1     2      2      2   ID068

Enter [+]<n> or -<n> for new page; PID, S<n> (1-32) or =<n> (1-32) to set top
row; ">" or "<"<n> to shift window; I<n> (1-21) to set window; L to lock
columns; * to switch files; N for new file; F to find; H for help; Q to quit.
>
```

Commands:

+<n>, **-<n>** Entry of **+** or **-** displays a new screen ±<n> pages from the current page. The number of the top record on the page displayed in the upper right-hand corner of the screen is updated. Horizontal position remains unchanged. The default value of <n> is 1, so that entry of **RETURN** alone advances the display by one page. The total number of pages may be estimated by dividing the number of records by the number of records that appear on a single page (see first Note below). Page contents do not overlap, that is, the top record of the next adjacent page is set at one past the record displayed at the bottom of the previous page.

PID Entry of a valid PID (linked files only) sets the top of a new page at the corresponding record. The new page is displayed with the horizontal window position unchanged.

S <seq> Entry of **s** followed by a Sequential ID sets the top of a new page at the corresponding record. For Master Files and unlinked Data Files <seq> must be in the range 1 through <nrecs>, where <nrecs> is the number of records in the file. For Linked Data Files <nrecs> is the number of records in the Master File. The new page is displayed with the horizontal position of the window unchanged.

= <n> where <n> is an integer in the range 1 through <nrecs>, the number of records in the file. Entry of **this sequence** sets the top of a new page at the corresponding record. The new page is displayed with the horizontal position of the window unchanged. Note that for Master Files and unlinked Data Files the **=** and **S** commands are interchangeable.

><i> or **<<I>** where <i> is an integer in the range 1 through <nitems>, the number of items on the record. Entry of this sequence shifts the unlocked portion of the window horizontally by ±<i>items, without changing page position. In the example above $1 \leq \text{<i>} \leq 21$, since there are 21 items per record in the Example Master File.

I <I> where <i> is an integer in the range 1 through <nitems>, the number of of items on the record. Entry of this sequence sets the horizontal position of the item selected at the left-most unlocked

column. In the example above 1 ≤ <i> ≤21, since there are 21 items per record in the Example Master File.

L Entry of this command makes it possible to lock columns, that is, fix their position so that they remain visible despite horizontal shifts of unlocked items in the display window. (See explanation of column locking below.)

***** This command makes it possible to switch the BROWSE record display rapidly between linked files selected when the program is started. Entry of an asterisk switches the display cyclically to one of the selected files. If an individual has been chosen explicitly (by entry of either PID or SEQ) in one linked file, the top record of each of these displays will belong to the same individual, providing that there is in fact an entry for that individual in the newly selected file. A warning message will appear if there is no entry for that ID in the file. This*option can be used only when a Master File or linked Data File has been selected.*

N This command results in a return to the File List, from which a new file (or set of files) may be selected for display in the usual manner.

H Entry of **H** produces the following help display pages:

```
To set the record that will appear at top of page, enter a Permanent ID,
"S" followed by EGO Sequential ID, or "=" followed by a number representing a
record's position in the file, and then hit RETURN.

To page FORWARD through a file, enter RETURN for steps of one, precede
RETURN with "+<n>" for steps of <n> pages.  To page BACKWARD through file,
precede RETURN with "-" for steps of one, "-<n>" for steps of <n>.

To shift the window RIGHT or LEFT, enter ">" or "<" for shifts of one item,
">"<n> or "<"<n>, respectively, for shifts of <n> items.

To set the item that will appear in the left-most unlocked column, enter
"I <n>" where <n> is the item number, then hit RETURN.

Enter "L" to select columns for LOCKED display, that is, fixed in position
on the left-hand side of the screen despite horizontal movement of the dis-
play window.

Enter [C] to return to data display. Enter "P" to page.
>
```

and

```
Enter "*" to switch cyclically from one LINKED file to the next. If files
have not been preselected, initial entry of "*" presents a File List, from
which they may be chosen.

Enter "N" to select new files for display.

Enter "F" to find a specific entry for an item.

Enter "Q" to exit program.

Press RETURN to continue.
>
```

F Initiates the find function (see below).

Q Exits the program.

Column locking:

By default, all items in the record may appear in the display window in their original order, depending on the horizontal position of the window with respect to the record. This default may be overridden so that some items may retain a fixed position in the window. Entry of **L** produces the following Item List Display:

```
 * 1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ   16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE  *12 EGO Permanent ID       19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE   13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  BROWSE                                 File - MASTER.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to items shown above in the order they are to be
assigned to locked left-hand columns. Enter [C] to continue
>
```

In the example above, EGO Sequential ID and EGO Permanent ID (Items 1 and 12) have been allocated to locked columns (using common spreadsheet terminology) at the left-most side of the display. The remaining items remain unlocked. The result of this selection can be seen in the following display:

```
 MASTER.XMP                              Items: 21       Top record: 31/32
  (1)   (12)  (2)   (3)   (4)   (5)   (6)   (7)  (8)   (9)    (10)   (11)
  SEQ   ID    SSEQ  DSEQ  KID1  PSIB  MSIB  FSIB NKID PSIBSZ MSIBSZ FSIBSZ

    31 ID124    29    24    32    30     0     0   1     2      1      1
    32 ID130    31    30     0     0     0     0   0     1      1      1

Enter [+]<n> or -<n> for new page; PID, S<n> (1-32) or =<n> (1-32) to set top
row; ">" or "<"<n> to shift window; I<n> (1-21) to set window; L to lock
columns; * to switch files; F to find; H for help; Q to quit.
>
```

Columns may also be locked in sequence from the left by entering **L**, followed by an Item Number. For example, columns can be ordered in the same sequence as shown in the display above by entering **L1**, followed by **L12**.

- Items in **locked** columns form a subset of other items displayed, and are positioned in order of selection from the left-most boundary of the screen. They remain visible in their locked positions despite subsequent horizontal shifts of the remaining items selected for display. Locking is particularly useful for maintaining visibility of a PID, while shifting horizontally across long records.

- Any item not explicitly designated as locked is assigned by default to the **unlocked** display. Items in the unlocked portion are positioned in order from the right-most locked column. Visibility of

items in the unlocked portion depends on their number, the width of the unlocked segment of the window, and the position of the window as specified by the relevant command sequence.

- The total formatted width of locked items may not exceed width of the screen.
- Columns may be unlocked (that is returned to their original order) by entering **L0**.

The Find function:

The Find function can be activated in one of two ways. Entering **F** alone at the command line results in a Standard Item List with a request for the value sought for the particular item in EGO's record. When this has been entered, instructions for starting the search are given:

```
   1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ   16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE   12 EGO Permanent ID       19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE   13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 GENERATION NUMBER

Enter item number, relational (EQ or NE) and value.  (Enclose substrings in
 quotes)
>8 eq 2

Enter record number at which to start search (1-32), or RETURN to start at
current record (1)
>
```

In this example, the Master File will be searched for the first record in which the number of offspring (Item 8) has the value 2. When a record containing the item sought is found, it is displayed on the top line of the display page in the format of the version selected when the program was started:

```
 MASTER.XMP                              Items: 21       Top record: 4/32
  (12)  (1)   (2)   (3)   (4)   (5)   (6)   (7)  (8)   (9)    (10)   (11)
  ID    SEQ   SSEQ  DSEQ  KID1  PSIB  MSIB  FSIB NKID PSIBSZ MSIBSZ FSIBSZ

 ID042     4     0     0    10     0     0     0   2     0      0      0
 ID066     5     0     0    11     0     0     0   1     0      0      0
 ID055     6     0     0    13     0     0     0   1     0      0      0
 ID062     7     0     0    14     0     0     0   2     0      0      0
 ID045     8     0     0    20     0     0     0   2     0      0      0
 ID019     9     0     0    19     0     0     0   1     0      0      0
 ID063    10     3     4    16     0    12     0   3     1      2      1
 ID047    11     2     5    14     0     0     0   2     1      1      1
 ID065    12     1     4    16     0    10     0   4     1      2      1
 ID034    13     6    12    20     0    17     0   2     1      4      1

Enter "F" to find next occurrence of NUMBER OF OFFSPRING EQ   2
Press RETURN to end search.
>
```

Alternatively, the Item List display can be bypassed by entering **F**, an item number, a relational, and the value sought (all separated by blanks) which leads directly to the request for a starting record number. For example, entry of **f 8 eq 2** at the command line is equivalent to the sequence described above.

Subsequent records containing the same value can be found by entering **F** again, etc. The message **End of search** appears when the value sought is not found in subsequent records in the file. The message will also appear if the value is not found at all in the file. A substring feature permits a search for a sequence of characters enclosed in quotes that occurs at any position in the field occupied by the item. For example the substring '1985' would search for all occurrences of the year, regardless of the month or day included in the date field.

Notes:

- Partial items are not displayed. This means that it may not be possible to view items that exceed the width of the unlocked portion of the window.

- When a PID or SEQ is entered for a Multiple-entry Data File, the most recently entered record for the individual chosen is displayed as the first line on the screen. To see other records for that individual, use the **=** command with the number found in the last item of the record. This is the **internal pointer** to the next most recently entered record.

- Alternates are provided for the leading command characters specifying Sequential ID, Item and page numbers. This is because there is the possibility of conflict between the defaults and PIDs that may begin with the same characters. For example, suppose a Master File contains records of individuals identified by PIDs, all or some of which begin with the letter **S**, followed directly by a number (S10, S11, S12, etc.). In this case, there is no way to distinguish between entry of the PID **S10** and the Sequential ID Number **10**, which must be specified in the command line with a leading **S**, that is, also as **S10**. The same problem may occur if PIDs begin with command characters **I**, or **i**, and **P** or **p**. To circumvent this possibility, the following alternates may be used:

#<n>	may be substituted for	**S**<n>	or	**s**<n>
@<n>	" " "	**I**<n>	or	**i**<n>

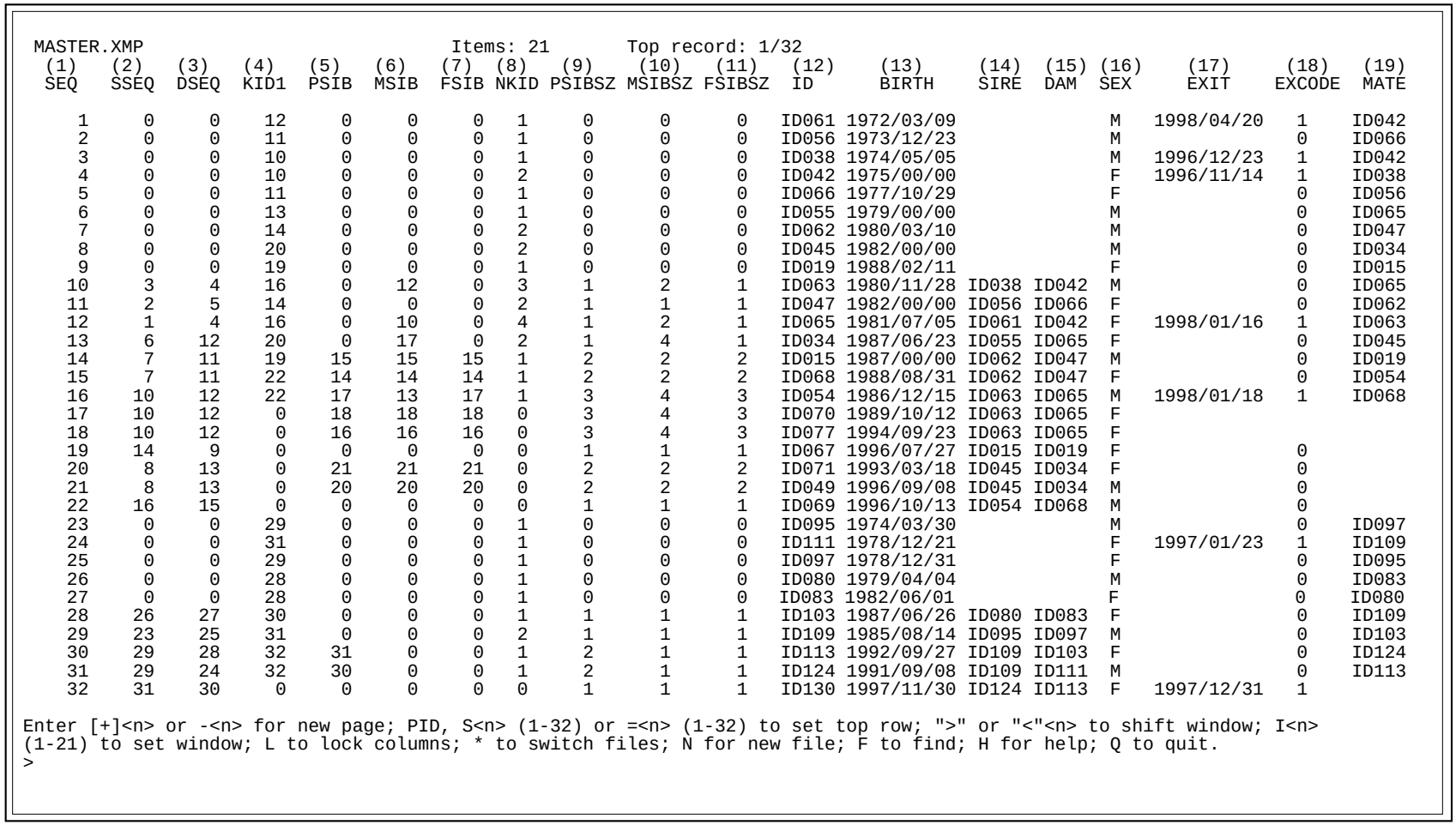
  In the example just given, specification of Sequential ID **10** would be **#10** to distinguish it from PID **S10**. Likewise, **@5** may be substituted for **i5** to specify that Item 5 be placed in the left-most unlocked column.

  The default characters were chosen to speed up command sequence entry by avoiding the use of the keyboard shift key. The alternates are available in all relevant PEDSYS programs.

- To save screen space for data, a maximum of two mnemonics per item are displayed in the header.

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **browse subset.out**.

- The display on the next page illustrates output from program BROWSE using expanded screen display settings (130 columns by 52 rows) which permit single page display of 19 items for all 32 records of the Example Master File. This contrasts with the 12 items of only 15 records that result from the 80 column by 34 row setting used for illustrations in this manual. See Chapter II, Setup and Installation for details on these capabilities.



```
MASTER.XMP                                Items: 21       Top record: 1/32
 (1)   (2)   (3)   (4)   (5)   (6)   (7)  (8)   (9)    (10)   (11)   (12)     (13)     (14)  (15) (16)    (17)     (18)   (19)
 SEQ   SSEQ  DSEQ  KID1  PSIB  MSIB  FSIB NKID PSIBSZ MSIBSZ FSIBSZ  ID      BIRTH    SIRE  DAM  SEX     EXIT    EXCODE  MATE

   1     0     0    12     0     0     0   1     0      0      0   ID061 1972/03/09                M   1998/04/20   1    ID042
   2     0     0    11     0     0     0   1     0      0      0   ID056 1973/12/23                M                0    ID066
   3     0     0    10     0     0     0   1     0      0      0   ID038 1974/05/05                M   1996/12/23   1    ID042
   4     0     0    10     0     0     0   2     0      0      0   ID042 1975/00/00                F   1996/11/14   1    ID038
   5     0     0    11     0     0     0   1     0      0      0   ID066 1977/10/29                F                0    ID056
   6     0     0    13     0     0     0   1     0      0      0   ID055 1979/00/00                M                0    ID065
   7     0     0    14     0     0     0   2     0      0      0   ID062 1980/03/10                M                0    ID047
   8     0     0    20     0     0     0   2     0      0      0   ID045 1982/00/00                M                0    ID034
   9     0     0    19     0     0     0   1     0      0      0   ID019 1988/02/11                F                0    ID015
  10     3     4    16     0    12     0   3     1      2      1   ID063 1980/11/28 ID038 ID042    M                0    ID065
  11     2     5    14     0     0     0   2     1      1      1   ID047 1982/00/00 ID056 ID066    F                0    ID062
  12     1     4    16     0    10     0   4     1      2      1   ID065 1981/07/05 ID061 ID042    F   1998/01/16   1    ID063
  13     6    12    20     0    17     0   2     1      4      1   ID034 1987/06/23 ID055 ID065    F                0    ID045
  14     7    11    19    15    15    15   1     2      2      2   ID015 1987/00/00 ID062 ID047    M                0    ID019
  15     7    11    22    14    14    14   1     2      2      2   ID068 1988/08/31 ID062 ID047    F                0    ID054
  16    10    12    22    17    13    17   1     3      4      3   ID054 1986/12/15 ID063 ID065    M   1998/01/18   1    ID068
  17    10    12     0    18    18    18   0     3      4      3   ID070 1989/10/12 ID063 ID065    F
  18    10    12     0    16    16    16   0     3      4      3   ID077 1994/09/23 ID063 ID065    F
  19    14     9     0     0     0     0   0     1      1      1   ID067 1996/07/27 ID015 ID019    F                0
  20     8    13     0    21    21    21   0     2      2      2   ID071 1993/03/18 ID045 ID034    F                0
  21     8    13     0    20    20    20   0     2      2      2   ID049 1996/09/08 ID045 ID034    M                0
  22    16    15     0     0     0     0   0     1      1      1   ID069 1996/10/13 ID054 ID068    M                0
  23     0     0    29     0     0     0   1     0      0      0   ID095 1974/03/30                M                0    ID097
  24     0     0    31     0     0     0   1     0      0      0   ID111 1978/12/21                F   1997/01/23   1    ID109
  25     0     0    29     0     0     0   1     0      0      0   ID097 1978/12/31                F                0    ID095
  26     0     0    28     0     0     0   1     0      0      0   ID080 1979/04/04                M                0    ID083
  27     0     0    28     0     0     0   1     0      0      0   ID083 1982/06/01                F                0    ID080
  28    26    27    30     0     0     0   1     1      1      1   ID103 1987/06/26 ID080 ID083    F                0    ID109
  29    23    25    31     0     0     0   2     1      1      1   ID109 1985/08/14 ID095 ID097    M                0    ID103
  30    29    28    32    31     0     0   1     2      1      1   ID113 1992/09/27 ID109 ID103    F                0    ID124
  31    29    24    32    30     0     0   1     2      1      1   ID124 1991/09/08 ID109 ID111    M                0    ID113
  32    31    30     0     0     0     0   0     1      1      1   ID130 1997/11/30 ID124 ID113    F   1997/12/31   1

Enter [+]<n> or -<n> for new page; PID, S<n> (1-32) or =<n> (1-32) to set top row; ">" or "<"<n> to shift window; I<n>
(1-21) to set window; L to lock columns; * to switch files; N for new file; F to find; H for help; Q to quit.
>
```

Files:

Input: Any Master File or Data File, and an associated Code File.

Output: Console screen.

CALC

Program CALC generates simple functions (+, -, *, /), natural and base 10 logarithms and exponentiation of numerical variables kept on individual records, and/or will add to each record new items consisting of (a) an integer corresponding to the order of the record in the file, (b) a randomly generated fraction from a uniform distribution between 0.0 and 1.0, (c) the sum, count, mean, and standard deviation of selected numerical variables on each record, (d) measurements of selected items converted between conventional units and standard Systèm International D'Unités (SI) units, and (e) Z-scores computed on previously stored or computed numerical values. Precision (that is, placement of the decimal point) of numerical output can be controlled.

Introductory Display:

```
PEDSYS program CALC - Copyright 1992, 1999 SFBR

CALC generates functions of one or more numerical variables.

  1.  Unlinked File
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter number of the file or directory to use, or "Q" to quit.
>
```

After a file has been selected from the Introductory File List, the following display appears:

```
Functions of numerical values are performed in the order listed below:

  1. Calculate +, -, *, /, ln, log, exp; random no., integer seq., SI Units
  2. Find sum, count, mean, Std. Dev. and Std. Err. of items on same record.
  3. Calculate Z-scores
  4. Set output precision (place decimal point).
  5. Quit without running program.

Enter number corresponding to function
>
```

Functions are performed in the order listed above; that is, there is no provision for returning to simple calculations such as addition and subtraction, etc., after a sum, mean and standard deviation of variables has been specified. It is not necessary to perform all functions; for example, output precision may be set without specifying prior calculations.

1. Calculating simple functions, addition of numerical sequences

Choice **1**, specifying simple calculations, results in the following display:

```
   1 EGO SEQUENTIAL ID      9 PATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   2 SIRE'S SEQ            10 MATERNAL SIBSHIP SIZ   18 Exit Code
   3 DAM'S SEQ             11 FULL SIBSHIP SIZE      19 Mate's Permanent ID
   4 FIRST OFFSPRING SEQ   12 EGO Permanent ID       20 PEDIGREE NUMBER
   5 NEXT PATERNAL SIB SE  13 Birth Date (YYYYMMDD   21 GENERATION NUMBER
   6 NEXT MATERNAL SIB SE  14 Sire's Permanent ID   *22 PSIBSZ / FSIBSZ
   7 NEXT FULL SIB SEQ     15 Dam's Permanent ID
   8 NUMBER OF OFFSPRING   16 Sex (M, F or U)
--------------------------------------------------------------------------------
 CALC Simple calculations, sequences   File - MASTER.XMP
--------------------------------------------------------------------------------
Enter first operand from the list above, (or a constant preceded by "K"), an
operator (+,-,*,/,E), then a second operand. Precede operand with "Log" or
"LN" for logs, "U" for SI Unit conversion. Enter "R" for random fractions,
"S" for integer sequence, "H" for help, [C] to continue to next step.
>10/11              <--  COMMAND
```

Commands:

Operators +, **-**, *****, **/** and **E**

A first operand is selected by entering a number corresponding to a numerical value from the list of items above, or a constant (a number preceded by a **k**). Immediately follow this operand with an operator, and follow the operator by a second operand selected in the same way as the first.

In the example above, Item 22 (PSIBSZ / FSIBSZ) has been added to those normally found on Example Master File records. This was produced by entering **9/11** in the command line, which specifies calculation of the ratio of PATERNAL SIBSHIP SIZE (Item 9) to FULL SIBSHIP SIZE (Item 11). The sequence **10/11** has also been entered in the command line, which will specify calculation of the ratio of MATERNAL SIBSHIP SIZE (Item 10) to FULL SIBSHIP SIZE.

Raising a variable to a constant power (with **E** or **e**) requires a leading **K** (**k**) before the exponent. Roots are taken by raising a variable to a fractional power (**.5** for square root, etc.). Examples: **8EK2**, **9ek-1**, **20ek.5**, etc.

Operators LOG and **LN**

Preceding an item number corresponding to a numerical variable with **LN** (**ln**) or **LOG** (**log**) causes its natural or base 10 logarithm to be appended to the end of each record. Example: the natural log of Generation number (Item 20) is specified as **ln20**.

R Entering an **R** (**r**) produces a request for an integer seed (1 - 8 digits) for a pseudo-random number generator. An item containing a randomly generated 10-digit fraction is added to the record. The seed is saved in the appropriate description field of the Code File **calc.cde**.

S Entering an **S** (**s**) adds to each record an item containing an integer (1-99999) corresponding to the record's position in the file.

H Entry of **H** (**h**) results in display of a sequence of explanatory text pages.

U Preceding an item number with **U** (**u**) results in the following display, from which constants for conversion between selected conventional measurements and Système International D'Unités standard units (SI Units) may be chosen. Converted items are added to the end of each record. For example,

entry of **u4** in the command line shown in the Item List Display of the **QUANTGEN.XMP** file below specifies conversion of Total serum Cholesterol measurements (Item 4) from mg / dl to SI Units:

```
 1 Ego ID                  3 APOB Pvu2a Genotype     5 APOAI Level
 2 ADA  Phenotype          4 Tot. Ser. Cholestero
--------------------------------------------------------------------------------
 CALC Simple calculations, sequences   File - QUANTGEN.XMP
--------------------------------------------------------------------------------
Enter first operand from the list above, (or a constant preceded by "K"), an
operator (+,-,*,/,E), then a second operand. Precede operand with "Log" or
"LN" for logs, "U" for SI Unit conversion. Enter "R" for random fractions
"S" for integer sequence, "H" for help, [C] to continue to next step.
>u4                  <--  COMMAND
```

Entry of RETURN produces the following display, from which the appropriate conversion factor may be chosen:

```
Convert Tot. Ser. Cholesterol

System        Input Units     SI Units     Conversion Constant

 1. Calories        kcal           kJ               (4.1868)
 2. Cholesterol     mg/dl          mmol/l           (0.02586)
 3. Glucose         mg/dl          mmol/l           (0.05551)
 4. Insulin         uU/ml          pmol/l           (6.00000)
 5. Triglycerides   mg/dl          mmol/l           (0.01143)

Enter number above corresponding to system to be converted TO SI units.
Precede number by "R" to convert FROM SI units.
>2
```

2. Calculating sums, etc. of variables on the same record

CALC makes it possible to calculate the sum, count, mean, standard deviation, and standard error of variables that appear on the same record. This function is typically used to compute these statistics for multiple measurements made on the same individual, when meaurements are kept on a single record per individual. The function may be accessed at one of two points in program CALC's sequence of operations. One opportunity to choose this function occurs when choice **2** in the Introductory Display (**Find sum, count, mean, Std. Dev. and Std. Err. of items on same record**) has been selected; the other occurs when choice **1** (**Calculate +, -, *, /, ln, log, exp**) has been selected from the Introductory Display at which point the following display appears after the confirmation step is finished:

```
Calculate sum, N, mean, Std. Dev. and Std. Err. of variables on the same
record? Y/[N]
>
```

However accessed, an Item List Display appears, from which items may be chosen for calculations. In the example below, Items 15 through 27, repeated measurements of Hemoglobin Level for each individual, are selected. The file containing these records was generated with program COMBINE, which extracts items from multiple records with the same identifier, and combines them on a single record:

```
   1 Ego                    10 Rec9 Sample Date      *19 Rec5 Hemoglobin Leve
   2 Rec1 Sample Date       11 Rec10 Sample Date     *20 Rec6 Hemoglobin Leve
   3 Rec2 Sample Date       12 Rec11 Sample Date     *21 Rec7 Hemoglobin Leve
   4 Rec3 Sample Date       13 Rec12 Sample Date     *22 Rec8 Hemoglobin Leve
   5 Rec4 Sample Date       14 Rec13 Sample Date     *23 Rec9 Hemoglobin Leve
   6 Rec5 Sample Date      *15 Rec1 Hemoglobin Leve  *24 Rec10 Hemoglobin Lev
   7 Rec6 Sample Date      *16 Rec2 Hemoglobin Leve  *25 Rec11 Hemoglobin Lev
   8 Rec7 Sample Date      *17 Rec3 Hemoglobin Leve  *26 Rec12 Hemoglobin Lev
   9 Rec8 Sample Date      *18 Rec4 Hemoglobin Leve  *27 Rec13 Hemoglobin Lev
--------------------------------------------------------------------------------
  CALC        Sum, N, mean, Std. Dev/Err.      File - combine.out
--------------------------------------------------------------------------------
Enter numbers corresponding to numerical values above for which a sum, N, mean,
and Std. Dev./Err. are to be calculated.  Enter [C] to continue
>
```

Once items have been selected, the following display appears:

```
It may be desirable to substitute a pooled intra-individual standard deviation
(SDs averaged across all records) in calculations of the standard error when
N, the count of individual values is very small.  Enter a maximum N for which
this adjustment will be made to the standard error. Enter RETURN to skip this
adjustment.
>2
```

Here, the adjustment has been selected for calculations of standard error for any record where the number of measurements N ≤ 2. To see the effect of the adjustment, compare the two displays below. The first shows selected results for the unadjusted standard deviation:

```
 calc.out                                Items: 32       Top record: 18/27
  (1)      (28)      (29)     (30)         (31)         (32)
  EGO      SUM      COUNT     MEAN        STDDEV       STDERR

 ID077     12.90000     1     12.90000      0.00000      0.00000
 ID080    187.90000    13     14.45385      1.14573      0.31777
 ID083    131.40000    10     13.14000      0.59666      0.18868
 ID097    175.80000    13     13.52308      0.83582      0.23181
 ID103    112.40000     8     14.05000      1.39079      0.49172
 ID109    133.80000    10     13.38000      1.18678      0.37529
 ID111    195.50000    13     15.03846      0.76217      0.21139
 ID113     26.50000     2     13.25000      4.45477      3.15000
 ID124     28.70000     2     14.35000      3.74767      2.65000

Enter [+]<n> or -<n> for new page; <n> (1-27) to set top row; ">" or "<"<n>
to shift window; I<n> (1-32) to set window; L to lock columns; N for new
file; F to find; H for help; Q to quit.
>
```

Note standard deviations and standard errors for individuals **ID077** (N = 1, SD = 0), **ID113** (N = 2, SD =4.45477) and **ID124** (N = 2, SD = 3.74767). The next display shows results based on the adjusted standard deviation.

```
calc.out                                  Items: 32      Top record: 18/27
 (1)      (28)      (29)     (30)         (31)          (32)
 EGO      SUM       COUNT    MEAN        STDDEV        STDERR
                                         (ADJ.)        (ADJ.)

 ID077     12.90000     1     12.90000      1.26833      1.26833
 ID080    187.90000    13     14.45385      1.14573      0.31777
 ID083    131.40000    10     13.14000      0.59666      0.18868
 ID095    167.10000    13     12.85385      0.55017      0.15259
 ID103    112.40000     8     14.05000      1.39079      0.49172
 ID109    133.80000    10     13.38000      1.18678      0.37529
 ID111    195.50000    13     15.03846      0.76217      0.21139
 ID113     26.50000     2     13.25000      1.26833      0.89685
 ID124     28.70000     2     14.35000      1.26833      0.89685

Enter [+]<n> or -<n> for new page; <n> (1-27) to set top row; ">" or "<"<n>
to shift window; I<n> (1-33) to set window; L to lock columns; N for new
file; F to find; H for help; Q to quit.
>
```

Here the computed values for individuals with more than 2 measurements remain unchanged from the previous display, but that the pooled inter-individual standard deviation (1.26833) has been substituted for the unadjusted values for **ID077**, **ID113** and **ID124**, and that standard errors have been changed accordingly.

3. Calculating Z-scores

Z-scores transforming numerical values to a standard Normal distribution can be calculated by selecting Choice **3** in the Introductory display, and in the sequence of commands following the confirmation step of the primary functions of Choices **1** and **2**.

4. Setting output precision

Precision of numerical variables sent to the output file **calc.out** (that is, the number of digits to the right of the decimal place) may be set by selecting Choice **4** in the Introductory display, and in the sequence of commands following the confirmation step of the primary functions of Choices **1**, **2** and **3**.

However precision setting is selected, the following display appears:

```
   1 EGO SEQUENTIAL ID       9 PATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   2 SIRE'S SEQ             10 MATERNAL SIBSHIP SIZ   18 Exit Code
   3 DAM'S SEQ              11 FULL SIBSHIP SIZE      19 Mate's Permanent ID
   4 FIRST OFFSPRING SEQ    12 EGO Permanent ID       20 PEDIGREE NUMBER
   5 NEXT PATERNAL SIB SE   13 Birth Date (YYYYMMDD   21 GENERATION NUMBER
   6 NEXT MATERNAL SIB SE   14 Sire's Permanent ID   *22 PSIBSZ / FSIBSZ
   7 NEXT FULL SIB SEQ      15 Dam's Permanent ID
   8 NUMBER OF OFFSPRING    16 Sex (M, F or U)
--------------------------------------------------------------------------------
  CALC        Set precision                  File - MASTER.XMP
--------------------------------------------------------------------------------
Enter the numbers corresponding to numerical values whose precision is
to be set.  Enter [C] to continue.
>
```

After items have been selected for change of precision, the following display appears:

```
PSIBSZ / FSIBSZ.  Enter desired number of digits (0-5) to the right of the
decimal place:
>3


Enter  "R" to Round number
       "T" to Truncate number
>r
```

Here, the ratio PSIBSZ/FSIBSZ has been set to 3 decimal places and rounded. This set of instructions is repeated for each variable selected. When done, the following instruction appears:

```
Reduce field width to length of largest number?  Y/[N]
>y
```

Here, the field width of all selected items is set to the minimum required to accommodate the fraction with the largest number of digits (subject to numerical limits described in Appendix A). Note: Choice of rounding or truncating is given only for Data Type **R**.

Notes:

- At least one operand must be a variable from the list shown on the screen, but calculations are limited to numerical items (that is, Data Types I, R, and D). An error message will result if this requirement is not met.

- Blank fields are treated as missing values and are not included in calculations.

- Operands may be either numbers corresponding to items in the list on the screen, or constants designated by a leading **K** (or **k**). The results of each operation are kept as a new item appended to the list (corresponding to the end of the output record). Operations may be chained, that is, the Item containing the result of a previous operation may be used as an operand in a subsequent calculation.

- The random number generator used is a FORTRAN implementation of a hardware-independent linear congruential algorithm described by R. Sedgewick (p. 37 in Algorithms, Addison-Wesley Publishing Company, Reading, Massachusetts, 1983).

- Calculations in CALC are carried out in single precision, and computed output is limited to the PEDSYS default of five digits to the right of the decimal place. However, the precision of numbers up to 12 places to the right of the decimal may be changed (reduced) with the precision-setting routines.

- Precision of real numbers may not be set greater than that found in input values. For example, a five-digit fraction may not be changed to six digits. This limit does not apply to the change of integers to real numbers (which permits the addition of up to five zeros to the right of the decimal).

- Setting the number of digits to the right of the decimal place to zero results in integer output.

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **calc subset.out**.

Files:

Input: Any Master File or Data File, and an associated Code File. Recursive use of output files is permitted, that is, file **calc.out** may be used as input to program CALC.

Output: **calc.out** contains all records in the input file in their original order; each record contains all items from the input record in their original order, followed by items containing calculations and numerical sequences in the order they were chosen. When precision is changed, items remain in their original position with their field width and Code File entry adjusted as necessary.

calc.cde defines the record structure of **calc.out**.

Program limits:

50 new items (results of calculations, or numerical sequences) may be added to records.

CODE

Program CODE is used to create and / or edit a Code File.

Introductory Display:

```
PEDSYS program CODE - Copyright 1992, 1999 SFBR

Enter -  C  to CREATE a new Code File
        [E] to EDIT an existing Code File
         Q  to QUIT program.
>
```

Create and Edit Modes in CODE are separate functions each having their own style of console entry. Either mode may be selected at the time the program is started; starting in Edit Mode requires that a pre-existing Code File be present.

CREATING a new Code File

When initial choice **C** for CREATE is selected, the following instruction appears:

```
Enter - [D] to create a Code File for a Data File
         H  to create a Code File for a Master File (HUMAN population)
         N  to create a Code File for a Master File (NONHUMAN population)
>
```

Option D -- Create a Code File for a Data File

Entry of **D** at this point begins the process of creating a Code File for a Data File.

```
Enter TITLE of file with which Code File will be associated
>Tissue Inventory
```

This entry (47 characters maximum) will be stored as the Code File Label Record (see Code Files, Appendix A). The entry may be a file name, but since it is used as a title in screen and print displays, it is usually preferable to enter a more complete or mnemonic description. Consequently, we find it advantageous to use a title such as "Tissue Inventory" here, rather than simply "TISSUES.XMP". CODE then begins a repetitive cycle of requests for information about each item to be included in the Data File records.

```
>Animal ID           < Enter label for Item 1, or RETURN for next step.
>ID    < Enter mnemonic 1 for this item, or "]" to re-enter
>      < Enter mnemonic 2 for this item, or "]" to re-enter
>5 < Enter width (columns) of this item, or "]" to re-enter
>c< Enter data type of this item, or "]" to re-enter
        I for integer value
        R for real value
        C for character value
```

These requests are (in order):

a. The item name or description (21 characters maximum).

b. Up to four mnemonic words describing the item (maximum 6 characters each). These should be chosen carefully, as they are used as column headings for the output of program LIST, and may serve as the only convenient means of identifying data presented in long listings. It is important that no two items have exactly the same mnemonics. Completion of entry of the fourth word results in a jump to the next class of requests (that is, to c. below), but all four mnemonics need not be entered; a carriage return alone also results in a jump to the next request.

c. Field length of the item (1 - 99). An error message will result if an appropriate numerical value is not entered here.

d. Data type of the item (I, R, C and D). An error message will result if an appropriate letter is not entered here. Type D (date) is permitted only if field width is 3, 4, 7, or 8 (corresponding to date formats YYY, YYYY, YYYMMDD, or YYYYMMDD).

Once data type has been entered, the Descriptor Record is displayed at the top of the screen, and an opportunity is provided to make changes in it by shifting to Edit Mode:

```
   1.    5 Animal ID            ID                          C


Edit this item?  Y/[N]
>
```

If the Edit option is not selected, the choice of entering the next Descriptor Record, or of leaving Create Mode is given:

```
Item 1:
    Label                   Mnemonics                       Width     Type
    Animal ID               ID                               5        C
================================================================================

>                   < Enter label for Item 2, or RETURN for next step.
```

Note that the previouly entered Descriptor Record is shown just above the horizontal line in this display.

When entries for all items have been made, Descriptor Records are displayed in the upper portion of the screen, and a number of options are given to edit the Code File as shown below:

```
   1.    5 Animal ID             ID                          C
   2.    8 Date Necropsied       DATE   NECROP               D
   3.    2 Freezer Rack          RACK                        C
   4.    2 Tissue Box            BOX                         C
   5.    3 Tissue Type           TYPE                        C
   6.    2 Quantity              QUANT                       I
   7.    3 Health Status         HEALTH STATUS               C
   8.    4 Diet                  DIET                        C
--------------------------------------------------------------------------------
  Tissue Inventory                    File - code.tmp
--------------------------------------------------------------------------------

Enter - [E] to EDIT an item
         A  to ADD an item
         R  to REPLACE an item
         D  to DELETE an item
         T  to CHANGE file TITLE
         X  (or S) to SAVE Code File and EXIT program.
         Q  to QUIT program without saving Code File.
>e4
```

Command E -- Edit an Item

Entry of **e**<n>, where <n> is a number corresponding to one of the items to be modified produces the display shown below (entry of **e** alone results in a request for an item number before the display appears):

```
Re-enter item to right of pointer (>), or enter one or more spaces to blank
out item, press RETURN to advance cursor, or enter "]" to go to next step.

>Tissue Box           <- Label 4
>BOX   <- Mnemonic 1
>      <- Mnemonic 2
>      <- Mnemonic 3
>      <- Mnemonic 4
>2 <- Width
>C<- Data type
```

This example shows a previously entered label (Tissue Box) and single mnemonic (BOX) for a character item with a field width of 2. Entries may be replaced when the cursor is in the initial position to the right of the pointer. Entry of RETURN alone advances through the entries without changing them. Entry of a space as the initial character of any field will erase the entire field. Entry of **]** as the initial character of any field results in a return to the initial Edit options screen.

Command A -- Add an Item

This command produces a request for the number of an item, *after* which a new item will be inserted.

Command R -- Replace an Item

Command **r**<n>, results in an entry sequence that permits replacement of item <n>; entry of **r** alone results in a request for an item number before entry begins.

Command D -- Delete an Item

Entry of **d**<n>, results in deletion of item <n>; entry of **d**<n-m> results in deletion of items <n> through <m>; entry of **d** alone results in a request for a number (or range of numbers) corresponding to items to be deleted.

Command T -- Change Data File title

This command makes it possible to replace the file title, that is, the contents of the Label Record (see Appendix A).

Command S (or **X**) **-- Save Code File and exit program**

When a Code File has been newly created, entry of **x** (or **s**) produces the following display, which gives two alternatives for saving output:

```
Enter a name for this Code File, or RETURN alone to save output as code.tab.
>
```

Command Q -- Quit program without saving Code File

Entry of this command makes it possible to quit program CODE without saving the Code File if it has been generated. Confirmation is required before final exit.

Option H -- Create a Code File for a Master File (human population)

If a Master File is to be created from keyboard entry using program ALTMASTR, entry of **H** automatically creates and formats the first 11 Master File items (SEQ, FSEQ, MSEQ, KID1, PSIB, MSIB, FSIB, NKID, PSIBSZ, MSIBSZ and FSIBSZ). The next five Master File items (EGO, BIRTH, FA, MO and SEX) appear on the screen with most entries filled in. Exceptions which require keyboard entry are width and data type (limited to C or I) of the EGO PID field, width of the BIRTH Date field, and the data type of the SEX field, which may be either C (for M, F, U) or I (for 1, 2, or 3). Widths of FA and MO PIDs are automatically set to the width chosen for EGO PID. Subsequently (starting with Item 17) the cycle of requests for information described below for Data File items begins, making it possible to add optional items to the Master File record. This constrained data entry is required to ensure that mnemonics, field widths and data types are correctly configured for processing with ALTMASTR.

Option N -- Create a Code File for a Master File (nonhuman population)

Entry of **N** likewise creates and formats Master File items, but substitutes mnemonics ID, SIRE, and DAM for EGO, FA and MO, respectively.

Starting CODE in EDIT Mode

When initial choice **E** for EDIT is selected, the screen shown below is displayed, from which the Code File to be edited may be chosen:

```
 1.  Unlinked Code File
 2.  Example

Enter number of the file or directory to use.
>
```

Choice of **1** from the list above results in a display of Code Files associated with unlinked Data Files. Included in this list are **code.tab** (if it exists), any file with a name in the form **<filename>.cde**, or the form **CODE.<ext>**, where <filename> is the name of the data file and <ext> is the file name extension. Both upper and lower case characters will be shown.

Once the Code File has been selected, editing begins. The edit display and its commands process are the same as that described above, except that the options provided for naming the output file are slightly different from those given when a new Code File has been created. When a previously created Code File has been modified, entry of **s** (or **x**) directs the program to save the output and exit. Naming the output file depends on the origin of the input file:

```
Enter a name for this Code File, RETURN alone to save output as code.tab, or
"R" to replace examp.cde.
>
```

Notes:

- ***VERY IMPORTANT NOTE:*** Program CODE alters **ONLY** the Code File; *modifying the Code File alone does not alter the structure of records in its accompanying Data File*. Use program REFORMAT to change the structure of records in a Data File. REFORMAT alters the data records and automatically produces a Code File (named **reformat.cde**) that reflects the changes. **A common mistake** made by first users of PEDSYS is to try to change the format of records in a Master File or Data File simply by changing the definition of these records in the associated Code File. *This does not work.* Changing a Code File by itself *does not* change the contents of records in the file with which it is associated.

- Changing Item Labels and Mnemonics affects the description of items as read from screen and output display, which may be crucial for understanding and interpreting the *meaning* of stored data, but such changes do not alter the physical contents of the records, nor do they affect the operation of the PEDSYS programs that use them.

  Likewise, simply adding or deleting items, or changing the specification of item field widths in a Code File does not change the actual structure or content of records in the associated file. Moreover,

PEDSYS programs require a Code File to determine the format of records in any input file. Thus, if the Code File reflects changes in structure that have not been made to the file it is supposed to describe, it is unlikely that programs will accept input without error and termination.

- ***Also important:*** The order of items entered, field width, data type, and the exact spelling of the mnemonic words are invariant in some Code Files. See Master Files, Data Files, and **CODE.STD** in Appendix A, and documentation for Program INDEX for details.

- A useful way of checking whether or not a Code File fits the Data File is to run program BROWSE or SHOW, choosing the Unlinked File category and then specifying the Data File name and **code.tab** as the associated Code File. If a few pages of the screen display are correctly formatted, it is likely that the Code File was properly constructed. If, however, records are out of alignment, immediately run program ITEMIZE with the same input files. ITEMIZE will readily expose errors in field widths, etc. and makes it easy to correct them.

Files:

Input:
1. Console, and optionally a previously constructed Code File.
2. Any Code File when starting program in Edit Mode.

Output: The new Code file may be saved either as

1. **code.tab**,
2. **<filename>**, where <filename> is the either a newly chosen name, or the name of the Code File selected as input for modification.

Program limits:

Up to 100 Code File Descriptor Records (each corresponding to an item in records of the associated Master File or Data File).

COMBINE

COMBINE produces single new records, each containing one or more values extracted in sequence from a Data File containing *sets* of records defined by a common identifier. The most obvious use of the program is to extract items from record sets in a Multiple-entry file (where sets are defined by Ego PID), and place these items in some order on a single output record also identified by Ego PID. The program is also used with Single-entry files where record sets are defined by some identifier occurring on more than one record (such as a parental PID).

Introductory Display:

```
PEDSYS program COMBINE - Copyright 1992, 1999 SFBR

COMBINE produces single new records, each containing one or more values ex-
tracted in sequence from a Data File containing SETS of records defined by
a common identifier (for example, the same Ego ID or same parents, etc.).

1.  Unlinked file
2.  EXAMPLE
3.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>2
```

In this initial example, we select the Hemoglobin Levels File (a Multiple-entry Data File), and proceed through the steps that will combine hemoglobin measures for each individual on one record.

Step 1 - Selecting Key Items by which Input Records Are Sorted into Sets

Once a file has been selected, the screen shown below is displayed, from which one or more key items defining record sets are chosen:

```
 * 1 Ego                    3 Hemoglobin Level
   2 Sample Date            4 INTERNAL POINTER
--------------------------------------------------------------------------------
  COMBINE     Select Identifiers              File - HEMOGLOB.XMP
--------------------------------------------------------------------------------
Enter one or more numbers corresponding to identifiers by which input record
sets will be defined prior to extracting data items (typically an Ego ID).
Enter [C] to continue.
>
```

Here, Ego PID has been selected as the identifier which determines sets of input records from which data items will be extracted and combined onto a single output record.

- **Option - Sorting Input Records within Sets**

  Next, an opportunity is provided to sort records within sets. This will determine the order that items will eventually appear on the combined record:

```
 + 1 Ego                    3 Hemoglobin Level
   2 Sample Date            4 INTERNAL POINTER
--------------------------------------------------------------------------
  COMBINE       Sort within groups                 File - MASTER.XMP
--------------------------------------------------------------------------
Record sets will be defined as marked above.  Optionally, enter one or more
numbers corresponding to items by which records will be sorted within sets
(precede with "D" for descending order).  Enter [C] to continue to next
step.
>2
```

In this example, Sample Date (Item **2**) has been entered for sorting in ascending order. Note that the key item previously selected in Step 1 to define record sets (Item **1**, Ego PID) is preselected in this display. The option of sorting within sets may be skipped.

Step 2 - Selecting Items to Include on Combined Output Records

Next, items that are to be included on the combined (output) record are selected:

```
 * 1 Ego                    3 Hemoglobin Level
   2 Sample Date            4 INTERNAL POINTER
--------------------------------------------------------------------------
COMBINE       Select items to include            File - MASTER.XMP
--------------------------------------------------------------------------
Enter numbers corresponding to items to be included in the combined record.
Enter "A"  to include all items, "C" to continue to next step.
>2-3
```

Here, Sample Date, and Hemoglobin Level (Items **2** and **3**) have been chosen. Note also that the item selected to define record sets (Item **1**, Ego PID) has been preselected and will be automatically become the first item of the combined record.

- **Option - Grouping Items by type within Output Records**

  Normally, data on output records consist of sets of item sequences whose order is determined by the order of selection from input records. The result of this default is that items of any one type are interspersed with items of different types over the length of the record. This order may be changed by the following option so that items of the same type are grouped together on the combined record:

```
Enter "G" to order the output record so that the same item types are grouped
together.  Enter RETURN to leave items on output records in the same order
as they were picked from input records.
>
```

- **Option - Choosing Record Sets**

  Next, an opportunity is provided to specify a source of identifiers of input record sets for which combined output records are to be constructed:

```
Enter RETURN to combine items from all records in [HEMOGLOB.XMP].

  1.  Keyboard
  2.  Unlinked File
  3.  Example
  4.    LOCUS Sub-directory

Enter number corresponding to alternate source of record identifiers.
>1
```

Entry of **RETURN** results in extraction of items from all sets identified in the input Data File, **1** allows extraction of items from selected sets whose identifiers are entered from the keyboard, **2 - 4** allows extraction of items from selected sets whose identifiers are read from a separate file.

- **Example 1 -- Combining Items from Input Records in a Multiple-Entry File**

In our first example, we show the results of selecting a single record set from file **HEMOGLOB.XMP** by entering a PID from the keyboard:

```
Enter Ego (RETURN alone starts execution)
>ID015

Enter Ego (RETURN alone starts execution)
>
```

To see the effects of COMBINE, look first at the display from SHOWDATA below, which shows the set of five records belonging to individual ID015:

```
HEMOGLOB.XMP  ID:ID015  Items:4  Recs:5 (LIFO)    Page:1/1    FilePos:1/238

 ( 1)    ( 2)     ( 3)  ( 4)
 EGO     DATE     hgb   PNTR

ID015 1988/05/27  14.2     4
ID015 1989/03/24  13.5     3
ID015 1990/01/24  12.3     2
ID015 1991/01/08  13.2     1
ID015 1993/05/06  12.5     0




Enter [+]<n> or -<n> for new page; PID, S<n> (1-32) or =<n> (1-238) to set
top row; ">" or "<"<n> to shift window; I<n> (1-4) to set window; L to lock
columns; O to change order; * to switch files; N for new file; H for help;
Q to quit.
>
```

The results of combining Items **2** and **3** for individual ID015 into a single record can be seen in the display from BROWSE below, which shows the record found in file **combine.out** (default ungrouped format):

```
 combine.out                          Items: 11       Top record: 1/1
 (1)     (2)      (3)     (4)      (5)     (6)      (7)     (8)      (9)     (10)     (11)
 EGO     DATE     hgb     DATE     hgb     DATE     hgb     DATE     hgb     DATE     hgb
         Rec1     Rec1    Rec2     Rec2    Rec3     Rec3    Rec4     Rec4    Rec5     Rec5

 ID015 1988/05/27  14.2 1989/03/24  13.5 1990/01/24  12.3 1991/01/08  13.2 1993/05/06  12.5

Enter [+]<n> or -<n> for new page; <n> (1-1) to set top row; ">" or "<"<n> to
shift window; I<n> (1-11) to set window; L to lock columns; N for new file;
F to find; H for help; Q to quit.
>
```

Note that entries corresponding to Items 2 and 3 from the original record appear in five sets, with items in each set repeated from left to right in the order originally selected in Step 1. The source of each item on the new record (designated by Rec1, Rec2, ...) is placed in the first unused (or the last) mnemonic word of the output Code File **combine.cde**. Because the BROWSE display is limited to the first two mnemonic words, this information sometimes cannot be seen.

If the item grouping option is selected, items are grouped by type, as shown below:

```
 combine.out                          Items: 11       Top record: 1/1
 (1)     (2)        (3)        (4)        (5)        (6)      (7)   (8)   (9)   (10)  (11)
 EGO     DATE       DATE       DATE       DATE       DATE     hgb   hgb   hgb   hgb   hgb
         Rec1       Rec2       Rec3       Rec4       Rec5     Rec1  Rec2  Rec3  Rec4  Rec5

 ID015 1988/05/27 1989/03/24 1990/01/24 1991/01/08 1993/05/06  14.2  13.5  12.3  13.2  12.5

Enter [+]<n> or -<n> for new page; <n> (1-1) to set top row; ">" or "<"<n> to
shift window; I<n> (1-11) to set window; L to lock columns; N for new file;
F to find; H for help; Q to quit.
>
```

Note now that all dates are grouped together, as are all hemoglobin levels, and that the original order of items is preserved within group.

- **Example 2 -- Combining Items from Input Records in a Single-Entry File**

In the second example, we show the results of combining items from input record sets derived from a Single-entry file, the Example Master File. In this case, key items defining record sets must appear on more than one record. In Step 1 shown in the display below, Sire and Dam Permanent ID have been chosen jointly to define sets, so that items extracted from full sibships will be combined onto a single output record:

```
   1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING   *15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ   16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE   12 EGO Permanent ID       19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE   13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ     *14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  COMBINE     Select Identifiers              File - MASTER.XMP
--------------------------------------------------------------------------------
Enter one or more numbers corresponding to identifiers by which input record
sets will be defined prior to extracting data items (typically an Ego ID).
Enter [C] to continue.
>
```

Next, the option to sort records within sets is selected:

```
   1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING   +15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ   16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE   12 EGO Permanent ID       19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE  +13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ     +14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  COMBINE     Sort within groups              File - MASTER.XMP
--------------------------------------------------------------------------------
Record sets will be defined as marked above.  Optionally, enter one or more
numbers corresponding to items by which records will be sorted within sets
(precede with "D" for descending order).  Enter "C" to continue to next step.
>
```

Here, input records will be sorted by Birth Date (Item 13) so that sibship data will be ordered likewise on the output record. Step 2 (Selecting Items to Include on Combined Output Records) is next:

```
   1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING   *15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ   16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE  *12 EGO Permanent ID       19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE  *13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ     *14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  COMBINE     Pick items to be grouped        File - MASTER.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to items to be included in the combined record.
Enter "A"  to include all items, "C" to continue to next step.
>
```

Here, we have chosen Ego's Permanent ID and Birth Date (Items **12** and **13**) for inclusion on output records. The results of combining these items can be seen in the display from BROWSE below, which shows the records found in file **combine.out**. Items have not been grouped so that PIDs alternate with birth dates for as many offspring as there are in each full sibship:

```
combine.out                          Items: 8       Top record: 1/14
 (1)   (2)   (3)     (4)      (5)     (6)      (7)     (8)
 SIRE  DAM   ID      BIRTH    ID      BIRTH    ID      BIRTH
             Rec1    Rec1     Rec2    Rec2     Rec3    Rec3

 ID015 ID019 ID067 1996/07/27
 ID038 ID042 ID063 1980/11/28
 ID045 ID034 ID071 1993/03/18 ID049 1996/09/08
 ID054 ID068 ID069 1996/10/13
 ID055 ID065 ID034 1987/06/23
 ID056 ID066 ID047 1982/00/00
 ID061 ID042 ID065 1981/07/05
 ID062 ID047 ID015 1987/00/00 ID068 1988/08/31
 ID063 ID065 ID054 1986/12/15 ID070 1989/10/12 ID077 1994/09/23
 ID080 ID083 ID103 1987/06/26
 ID095 ID097 ID109 1985/08/14
 ID109 ID103 ID113 1992/09/27
 ID109 ID111 ID124 1991/09/08
 ID124 ID113 ID130 1997/11/30

Enter [+]<n> or -<n> for new page; <n> (1-14) to set top row; ">" or "<"<n>
to shift window; I<n> (1-8) to set window; L to lock columns; N for new
file; F to find; H for help; Q to quit.
>
```

Notes:

- A blank field will be included in the output record if a blank is found in the input record. Output records will be padded with trailing blanks as needed.

- Order of access depends on the position of records in the input file. The order in which records were entered in linked Multiple-entry Files (which may or may not be chronological) can be recovered by sorting on the **INTERNAL POINTER** field.

- The compound sort in program COMBINE provides control over which record will be first in the list for comparison. The major sort is for the item (or items) selected for grouping records together, but other items (such as dates) may be added as minor sorts so as to order records within groupings.

  WARNING: Care should be taken not to combine so many values that the output record becomes excessively long and unwieldy. PEDSYS limits records to 1024 characters in length. Combined records that would exceed this limit are truncated, and a message is sent to the error log, **combine.err**.

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **combine subset.out**.

Files:

Input: Any Data File containing more that one record with the same identifier and an associated Code File, and (optionally) the console keyboard or an input file containing IDs of individuals whose records are to be combined.

Output: **combine.out** contains one record for each individual for which Multiple-entry Data records exist.

combine.cde defines the record structure of **combine.out**.

combine.err contains information about records that cannot be successfully combined, that is, by identifiers entered from keyboard entry or other alternate source, and IDs of output records longer than 1024 characters.

COUNTPED

Program COUNTPED (1) identifies genealogical links in a file and assigns numbers to each individual corresponding to the pedigree to which he or she belongs, (2) assigns the size of the pedigree and the generational position of that individual in the pedigree (founders are assigned to Generation 0, with numbers increasing through more recent generations), and (3) assigns to each individual a number designating the founder of his or her paternal or maternal ancestral lineage. The file containing data to be analyzed may be a Master File, or any pedigree file consisting of records each of which contains a unique identifier for Ego, Father and Mother.

Introductory Display:

```
PEDSYS program COUNTPED - Copyright 1992, 1999 SFBR

COUNTPED (1) identifies genealogical links and assigns numbers to individuals
corresponding to the pedigree to which they belong, (2) assigns pedigree size
and generational position of each individual in the pedigree, and (3) assigns
a number designating the founder of each individual's paternal or maternal
ancestral lineage.

 1.  Unlinked file
 2.  EXAMPLE
 3.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>2
```

After a file has been selected, COUNTPED searches its Code File for the standard mnemonics as specified in **CODE.STD** (see File Types in Appendix A). If standard mnemonics for both Sequential IDs and Permanent IDs are missing, the following display appears, from which IDs of Ego, father and mother must be selected:

```
1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZE  16 Sex (M, F or U)
3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZE  17 Exit Date (YYMMDD)
4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
5 NEXT PATERNAL SIB SEQ  12 EGO Permanent ID       19 Pedigree ID Number
6 NEXT MATERNAL SIB SEQ  13 Birth Date (YYMMDD)    20 Generation Number
7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 Mate's Permanent ID

--------------------------------------------------------------------------------
COUNTPED      Identify EGO and parents            File - MASTER.XMP
--------------------------------------------------------------------------------

Enter numbers above corresponding to ID of EGO, FATHER and MOTHER.
>
```

When these individuals have been identified, or if standard mnemonics are found, the following display appears, from which items are selected for inclusion in the output file. At least one item must be selected:

```
1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZE  16 Sex (M, F or U)
3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZE  17 Exit Date (YYMMDD)
4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
5 NEXT PATERNAL SIB SEQ  12 EGO Permanent ID       19 Pedigree ID Number
6 NEXT MATERNAL SIB SEQ  13 Birth Date (YYMMDD)    20 Generation Number
7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 Mate's Permanent ID

--------------------------------------------------------------------------------
COUNTPED      Items to be included               File - MASTER.XMP
--------------------------------------------------------------------------------

Enter numbers corresponding to the items to be included in the output file.
Pedigree Number, Generation Number and Pedigree Size will be added to the
end of each record.  Enter "A" to select all, [C] to continue.
>
```

When this selection has been made, the following display appears:

```
Assign individuals that have neither parents nor offspring

 1. All to the same pedigree (Pedigree 0)
 2. Each to a separate pedigree.

Enter number corresponding to choice
>

If EGO's record contains one or more parental IDs for which individual
records are not present in the pedigree file

[1] Establish pedigree links by creating new parental records
 2. Use EGO's parental IDs, but do not create new parental records
 3. Remove missing parental IDs from EGO's record

Enter number corresponding to choice
>

 1. Designate paternal and maternal lineage markers
[2] Skip lineage markers
>
```

When these decisions have been made, COUNTPED begins execution.

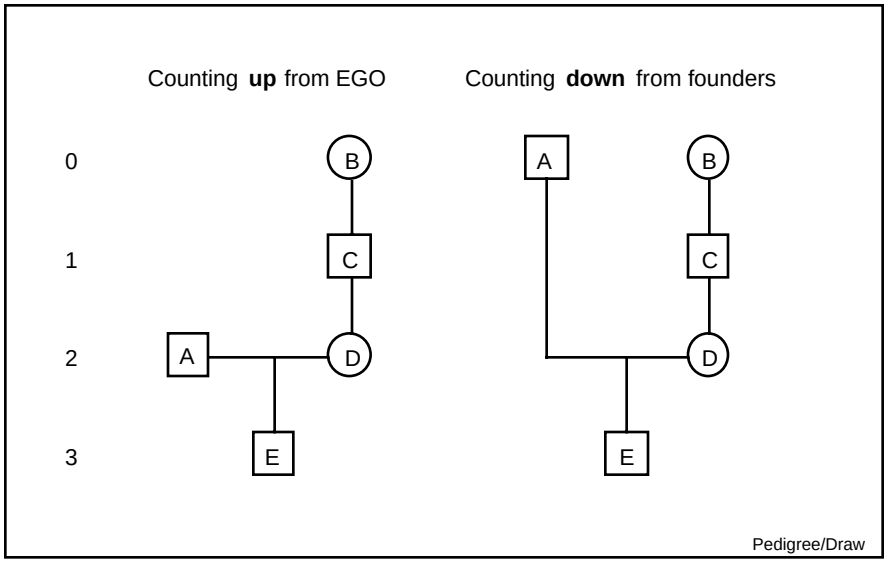
Notes on assignment to pedigrees:

- COUNTPED starts pedigree assignment by locating the first founder (an individual with no parents) in the file, and assigning this individual to Generation 0. Children of this founder are assigned to Generation 1, children's children to Generation 2, etc. until all descendants have been assigned. The next founder is located, and the process is repeated in the same way until all descendants of all founders have been assigned. Different generation numbers may be computed for the same individual as the result of descent traced from different founders. In this case, the larger number is assigned.

- Occasionally, a data entry error may connect families in such a way that a PID appears more than once in the same pedigree. When COUNTPED detects this error, it terminates execution with a FATAL ERROR message, and saves PIDs of all individuals for which an incorrectly entered PID (including parental PIDs) might account for the the error in file **countped.err**. In a large family, it may be easier to locate the incorrect data entry by running the input data through program INDEX, which, in addition to a check for multiple PIDs, identifies individuals whose birth dates are inconsistent with those of their parents.

- COUNTPED finds all independent pedigrees by tracing connections between individuals using a FORTRAN adaptation of algorithms taken from program PEDSPLIT, part of the LINKAGE package (version 3.5) by M. Lathrop and J-M. Lalouel.

- COUNTPED's convention of counting generations down from the top of the pedigree can lead to apparent anomalies because parents and offspring may not be assigned to adjacent generations. For example, in the pedigree below individual **A** is the father of Ego **E**. If generations are assigned by counting up from Ego, **A** would be assigned to Generation 2 (left-hand diagram). Results of the COUNTPED convention are illustrated in the right-hand diagram, where it can be seen that generation numbers of Ego and **A** are separated by two.



- Frequently it is more convenient to group all unrelated individuals in the same pedigree; such individuals are of course all assigned to Generation 0.

- If the input file contains a subset of a larger pedigree, records for individuals appearing as parents on Ego's record may be missing. In this case, COUNTPED can either use or ignore information from Ego's parental IDs. Parental information may be used with or without creation of new parental records; if it is ignored, both Sequential IDs and Permanent IDs are removed from Ego's record

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **countped subset.out**.

- If the input Code File file contains mnemonics PEDNO, GEN and PEDSIZ, COUNTPED preserves these items, renames them OLD PEDNO, OLD GEN and OLD PEDSIZ, respectively, and appends newly computed items at the end of each record. This is in contrast to program INDEX, where the original items are replaced.

Note on lineage marking:

- COUNTPED starts lineage marking by assigning an arbitrary integer code to each founder. Then, beginning with individuals of either sex in the most recent generation, maternal and paternal lineages are determined by tracing links upward through ancestors of the same sex until a founder of the same sex is reached. All descendents in this line are assigned the integer code of the female founder as a maternal lineage marker, and the code of the male founder as a paternal lineage marker. The file may then be sorted on markers to isolate and identify paths of maternal or paternal genetic transmission.

Files:

Input: 1. Any Master File or pedigree file (having records containing a unique identifier for an individual and his or her parents), and an associated Code File.

2. The Standard Mnemonics File (**CODE.STD**).

Output: **countped.out** contains all records in the input file in their original order; records contain items in the input records in the order selected, followed by items

a. `PEDNO` Identification number of the pedigree of which Ego is a member.
b. `GEN` Ego's generational position in the pedigree.
c. `PEDSIZ` The size of the pedigree of which Ego is a member.

and if lineage markers have been selected

d. `PATERNL LINES` Paternal lineage markers
e. `MATERNL LINES` Maternal lineage markers

countped.cde defines the record structure of **countped.out**.

countped.err contains PIDs of individuals for which an incorrectly entered PID might account for families in which Ego appears more than once.

DATALINK

Program DATALINK is a utility for linking a Single-entry or Multiple-entry Data File to a Master File, and for establishing the internal pointers of a Multiple-entry Data File.

Introductory Display:

```
PEDSYS program DATALINK - Copyright 1992, 1999 SFBR

DATALINK is used to link a Single-entry or Multiple-entry Data File to a Master
File, and to establish the internal pointers of a Multiple-entry Data File.

1.  EXAMPLE MASTER FILE

Enter number corresponding to a Master File to which Data Files are to be linked.
>
```

Once a Master File has been selected, a Standard File List appears from which a Data File to be linked to the Master File is selected.

Error Messages:

A number of conditions force DATALINK to terminate without establishing links between files. When any of these occurs, a message is written to file **datalink.err**. These messages are as follows:

```
ERROR: No Data Files to link to MASTER.<EXT>
```

This error is most likely to occur when no Data File information has been entered in the Directory File DBFILES. It can also occur if the filename extension .<EXT> on a Data File to be linked has been left blank, or does not match the extension found in DBFILES. To correct the error, make sure that all necessary information about the Data File has been entered in DBFILES (see chapter describing program DBFILER), and that this information is consistent with the name and extension of the Data File.

```
ERROR: More than 50 Data Files.
```

This error occurs when the PEDSYS limit of 50 Linked Data Files per Master File is exceeded. To correct this error, one or more Data Files will have to be removed from DBFILES.

```
ERROR: Master File and Data Files must have valid extensions.
```

This error occurs when attempting to create a linked database without using a filename extension .<EXT>, that is, the extension is missing from DBFILES and from Master File and Data Files. To correct the error, make sure that all necessary information about the Master File and Data File has been entered in DBFILES (see chapter describing program DBFILER), and that names and extensions of all files are consistent.

```
ERROR: DPOINT.<EXT> does not exist.
```

DATALINK creates a Data Pointer File if it does not already exist when only one Data File is listed in DBFILES. This error message occurs when DPOINT.<EXT> does not exist, but more than one Data File is listed in the Directory File. To correct this error, begin the file linking process from the start: remove all Linked Data File Descriptor Records from DBFILES using program DBFILER. Then

(also with DBFILER) add a single Data File to DBFILES, and run program DATALINK. Repeat the DBFILER-DATALINK sequence until all files have been linked.

```
ERROR: Number of Data Files listed in DBFILES does not correspond to the number of
pointer fields in DPOINT.<EXT>
```

The safest way to recover from this error is to begin the file linking process from the start: delete file DPOINT.<EXT> and remove all Linked Data File Descriptor Records from DBFILES using program DBFILER. Then (also with DBFILER) add a single Data File to DBFILES, and run program DATALINK. Repeat the DBFILER-DATALINK sequence until all files have been linked.

```
ERROR: Permanent Ids in Master File and Data File not the same field width.
```

To correct this error, change PID field width in the Data File using program REFORMAT.

Notes:

- Only one Data File may be linked at a time; DATALINK must be run each time a Data File is added to the Directory File **DBFILES** and linked to a Master File.

- <FILENAME>.<EXT> (the Data File to be linked) must be named in DBFILES. That is, there must be a Data File Descriptor Record for **<FILENAME>.<EXT>** that contains an appropriate file extension (see Note **k** in the section on The Directory File, Appendix A), a file name (Note **l**),a label (Note **m**), a pointer position number (Note **n**), a multiple-entry indicator (Note **o**), a Log File indicator (Note **p**), and passwords, as needed (Note **q**). The Data File Count Record must also reflect the presence of the file (Note **i**).

- **DPOINT.<EXT>** (The Data Pointer File, Appendix A) will be created by DATALINK if it does not exist.

- A new pointer will be appended to each record in **DPOINT.<EXT>** at the time **<FILENAME>.<EXT>** is first linked to a Master File. Subsequently, the pointer position number specified in the relevant Data File Descriptor Record of DBFILES (see Note **n** in the section on The Directory File, Appendix A) will determine the pointers to be updated.

- DATALINK creates an updated version of a preexisting version of **DPOINT.<EXT>** (if it exists), but saves a copy of the old file as **DPOINT.OLD**. *Warning. Prior versions of* ***DPOINT.OLD*** *will be deleted and over-written each time DATALINK is run unless they have been renamed.*

- DATALINK creates an updated version of **<FILENAME>.<EXT>** if it is a Multiple-entry Data File, but saves a copy of the old file as **<FILENAME>.OLD**. *Warning. Prior versions of* <**FILENAME>.OLD** *will be deleted and over-written each time DATALINK is run unless they have been renamed.*

- Ordering Multiple-entry Files: It is often necessary for multiple-entry data to be in a particular order, for example, by date of measurement. To ensure that entries follow a specific order, Multiple-entry Files may be sorted on the critical variable (such as a date) with program SORT before being run with DATALINK.

- The last item on Linked Multiple-entry Data File records is the internal pointer (standard mnemonic PNTR). DATALINK automatically adds this field to Data File records if it is missing (as it would be prior to the first time the file is linked), and adds a descriptor record to the Data Code File to reflect the change.

Files:

Input: A Master File and a Data File **<FILENAME>.<EXT>** named in DBFILES, and their associated Code Files.

The Master Pointer File **MPOINT.<EXT>**, and Data Pointer File **DPOINT.<EXT>**, if the latter exists.

Output: An updated version of **DPOINT.<EXT>** is produced when the program is run, and the original version of the Data Pointer File is saved as a backup under the name **DPOINT.OLD**. **DPOINT.OLD** is not produced if **DPOINT.<EXT>** is newly created.

An updated version of **<FILENAME>.<EXT>** is produced if it is a Multiple-entry Data File. When this is done, the original version of the Data File is saved as a backup under the name **<FILENAME>.OLD**.

datalink.err contains information about Data Records for which no corresponding record can be found in the Master File.

DATES

Program DATES changes the format of calendar dates. PEDSYS Data Type **D** refers only to dates that are in numerical Gregorian form in order year-month-day, without separators (slash / separators are added temporarily in some PEDSYS displays, but are not stored as part of the date). Simple conversion is typically used when an alternate calendar date representation is required for use with a foreign database. Adding a constant time or age interval produces a new date at which the interval will have elapsed, which is useful for setting schedules, etc.

Introductory Display:

```
PEDSYS program DATES - Copyright 1992, 1999 SFBR

DATES changes the format of calendar dates. New dates replace those selected
for conversion, except where constants are added to or subtracted from the
original, in which case they are appended as items at the end of the record.

1.  Unlinked file
2.  EXAMPLE
3.    LOCUS Sub-directory

Enter number of the file or directory, filename or "Q" to quit.
>
```

When the PEDSYS file containing dates to be re-formatted has been selected, the following display appears:

```
1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZE  16 Sex (M, F or U)
3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZE  17 Exit Date
4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
5 NEXT PATERNAL SIB SEQ  12 EGO Permanent ID       19 Pedigree ID Number
6 NEXT MATERNAL SIB SEQ  13 Birth Date             20 Generation Number
7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 Mate's Permanent ID
--------------------------------------------------------------------------------
DATES                           File - MASTER.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to dates to be converted. Enter "<i>+/-<n>Y/M/D"
to add or subtract <n> years, months or days from date <i> before conver-
sion. Enter "*<i1>+/-<i2>Y/M/D" to add or subtract contents of item <i2>
from date <i1>. Enter [C] to continue, "H" for help.
>
```

Commands:

Simple conversion of date format is initiated by entering a number corresponding to an item containing a date. In order to preserve the freedom to input dates having many different formats, program DATES makes no assumptions about the Data Type of the item to be converted, which means that it is quite possible by mistake to select an item that does not contain a date. This error may, or may not be detected when the program checks to make sure that the input field width (read from the Code File) is consistent with the input date format selected (as shown below). In general, however, *it is left to the user to know which fields in the input file contain calendar dates, and what their format is.*

<i>+/-<n>Y/M/D

This sequence adds (or subtracts) constant **<n>**, designated as year, month or day by a following **Y**, **M**, or **D**, respectively, from item **<i>** containing a date. Year may be expressed as integer plus a three-digit fraction, month as an integer plus a two-digit fraction, but day may be expressed as an integer only. Combinations of year, month and day are not permitted. Example: **13+6m** yields the date at which each individual reaches six months of age.

***<i1>+/-<i2>Y/M/D**

This sequence adds (or subtracts) the contents of numerical item **<i2>**, designated as year, month or day by a following **Y**, **M**, or **D**, respectively, from item **<i1>** containing a date. This feature is particularly useful when computing the date that an event occurs when only Ego's age at the time of the event is given. Example: suppose Ego's *age* in years at the birth of his or her first offspring is given in Item 22 of each Master File record. The *date* of first birth could be determined by entering ***13+22m**.

C Entry of **C** produces the following display:

```
Operation 1
Select format of INPUT date representation for EXIT        (Width = 8)

    Unit    Format           Interpretation
    ------------------------------------------------------------------
    Year  : YY*, YYY, YYYY : Number of integers (3 or 4 only)
    Month : MM, MON, MONTH : Integer, abbreviated and full alphabetic
    Day   : DD, F, J       : Integer, fraction of year, days since 01 JAN

    Julian: J<base>        : Days since base date (1, 2, and 3 only)
                           :        1 = 1900/01/01
                           :        2 = 01 Jan 4713 BC (true Julian base)
                           :        3 = arbitrary base date
    ------------------------------------------------------------------
  (Examples: YYYYMMDD; yyy/mm/dd; yyy.f; MONTH dd, yyy; ddmonyyy; J1, etc.)

  * Two-digit year valid for input only

Enter date FORMAT (include separators if present), RETURN to skip this
operation, or "H" for help.
>
```

A similar display (but not permitting two-digit year format) appears after **input** date format has been entered, from which **output** date format is selected. When input and output formats have been chosen, a confirmation step occurs:

```
Format of date representation for BIRTH

    Input : YYYYMMDD
    Output: YYYMONTHDD

Is this correct?  [Y]/N
>y
```

Two-digit representation of year is allowed only as input to program DATES, but must be converted to three- or four-digit format in DATES output. When this conversion is required, the following display appears:

```
Two-digit representation of year found in input.

Enter [1] to convert to a 20th Century date (1900's)
       2. to convert years 00 through 09 to a 21st Century date (2000's)
       3. to select alternate boundary dates between 20th and 21st Cent.
>3
```

Choice **1** above assumes that all two-digit dates occur prior to the year 2000. Choice **2** assumes that two-digit dates from **00** through **09** represent years 2000 through 2009, while dates from **10** through **99** represent years 1910 through 1999. These arbitrary boundaries can be overridden with choice **3,** which produces the following screen display:

```
Enter 2 digit number for earliest year to convert to 21st century date.
Enter RETURN for 00, "R" to return to conversion choices.
>



Enter 2 digit number for latest year to convert to 21st century date,
Enter RETURN for 09, "R" to return to conversion choices.
>
```

In most cases all dates on a given record will be converted in the same way. However, different formats can be accommodated. For example, if Birth Date and then Exit Date have been selected for conversion, the following display appears immediately after the format for Birth Date has been set:

```
Format of date representation for BIRTH

Input : YYYMMDD
Output: YYYYMMDD

Same format for EXIT? [Y]/N
>
```

Entry of **N** returns the format selection display, which need be used only if more than one date format is used in the input file. New dates replace those selected for conversion, except where constants are added to or subtracted from the original, in which case new dates are appended as an item at the end of the record.

H Entry of **H** produces the following help display:

Here are some calendar date formats accompanied by examples of their respective representations of January 1, 2000:

PEDSYS Format		Non-PEDSYS formats recognized by program DATES				
YYYYMMDD	YYYY	month dd, yyyy	DDMONYYYY	YYY.F	j1	
20000101	2000	January 01, 2000	01Jan2000	000.003	36525	
YYYMMDD	yyy	dd-mon-yyy	yyy/mm/dd	yyyyj		
0000101	000	01-Jan-000	000/01/01	2000001		

* Two-digit representation of year (YY and YYMMDD) is permitted as INPUT to DATES, although it is not a legitimate PEDSYS date format, and will result in an error message if encountered in input data to other PEDSYS programs. DATES will not allow two-digit representation of year in its OUTPUT.

* Three-digit representation of year (YYY and YYYMMDD) is permitted. Calculations using three-digit format assume that dates lie between A.D. 1200 and A.D. 2199, inclusive.

* Four-digit representation of year (including YYYY and YYYYMMDD) is always preferred.

* Abbreviated month (indicated by month code MON) results in a three-character representation (i.e., 01 Jan 1991). Full alphabetic month (specified by month code MONTH) results in full spelling of the month name.

* Gregorian and Julian forms may not be mixed, except to the extent that representing the partial year as number of days since 1 January (month code J) may be thought of as Julian.

* Year, month and day may be separated by a period, dash, comma, forward slash (/), or a blank. Any of these characters may be followed by a single additional blank. Use of a separator is not required.

* Simple conversion of date format is initiated by entering a number corresponding to an item containing a date.

* A constant number of years, months or days may be added to (or subtracted from) a date by following the number corresponding to the date with a plus (or minus) sign followed by the constant and a Y, M or D. Examples: 17-2.5y, 13+30d, where "2.5" and "30" are constants. (Note: days may be expressed as integers only.)

* The contents of a numerical item may be added to (or subtracted from) a date by beginning the command line with an asterisk. Examples: *17-23y, *13+22d, where "23" and "22" are the numbers of items containing numerical values. (Note: days may be expressed as integers only.)

Enter [C] to continue.
>

Examples:

Here are examples of legitimate PEDSYS date formats accompanied by examples of their respective representations of **January 1, 2000**:

YYYYMMDD	YYYY	YYYMMDD	YYY
20000101	2000	0000101	000

Note: Two-digit representation of year (YY and YYMMDD) is permitted as *input* to program DATES, although it is not a legitimate PEDSYS date format, and will result in an error message if encountered in input data to other PEDSYS programs. DATES will not allow two-digit representation of year in its *output*. Three-digit representation of year (including YYY and YYYMMDD) are permitted. Calculations using three-digit format assume that dates lie between A.D. 1200 and A.D. 2199, inclusive.

Here are some non-PEDSYS calendar date formats recognized by program DATES, accompanied by examples of their respective representations of **January 1, 2000**:

month dd, yyyy	DDMONYYYY	YYY.F	j1
January 01, 2000	01Jan2000	000.003	36525
dd-mon-yyy	yyy/mm/dd	yyyyj	
01-Jan-000	000/01/01	2000001	

Notes:

- Preferred PEDSYS date format (YYYYMMDD) conforms to International Standard ISO 8601 (see Appendix A, Data Types).

- Two-digit representation of year is not allowed in PEDSYS except as input to program DATES. Three-digit representation of year assumes that dates lie between A.D. 1200 and A.D. 2199.

- Integer values are preferably expressed with leading zeros to fill out full field width in output (January 01, 2000; 000/01/01; etc.). However, DATES will correctly interpret single-digit representation of month or day so long as this digit is separated from other numbers by an alphabetic string or character (1Jan1999, 1/1/2000, etc.).

- Abbreviated month (indicated by month code **MON**) results in a three-character representation (i.e., 01 Jan 1999). Full alphabetic month (specified by month code **MONTH**) results in full spelling of the month name.

- In the case of simple date format conversion, fractional representation of the partial year (indicated by the **F** code) is carried out to three decimal places. Denominator of fraction is 365 days except for Leap Years, when it is 366 days.

  The fractional part of a year added to (or subtracted from) a date as a constant, however, is computed on the basis of a 365-day year (that is, Leap Years are not taken into account). Likewise, fractional months are based on a constant 30-day month. If more precision than this is needed, it is best to express an added age or time interval as number of days.

- Gregorian and Julian forms may not be mixed, except to the extent that representing the partial year as number of days since 1 January (month code **J**) may be thought of as Julian.

- Year, month and day may be separated by a period, dash, comma, forward slash (/), or a blank. Any of these characters may be followed by a single additional blank.

- PEDSYS date format does not permit the use of separators. If separators are inserted, the resulting output will be designated as Data Type **C**, rather than Data Type **D** in a PEDSYS file.

- Blank or zero date fields in the input file are output as blank fields in output file **dates.out**. A full eight-digit field of **9**s or **UNK** (or **unk**) appearing in the first three characters of the field representing unknown dates in the input file will be left unchanged in the output file.

- If output format specifies more date elements than appear in input fields, the trailing portion of the output date will be filled with zeros if output is all numerical or blanks if alphabetic month is specified. Examples are shown in the table below, where the result of converting incomplete or abridged input dates to a more complete format is given:

	Output format specified	
Form of input date	month dd, yyyy	YYYY/MM/DD
19910100 (or 1991Jan00)	January 00, 1991	1991/01/00
19910000 (or 1991)	1991	1991/00/00
etc.		

- A constant day and/or month cannot be added (or subtracted) from an abridged input date that contains only year.

- Uninterpretable dates are written with a message to the error log file **dates.err**.

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **dates subset.out**.

Files:

Input: Any Master File or Data File and an associated Code File. Recursive use of output files is permitted, that is, file **dates.out** may be used as input to program DATE.

Output: **dates.out** contains all records of the input file in their original order, with specified date fields changed according to format codes. New dates replace those selected for conversion, except where constants are added to or subtracted from the original, in which case new dates are appended as an item at the end of the record.

dates.cde defines the record structure of **dates.out**.

dates.err contains information about records in which dates cannot be re-formatted.

Program limits:

Formats of 40 different dates be changed.

DBFILER

Program DBFILER is used to create and modify the Directory File **DBFILES**, making it possible to add or remove Master Files, Data Files, and sub-directories from a database. Changes are made only to **DBFILES** resident in the current parent directory, and to **DBFILES** resident in sub-directories resident in the parent directory.

Introductory Display:

DBFILER responds somewhat differently depending upon the prior existence of copies of the Directory File. If **DBFILES** already exists, the following Introductory Display appears:

```
PEDSYS program DBFILER - Copyright 1992, 1999 SFBR

Program DBFILER is used to create and modify the Directory File DBFILES
making it possible to add or remove Master Files, linked, and/or unlinked
Data Files from a database.

Select Directory
--------------------------------------------------------------------------------
  1. Example
  2.   LOCUS Sub-directory

Enter number of Directory or Sub-directory in which changes will be made,
or "Q" to quit.
>1

DBFILES.OLD already exists.  Enter RETURN to overwrite it and continue,
or "Q" to QUIT program.
>

Enter -  E to EDIT DBFILES entry
         A to ADD a new file or Sub-directory to DBFILES
         D to DELETE a file or Sub-directory from DBFILES
         N to edit NOTE
         V to VIEW DBFILES
         R to RESELECT directory
         Q to QUIT program
>
```

Here, the Example directory (choice **1** above) has been selected. If a sub-directory is chosen (choice **2** above), alternate instructions in the functions display will read:

```
Enter -  E  to EDIT DBFILES entry
         A  to ADD a new file to Sub-directory DBFILES
         D  to DELETE a file from Sub-directory DBFILES
         N  to edit NOTE
         V  to VIEW Sub-directory DBFILES
         R  to RESELECT directory
         Q  to QUIT program
>
```

If no copy of **DBFILES** exists in the current directory, the Introductory Display will read as follows:

```
PEDSYS program DBFILER - Copyright 1992, 1999 SFBR

Program DBFILER is used to create and modify the Directory File DBFILES,
making it possible to add or remove Master Files, linked, and/or unlinked
Data Files from a database.

Enter RETURN to create a new DBFILES, or "Q" to QUIT program.
>

DBFILES has been created.

Enter - E to EDIT DBFILES entry
        A to ADD a new file or Sub-directory to DBFILES
        D to DELETE a file or Sub-directory from DBFILES
        N to edit NOTE
        V to VIEW DBFILES
        Q to QUIT program
>
```

Commands:

E -- Command **E** (Edit a Descriptor Record) results in the following display:

```
 EDIT File Group
--------------------------------------------------------------------------------
  1. Example Master File
  2. Markers + Quant. P'types
  3. Blood Sample Inventory
  4. Hemoglobin Levels
  5.   LOCUS Sub-directory

Enter file number to be changed, or [R] to return to initial function menu
>2
```

Here, the Descriptor Record for the Markers + Quant. P'types file has been selected for modification. The edit sequence is the same as that described in the ALTERDAT section. Letters in parentheses in the display below (and in subsequent displays) refer to Notes (a) through (s) for the section describing the Directory File in Appendix A. In this example, a single password has been added.

Note: The asterisk marking line 4. (**Field in data pointer file**) *indicates that this variable may not be edited.*

```
EDIT QUANTGEN.XMP
-------------------------------------------------------------------------------
  1.  File extension                     >XMP<
  2.  File name                          >QUANTGEN<
  3.  Label                              >Markers + Quant. P'types <
* 4.  Field in data pointer file         > 1<
  5.  Multiple-entry Data File ? (T/F)   >F<
  6.  ALTERDAT.LOG normally ON ? (T/F)   >T<
  7.  1st passwor                        >pwd1<
  8.  2nd password                       >    <
  9.  3rd password                       >    <
 10.  4th password                       >    <

Enter the number of the field to be modified. Enter "S" to SAVE or [C] to
CONTINUE without saving changes.
>
```

Selection of choice **5** (specifying the Locus Sub-directory) in the **Edit File Group** display above results in the following display:

```
 EDIT Sub-directory name
-------------------------------------------------------------------------------

>LOCUS       < - Old
>            < - New
```

Note that only the name of the sub-directory may be changed at this point. To modify sub-directory files, the sub-directory itself must be selected in the introductory directory selection menu above.

A-- Command **A** (Add a Descriptor Record to DBFILES) makes the corresponding file accessible to PEDSYS programs, but does not affect the file itself. Entry of command **A** results in the following display:

```
 ADD DBFILES entry
-------------------------------------------------------------------------------

Enter -  M to add a Master File
         L to add a Linked Data File
         U to add an Unlinked File
         S to add a Sub-directory
        [R] to return to initial function menu
>
```

ADD Options:

M Addition of a Master File is normally done only in a new directory that (a) contains a new Master File (and its associated Code File and Master Pointer File), and (b) does not yet contain a copy of **DBFILES**. Program DBFILER will create a new copy of **DBFILES** in this case. Option **M** produces the following sequence:

```
 ADD Master File
--------------------------------------------------------------------------------

>DOG< - Enter a 3-character file extension identifying the new Master File.


>Beagles    < - Enter population identifier for the new Master File.

MASTER.DOG may be protected with up to 4 passwords:

>pwd1< - Enter 1st password
>pwd2< - Enter 2nd password
>    < - Enter 3rd password

>Beagle Master File       < - Enter label for MASTER.DOG
```

Here, a Master File Descriptor Record with extension **.DOG**, population label **Beagles**, two passwords, and file label **Beagle Master File** has been added to (a newly created) **DBFILES**.

L Option **L** causes a Descriptor Record for a Linked Data File to be added to a previously created **DBFILES** that already contains a Master File Descriptor Record:

```
 ADD Linked Data File
--------------------------------------------------------------------------------
  1. MASTER.XMP

Enter the number corresponding to the Master File to which the Data File
will be linked.
>1
```

When the new file name and a Master File have been selected, the following display appears:

```
 ADD Linked Data File
--------------------------------------------------------------------------------

>NEWDATA < - Enter Data File name (without extension).

Is this a Multiple-entry Data File?  [Y]/N
>n

Set ALTERDAT.LOG normally ON for this file?  [Y]/N
>y

NEWDATA.XMP may be protected with up to 4 passwords:

>pwd1< - Enter 1st password
>    < - Enter 2nd password

>New Data File            < - Enter label for NEWDATA.XMP
```

Here, a linked Single-entry Data File named **NEWDATA.XMP** has been added to the **.XMP** Directory. A log of changes (**alterdat.log**) will be created and automatically updated each time this file is modified with program ALTERDAT, and two passwords have been assigned. Addition of a Linked Data File Descriptor Record adds a new column of pointers to file **DPOINT.<EXT>** when program DATALINK is run.

Note: a Linked Data File Descriptor Record cannot be added unless **DBFILES** *contains an appropriate Master File Descriptor Record, and the Master File itself is present in the Directory.*

U Option **U** results in addition to **DBFILES** of a Descriptor Record for an unlinked Data File. This operation requires only that a file name be supplied. Although a file name is added to **DBFILES**, no actual file is created in the directory.

S Option **S** results in addition to **DBFILES** of a Descriptor Record for a sub-directory. This operation requires only that a sub-directory name be supplied. Addition of a sub-directory name to DBFILES results in creation of a sub-directory in the current parent directory.

D (DELETE)

Deletion of a Descriptor Record from **DBFILES** removes the corresponding file name from the PEDSYS File List Display, but does not affect the file itself. Removal of one unlinked file does not affect other files. Removal of a Linked Data File has the effect of removing the corresponding column of pointers on records in file **DPOINT.<EXT>**.

```
 DELETE DBFILES entry
-------------------------------------------------------------------------------
  1. Example
  2.   LOCUS Sub-directory

Enter number of the entry above, or [R] to return to initial function menu.
>1
```

Deletion of a Master File results in removal of all its linked Data Files from **DBFILES**. When this option is selected, the following warning display appears:

```
DELETE DBFILES entry
-------------------------------------------------------------------------------
WARNING:  Removing MASTER.XMP will also cause the following linked
Data Files to be removed from DBFILES:

    QUANTGEN.XMP
    SAMPLES.XMP
    HEMOGLOB.XMP

Delete MASTER.XMP from DBFILES?  [Y]/N
>
```

- ***Important warning:*** *DBFILER begins modification of DBFILES as soon as the last command in the EDIT, ADD or REMOVE sequence has been entered. Do not interrupt execution of the program before notification of successful completion of the changes is displayed on the screen. This is particularly important when adding or removing a Linked Data File, since these operations modify file* **DPOINT.<EXT>**, *as well as* **DBFILES**.

N (Edit NOTE)

As mentioned in Note (c), Section 9 of Appendix A, it is often useful to include a note on line 1 of DBFILES, reminding the user of the name of the database, its date of creation, etc. Text information of this sort can be added (or edited) with the **N** command.

```
 EDIT Notes
--------------------------------------------------------------------------------
        Master: Master Files
        Linked: Linked Data Files
      Unlinked: Unlinked Data Files
 Sub-directory: Subdirectories

Enter -  M  to edit Master File Note
         L  to edit Linked File Note
         U  to edit Unlinked File Note
         S  to edit Sub-directory Note
        [R] to return to initial function menu
>s
```

Here, the sub-directory note has been selected for change. RETURN produces the following display:

```
 EDIT Sub-directory Note
--------------------------------------------------------------------------------

Enter new note, one or more blanks to remove note, or RETURN to continue
without changing.
>Sub-directories (1999/02/15)                                              <
```

Here we have added a date to the original note ‘Sub-directories’ , making it ‘Sub-directories (1999/02/15)’.

V (VIEW)

Command **V** displays the current contents of **DBFILES**:

```
 VIEW DBFILES
--------------------------------------------------------------------------------
  1 Master Files
XMP          Example Master File       Example
  4 Linked Data Files
XMP QUANTGEN Markers + Quant. P'types   1 F T
XMP SAMPLES  Blood Sample Inventory     2 T T
XMP HEMOGLOB Hemoglobin Levels          3 T T
XMP NEWDATA New file example            4 T T
  3 Unlinked Data Files
PEDTRTEST.XMP
TRANSTEST.XMP
WEIGHTS.XMP
  1 Sub-directories (1999/02/15)
LOCUS

Press RETURN to continue.
>
```

This display shows the effects on the **.XMP** Directory File of previous example operations described above: the password added in to the Master File the EDIT sequence can be seen in line 2, the descriptor record for new Linked Data File (**NEWDATA.XMP**) can be seen in line 7, addition of the date to the sub-directory note appears in line 7. Comparing this example with the original **DBFILES** shows also that the new Linked Data File has been assigned to position 4 in the Data Pointer File **DPOINT.XMP**, and that the Data File Count record (line 3) has been incremented accordingly.

R (RESELECT DIRECTORY)

The RESELECT DIRECTORY command appears only when one or more sub-directories is present. When selected, the following screen display appears:

```
Select Directory
--------------------------------------------------------------------------------

  1.  EXAMPLE
  2.    LOCUS Sub-directory

Enter number of Directory or Sub-directory in which changes will be made,
or "Q" to quit.
>
```

Error Messages:

The following error messages may appear on the screen:

```
ERROR. No files named in DBFILES.
```

This error results in an attempt to edit or remove a file from an empty **DBFILES**. Only ADD mode is permitted in this case.

```
ERROR. MASTER.<EXT> does not exist.
ERROR. MASTER.<EXT> contains no records.
```

These errors occur when attempting to add a Data File to a non-existent or empty Master File, respectively.

```
ERROR. The extension of the Data File must be the same as the extension
       of the Master File to which it is linked.
```

This error occurs when attempting to add a Data File with the wrong extension. The error will not occur if the name is entered without an extension, in which case the it is automatically assigned the same extension as the Master File.

```
ERROR. No new Data Files with extension .<EXT> can be added until DATALINK has been
       run.
```

Program DATALINK can link a single Data File to a Master File each time it is run. DBFILER reflects this limit and permits addition of only one Data File to **DBFILES** at a time.

```
ERROR. Discrepancy between number of Data Files specified DPOINT.<EXT>,
       (<nfiles1>) and in DBFILES (<nfiles2>).
```

where <nfiles1> and <nfiles2> are the respective file counts.

Records in the Data Pointer File contain a separate pointer field for each Data File linked to the Master File. If the number of pointer fields does not match the number of Data Files listed in the Data File Count Record, there is a danger that the correct order of pointers has been lost, which could lead to serious damage to data. To recover from this error it will be necessary to delete **DPOINT.<EXT>** and **DBFILES**, and to start the DBFILER-DATALINK cycle from the beginning until all files are correctly linked and listed in **DBFILES**.

```
ERROR.  <FILENAME>.<EXT> is already listed in DBFILES.
```

This error occurs when attempting to add a Master File or Data File that is already listed in **DBFILES**.

Notes:

- We recommend using DBFILER for all modifications to the Directory File. This is particularly important when adding or removing Linked Data Files, which requires not only changes to **DBFILES**, but also addition or deletion of the appropriate pointer field in the Data Pointer File **DPOINT.<EXT>**.

- It is important to understand the conditions under which program DBFILER modifies actual file and directory content.

  1. A new copy of **DBFILES** is created each time DBFILER is run.

2. **DBFILES .OLD** is created whenever DBFILER is run in a directory that already contains **DBFILES**.

3. The DBFILER **ADD** function adds only file and sub-directory *names* from **DBFILES**; it does not create files or sub-directories.

4. Likewise, the DBFILER **DELETE** function removes only file and sub-directory *names* from **DBFILES**; it does not delete files or sub-directories.

5. Records in **DBFILES** are added, deleted and changed as the result of every program function.

6. Records in **DPOINT.<EXT>** are modified when **ADD** or **DELETE** functions are used with Linked Data Files.

7. The contents or structure of Master Files or Data Files are not affected by DBFILER.

- The only operations permitted with a newly created or empty **DBFILES** (that is, one in which the number of Master Files is 0) are to ADD a Master File, unlinked Data File or sub-directory.

- A Data File may be added only after an existing Master File has been added. When adding the first Data File, a Data Pointer File will be created if it does not already exist. However, at this stage **DPOINT.<EXT>** will be incomplete until program DATALINK has been run to establish pointers.

- Only one new file may be specified as a Linked Data File in a single session or DBFILER. Additional Linked Data Files may be specified only after the previous one has been linked using program DATALINK.

- Although the number of Data Files in the Data File Count Record is displayed while in Edit Mode, the field is set automatically by DBFILER, and cannot be changed manually.

Files:

Input: A non-empty **DBFILES** is required in REMOVE or EDIT mode.
A Master File is required when adding a Data File.

A Data Pointer File **DPOINT.<EXT>** is required when adding or removing any Linked Data File other than the first.

Output: **DBFILES** is the newly created Directory File; **DBFILES.OLD** is a backup copy of the Directory File prior to running the program.

DPOINT.<EXT> is created when the first Linked Data File is added.

DPOINT.<EXT> is modified and rewritten when a Linked Data File is added or deleted.

DELRECS

DELRECS is a second program of the two-step process required to make changes in both Master Files and Data Files (linked or unlinked). The program uses information from special-purpose output files named **DELETE.<EXT>** produced by program ALTMASTR in the case of Master Files, and **<filename>.DEL** produced by program ALTERDAT in the case of Data Files.

Introductory Displays:

```
PEDSYS program DELRECS - Copyright 1992. 1999 SFBR

DELRECS updates a Master File from changes stored in DELETE.<EXT>. Master
File and linked Data File records are deleted, and MPOINT.<EXT> and
DPOINT.<EXT> are regenerated.  Files as they were before modification are
saved with extension .OLD, and DELETE.<EXT> is renamed DELETE.OLD.

DELRECS updates a Data File from changes stored in <datafile>.DEL. Data
File records are deleted, and the file as it was before modification is saved in
<datafile>.OLD.  <datafile>.DEL is renamed <datafile>.ODE.  DPOINT.<EXT> is
regenerated when deletions occur in a Linked Data File.

1.  Unlinked file
2.  EXAMPLE
3.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>
```

Error Messages and Displays:

Program DELRECS is usually run as a last step in permanently updating Master Files or Data Files. Because its output plays such a critical part in the organization of pedigree databases, considerable effort has gone into preventing errors from being incorporated into the data. Protection from permanent damage is provided at three levels:

a. Errors that affect single records are recorded in the **delrecs.err** file, and the record in question is not deleted.

b. Whenever possible, the program is stopped before the final creation of new files when errors are found that might affect file structure.

c. Backup copies of original input files are created so that it is always possible to discard incorrect changes and return to an earlier version of the data.

Linked Files

There must be a record in the Master File for every PID to be deleted from a Linked Data File. The following errors may be reported if this condition is not met (the number of each error appears in parentheses at the end of the error record).

```
<ego>: Record in <FILENAME>.<EXT> not deleted; EGO not found in MF          (01)
<ego>: Record found in <FILENAME>.<EXT> that is not in Master File          (02)
```

where **<ego>** refers to EGO's PID as it was entered in the **DELETE.<EXT>** or **<FILENAME>.DEL** file, and **MF** is an abbreviation for 'Master File'.

The usual cause of error 1 is an attempt to delete from a Data File that is incorrectly linked to the Master File. To correct the error, run program DATALINK. Error 2 is a generalized linkage error that occurs when attempting to establish pointers to incorrectly created Data Files. See notes for program DATALINK for suggested corrections.

Unlinked Files

If records identified by PID (or other identifier) and record number in file **<filename>.DEL** are not found in an unlinked Data File, the following error will be reported:

```
<ID>: found in <filename>.DEL not in record <n> of <filename>.<ext>         (03)
```

The usual cause of error 3 is a change in the unlinked Data File since the creation of the file containing PIDs of records to be deleted, **<filename>.DEL**. To correct this error, either run DELRECS with the original Data File, or generate a new **<filename>.DEL** by selecting records for deletion in the current Data File using program ALTERDAT before running DELRECS again.

Notes:

- If both additions and deletions have been made to a Master File (that is, if ALTMASTR has produced both **UPDATE.<EXT>** and **DELETE.<EXT>**), both SETMASTR and DELRECS must be run to complete the update process. SETMASTR should be run before running DELRECS.

- If both additions and deletions have been made to a Data File (that is, if ALTERDAT has produced both **<filename>.UPD** and **<filename>.DEL**), both SETDATA and DELRECS must be run to complete the update process. SETDATA should be run before running DELRECS.

Files:

Case 1. When deleting records from a Master File

Input: Any Master File and its associated Code File.

Any linked Data Files from which records must be deleted in parallel with deletions from the Master File.

DELETE.<EXT> produced by program ALTMASTR, containing PIDs of individuals to be deleted from the Master File.

Output: **MASTER.<EXT>** is the updated Master File produced by the program.
MASTER.OLD is a backup copy of the input Master File.

DELETE.OLD is a backup copy of **DELETE.<EXT>**.

delrecs.err contains a record and diagnosis of deletions that were attempted, but were unsuccessfully completed because of an error or inconsistency detected by the program.

DPOINT.<EXT> (the Data Pointer File) is rewritten when the Master File is updated.

<filename>.<EXT> is an updated Data File from which records have been deleted. More than one Data File may be updated.

<filename>.OLD is a backup copy of a Data File as it was before deletions were made. There may be backups of more than one Data File.

Case 2. When deleting records from a Linked Data File

Input: Any Linked Data File from which records are to be deleted, and its associated Code File.

<filename>.DEL produced by program ALTERDAT, containing IDs of records to be deleted from the Data File.

Output: **<filename>.<EXT>** is the updated Data File from which records have been deleted.

<filename>.OLD is a backup copy of the Data File as it was before deletions were made.

<filename>.ODE is a backup copy of **<filename>.DEL**.

delrecs.err contains a record and diagnosis of deletions that were attempted, but were unsuccessfully completed because of an error or inconsistency detected by the program.

DPOINT.<EXT> (the Data Pointer File) is written when a Linked Data File is updated. This will be a modified version of the previously created file.

Case 3. When deleting records from an unlinked Data File

Input: Any unlinked Data File from which records are to be deleted, and its associated Code File.

<filename>.DEL produced by program ALTERDAT, containing PIDs of records to be deleted from the Data File.

Output: **<filename>.<EXT>** is the updated Data File from which records have been deleted.

<filename>.OLD is a backup copy of the Data File as it was before deletions were made.

<filename>.ODE is a backup copy of **<filename>.DEL**.

delrecs.err contains a record and diagnosis of deletions that were attempted, but were unsuccessfully completed because of an error or inconsistency detected by the program.

Program Limits:

Records for a maximum of 250 individuals may be deleted at one time.

[1]DOWNCODE

DOWNCODE simplifies the genotypic structure of a locus by combining selected alleles into a single type. The program iteratively combines pairs of alleles, and evaluates loss of information about parental origins of each allele. *Analysis is limited to autosomal codominant loci only.*

Introductory Displays:

```
PEDSYS program DOWNCODE - Copyright 1994, 1999 SFBR

DOWNCODE simplifies the genotypic structure of a locus by combining selected
alleles into a single type.

  1.  Unlinked File
  2.  merge.out
  3.  EXAMPLE
  4.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>
```

Loci may be selected for analysis from linked or unlinked files containing genotypes or phenotypes for one or more loci. When a Linked Data File containing genotypic or phenotypic data has been selected, DOWNCODE proceeds directly to selection of loci for analysis and processing. File **merge.out** (choice **2** above) shown below is an example of multiple genotypes (or phenotypes) in an unlinked file (display from program BROWSE):

```
 merge.out                                Items: 5       Top record: 1/32
  (1)   (2)   (3)  (4)  (5)  (6)   (7)
  ID    SIRE  DAM  D1S  D1S  D2S  SAMPLE
                   1656 515  1329 NUMBER

 ID015 ID062 ID047 BB   BB   DE    W236
 ID068 ID062 ID047 AB   AC   AB    W242
 ID054 ID063 ID065 AD   CC   AF    W234
 ID070 ID063 ID065 DD   CC   EE    W214
 ID077 ID063 ID065 DE   AC   EF    W216
 ID067 ID015 ID019 BB   AB   AE    W246
 ID071 ID045 ID034 BE   AB   BD    W244
 ID049 ID045 ID034 BE   CD   DD    W248
 ID069 ID054 ID068 AA   AC   AA    W250

Enter [+]<n> or -<n> for new page; <n> (1-32) to set top row; ">" or "<"<n>
to shift window; I<n> (1-5) to set window; L to lock columns; N for new
file; F to find; H for help; Q to quit.
>
```

When an unlinked file has been selected, DOWNCODE compares its Code File with entries in **CODE.STD** for standard mnemonics identifying Ego and Ego's parents. If the standard mnemonics **FA**

[1]

(or **SIRE**) and **MO** (or **DAM**) are found in the input Code File, the following query appears:

```
Use pedigree information found in merge.out? [Y]/N
>
```

If the answer is **N**, , or if the Code File does not contain standard mnemonics), the following display appears:

```
 1. Unlinked file
 2. Example Master File

Enter number or name of file containing pedigree.
>2
```

Here, the Example Master file has been chosen. When a source of pedigree information has been selected, the following display appears, from which loci are selected for analysis:

```
   1 EGO Permanent ID        4 D1S1656 Genotypes       7  Sample No.
   2 Sire's Permanent ID     5 D1S515 Genotypes
   3 Dam's Permanent ID    * 6 D2S1329 Genotypes
--------------------------------------------------------------------------------
  DOWNCODE    Select loci                    File - merge.out
--------------------------------------------------------------------------------
Enter number corresponding to loci for which genotypes/phenotypes are to
be downcoded, or [C] to continue.
>
```

Here, one locus (**D2S1329**) has been chosen. DOWNCODE makes two initial assumptions about genetic loci. First, it assumes that data are *phenotypic* unless one of the last three mnemonic words associated with the Code File entry for a locus is "**GENO**" or "**G'TYPE**", in which case the data are interpreted as *genotypic*. Second, the program assumes that phenotypes are autosomal codominant and follow conventions listed in Appendix C, Rules for Genotype and Phenotype Nomenclature.

Once loci have been selected, interactive or automatic mode must be selected:

```
Enter "I" to run DOWNCODE interactively, or enter RETURN alone to run
automatically.
>i
```

Here, **interactive mode** has been selected. The program begins its execution with an initial enumeration of alleles for the first locus (which may take a moment if the data file is large), followed by a request for the number of alleles desired after downcoding:

```
Processing D2S1329 Genotypes...

   6 alleles were found for D2S1329 Genotypes.

Enter the number of alleles desired (2 minimum).
>4
```

Here, a reduction to four of the six alleles found initially in the pedigree has been specified. In **Interactive mode**, the following display appears (this step is skipped in **Automatic mode**):

```
 D2S1329 Genotypes                                                  Page  1/ 1
-------------------------------------------------------------------------------
 Current/Start:     Alleles: 6/6,     Informative triplets: 13/13

       Allele Equivalence       Triplets    Heterozygotes   Nuc. Fams.
                                  Lost          Lost        Affected
  1.  Change C     to F            0             0             0
  2.  Change C     to B            0             0             0
  3.  Change C     to D            0             0             0
  4.  Change C     to A            0             0             0
  5.  Change D     to A            0             1             0
  6.  Change C     to E            0             2             0
  7.  Change F     to B            1             0             1
  8.  Change F     to D            1             1             1
  9.  Change B     to A            1             1             1
 10.  Change B     to D            1             3             1
 11.  Change F     to A            2             2             1
 12.  Change B     to E            2             2             2
 13.  Change D     to E            2             2             2
 14.  Change F     to E            2             3             2
 15.  Change A     to E            3             4             3

Enter RETURN for Equivalence 1, or enter number of alternate selection
(precede number of selection with "R" to reverse). Enter "X" to stop
downcoding at this stage.
>1
```

The second line in the display above gives the status of the enumeration. Since no downcoding has yet been done, the current and initial counts are identical (**6** in each category). Below the status line is a table of potential downcoding changes and their consequences:

- **Allele Equivalence** refers to the actual downcoding process, that is, making two alleles equivalent by changing the symbol of one to the symbol of another.

- **Triplets Lost** enumerates triplets that would no longer be informative for the locus selected if the particular downcoding equivalence is chosen.

- **Heterozygotes Lost** enumerates heterozygous genotypes that would be changed to homozygotes if

the particular equivalence is chosen.

- **Nuc. Fams. Affected** enumerates the families in which some loss of information would result if the particular equivalence is chosen.

Usually, less information is sacrificed when rare alleles are eliminated, and as a convenience, changes are listed in the table in order of increasing loss of information. In this example the default change **1** will eliminate rare allele **C** by changing it to allele **F**. From the table above it can be seen that changes **1** through **4** will result in no loss of informative triplets or heterozygotes in any family, and so any of these is a good candidate for downcoding. For this example, we have chosen change **1**, which produces the following result:

```
 D2S1329 Genotypes                                                  Page  1/ 1
-------------------------------------------------------------------------------
 Current/Start:     Alleles: 5/6,     Informative triplets: 13/13

       Allele Equivalence       Triplets    Heterozygotes   Nuc. Fams.
                                  Lost          Lost        Affected
   1.  Change D     to A            0             1             0
   2.  Change B     to F            1             0             1
   3.  Change F     to D            1             1             1
   4.  Change B     to A            1             1             1
   5.  Change B     to D            1             3             1
   6.  Change F     to A            2             2             1
   7.  Change B     to E            2             2             2
   8.  Change D     to E            2             2             2
   9.  Change F     to E            2             5             2
  10.  Change A     to E            3             4             3

Enter RETURN for Equivalence 1, or enter number of alternate selection
(precede number of selection with "R" to reverse). Enter "X" to stop
downcoding at this stage.
>1
```

The status line now reports that three alleles remain now that **C** is gone, and that no informative triplets have been lost. The list of potential changes shows that a change of allele **D** to allele **A** would result in the least loss of information. When the change has been selected, DOWNCODE reprocesses the data for this locus. Since a second iteration is sufficient to achieve the desired number of alleles in the example, no further displays are shown, and the program then proceeds to the next locus, if selected, or writes its output files and terminates.

In some pedigrees rare alleles are informative, and their loss by downcoding can be avoided by

a. Overriding the default order by choosing a higher numbered change, or

b. Reversing the order of the change by preceding the number with an **r**. For example, entering **r1** would result in changing allele **F** to allele **C**, instead of changing **C** to **F**.

To see the effects of DOWNCODE, we choose as an example locus D1S3721, which is a more realistic candidate for downcoding than that shown above. Note first the table below, which gives the

distribution of alleles and genotypes, and their respective frequencies for locus D1S3721 (output of program GENEFREQ):

D1S 3721	N	Typed	Genotype	Count	Freq.	Gene	Freq.
(AUCD)	32	32	158 168	1	0.0312	158	0.0938
			158 170	1	0.0312	164	0.0156
			158 176	3	0.0938	166	0.0156
			158 180	1	0.0312	168	0.1094
			164 176	1	0.0312	170	0.3437
			166 176	1	0.0312	176	0.2188
			168 170	4	0.1250	178	0.0156
			168 178	1	0.0312	180	0.1719
			168 180	1	0.0312	186	0.0156
			170 170	3	0.0938		
			170 176	6	0.1875		
			170 180	4	0.1250		
			170 186	1	0.0312		
			176 180	3	0.0938		
			180 180	1	0.0312		
			UNT	0			
			BLANKS	0			
(Inferred types excluded)			INFERRED	0			
			UNINTERP	0			

Note that nine alleles and 15 genotypes are shown. Compare this table with the one below derived from the same locus after it has been processed by DOWNCODE with reduction to five alleles specified:

D1S 3721 DWNCDE	N	Typed	Phenotype	Count	Freq.	Gene	Freq.
(AUCD)	32	32	158168	1	0.0312	158	0.0938
			158170	1	0.0312	168	0.1094
			158176	3	0.0938	170	0.3436
			158180	1	0.0312	176	0.2188
			168170	4	0.1250	180	0.2344
			168180	2	0.0625		
			170170	3	0.0938		
			170176	6	0.1875		
			170180	5	0.1562		
			176180	5	0.1562		
			180180	1	0.0312		
			UNT	0			
			BLANKS	0			
(Inferred types excluded)			INFERRED	0			
			UNINTERP	0			

Here, it can be seen that the number of alleles has been reduced to five, and genotypes to eleven. Tabular reporting of DOWNCODE processing is also given in file **downcode.tab**, as shown below, again for locus D1S3721:

```
DOWNCODE results for D1S3721

   Starting with       9 alleles and  14 informative triplets
      Allele 164 was changed to allele 166
            Leaving    8 alleles and  14 informative triplets

      Allele 178 was changed to allele 186
            Leaving    7 alleles and  14 informative triplets

      Allele 166 was changed to allele 186
            Leaving    6 alleles and  14 informative triplets

      Allele 186 was changed to allele 180
            Leaving    5 alleles and  14 informative triplets
```

Notes:

- DOWNCODE evaluates each nuclear family for informative EGO-FA-MO triplets (that is, those in which the parental origin of EGO's alleles can be determined).
- A nuclear family contains as many triplets as there are members of the offspring sibship.
- The list of potential downcoding changes (seen in the display above) is sorted in order of increasing loss of information.
- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **downcode subset.out**.

Error Checking:

If DOWNCODE detects one or more discrepant triplets, it creates error file **downcode.err**. Contents of this file are records showing PIDs and genotypes of EGO and each of EGO's parents. An example of this file is given below:

```
DOWNCODE Error report for Error Demo Phenotype
 Ego: ID063 B2    E     Sire: ID038 D     E     Dam: ID042 C     E
 Ego: ID068 B2    D     Sire: ID062             Dam: ID047 E     E
 Ego: ID069 B2    C     Sire: ID054 E     E     Dam: ID068 B2    D
 Ego: ID071 D     D     Sire: ID045 C     E     Dam: ID034 E     E
```

Files:

Input: A Master File and a Data File having records containing genotypic or phenotypic data, with associated Code Files. The Data File need not be linked if it contains pedigree data, or if the pedigree can be supplied by another file. Recursive use of output files is permitted, that is, file **downcode.out** may be used as input to program DOWNCODE.

Output: **downcode.out** consists of a record for each individual represented in the input data file. Each record contains PIDs and genotypes of EGO and each of EGO's parents.

downcode.cde defines the record structures of **downcode.out**.

downcode.tab contains results of each iteration of the program, and contains information on numbers of alleles and loss of information resulting from combining alleles.

downcode.err contains records for discrepant triplets showing PIDs and genotypes of EGO and each of EGO's parents.

DUPLIC

Program DUPLIC identifies records containing duplicate entries, and creates output such that each item value found (a) on *one or more records* in the input file is represented by a *single record* in the output file, (b) on *two or more records* in the input file is represented by the *same records* in the output file, or (c) on *exactly one record* in the input file is represented by the *same record* in the output file.

Introductory Display:

```
PEDSYS program DUPLIC - Copyright 1992, 1999 SFBR

DUPLIC identifies records containing duplicate entries, and creates output
files such that an item value found (a) on 1 or more input records is
represented by a single record in the output file, (b) on 2 or more input
records is represented by the same records in the output file, or (c) on
exactly 1 input record is represented by the same output file record.

  1.  Unlinked file
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>
```

When the PEDSYS file has been selected that contains items to be checked for duplication, the following display appears:

```
   1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ   16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE   12 EGO Permanent ID      *19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE   13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  DUPLIC      Select grouping criteria        File - MASTER.XMP
--------------------------------------------------------------------------------
Enter one or more numbers corresponding to key items by which records will
be grouped, or [C] to continue.
>
```

Here, we have chosen Mate's Permanent ID as the single item for comparison. Records in the input file will be grouped on the basis of this item prior to checking for duplication.

- **Option - Sorting Duplicate Records**

It is sometimes useful to order records containing duplicate entries. An opportunity to do this is provided in the next display, which allows records to be sorted within groups of duplicates:

```
   1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ   16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE   12 EGO Permanent ID      +19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE  +13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  DUPLIC      Sort within groups              File - MASTER.XMP
--------------------------------------------------------------------------------
Records will be group as marked above. Optionally, enter one or more
numbers corresponding to items by which records will be sorted within
groups (precede with "D" for descending order).  Enter [C] to continue
to next step.
>
```

In this example, EGO's Birth Date (Item **13**) has been entered for sorting in ascending order. Note that the item selected to define groups (Item **19**, Mate's PID) is preselected in this display. The within-group sorting option may be skipped if the within-group order of records is not critical.

Next, the following display is shown:

```
 [1] Keep single records: Each item value represented by one or more records
     in the INPUT file is represented by a single record in duplic.out.

  2. Keep duplicate records: Each item value represented by two or more
     records in the INPUT file is represented by the same records in
     duplic.out.

  3. Keep unique records: Each item value represented by only one record in
     the INPUT file is represented by the same record in duplic.out.

Enter number above corresponding to your choice, "Q" to quit program.
>1
```

This selection controls the output of unique and duplicate records. Here we have chosen to generate an output file consisting of a subset of Example Master File records, each of which will contain a unique Mate's Permanent ID. All Mate's PIDs found in **MASTER.XMP** will also be found in **duplic.out**, although in the latter file each will be found only once (in this case on the record of Ego having the earliest birth date). If choice **1** or **3** has been made, the following display appears:

```
Single records will be saved in duplic.out. Save excluded records in
noduplic.out?  Y/[N]
>
```

If choice **2** has been made, the following display is shown:

```
Duplicate records will be saved in duplic.out. Save excluded records in
noduplic.out?  Y/[N]
>
```

These options make it possible to divide the input file in various ways by saving records that do not fulfill the criteria for inclusion in **duplic.out**.

Notes:

- DUPLIC requires that records be sorted in an order determined by the value of items selected, and then comparing the record containing the first occurrence of any value with the value of entries in subsequent records.

- The compound sort in program DUPLIC provides some control over which record will be first in the list for comparison. The major sort should be for the item (or items) selected for comparison so that records are grouped together, but other items (such as dates) may be added as minor sorts so as to order records within groupings.

- The first record in each of the groupings is the one selected for inclusion in the Keep single records (Choice **1**) category above.

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **duplic subset.out**.

Files:

Input: Any Master File or Data File and an associated Code File. Recursive use of output files is permitted, that is, file **duplic.out** or **noduplic.out** may be used as input to program DUPLIC.

Output: duplic.out contains records from the input file in which selected entries have been found.

noduplic.out contains records from the input file that are excluded from **duplic.out**.

duplic.cde and **noduplic.cde** define the record structure of **duplic.out**.and **noduplic.out**, respectively.

FOUNDREP

Program FOUNDREP determines the extent to which population founders (individuals with both parents unknown) are represented in members of the current living population.

Introductory Display:

```
PEDSYS program FOUNDREP - Copyright 1999 SFBR

FOUNDREP determines the extent to which population founders (individuals with
both parents unknown) are represented in members of the current living
population.

IMPORTANT: The file from which the pedigree is generated must be an indexed
           pedigree file with an associated Master Pointer File.

 1.  Unlinked File
 2.  Example Master File

Enter number, file name, or "Q" to quit.
>2
```

I. Define descendant population (SUBSET first?):

All living individuals
Mating age individuals
Individuals younger than mating age

II. Define founders

All founders of living individuals
All founders of mating age individuals
All founders of individuals younger than mating age

III. Find risk of founder genome loss:

For each founder, find:

1. Number of descendants among living population.
2. Number of descendants in mating age population
3. Number of descendants in younger than mating age population

Proportions of total - i.e., N(i)/Sum(N(i))

IV. Find potential for equalizing founder representation

For each descendant (class?), find:

Proportional founder representation -

a. Order founders by risk of loss
b. For each founder, order descendant by representation of this founder

What to do with living founders?

Note:

•

Files:

Input: Any Master File with an accompanying Code File.

Output: **foundrep.out**

GENEFREQ

Program GENEFREQ calculates gene frequencies from phenotypic (or genotypic) data.

Introductory Display:

```
PEDSYS program GENEFREQ - Copyright 1994, 1999 SFBR

GENEFREQ calculates gene frequencies from genotypic (or phenotypic) data.

CAUTION: GENEFREQ uses a simple counting procedure to calculate gene
         frequencies, and is subject to limitations described in the
         PEDSYS User's Manual, Appendix A.

  1.  Unlinked File
  2.  merge.out
  3.  Example
  4.    LOCUS Sub-directory

Enter number of the file or directory to use.
>4
```

Loci may be selected for analysis in several ways.

1. An unlinked file containing genotypes or phenotypes for one or more loci

It is often convenient to organize multiple genotypes and/or phenotypes on the same record. An example of this organization is given by file **merge.out** (choice **2** above) in which genotypes for three loci have been assembled on records for each individual (display from program BROWSE):

```
merge.out                              Items: 5       Top record: 1/32
 (1)  (2)  (3)  (4)   (5)
 ID   D1S  D1S  D2S  SAMPLE
      1656 515  1329 NUMBER

 ID015 BB   BB   DE    W236
 ID019 BC   AA   AA    W240
 ID034 AE   AC   DF    W238
 ID038 AE   AC   AF    W216
 ID042 CD   AC   EF    W218
 ID045 BE   BD   BD    W230
 ID047 BB   AB   BE    W232

Enter [+]<n> or -<n> for new page; <n> (1-32) to set top row; ">" or "<"<n>
to shift window; I<n> (1-5) to set window; L to lock columns; T to change
instructions; N for new file; F to find; H for help; Q to quit.
>
```

When an unlinked file of this type is selected, the following display appears, from which loci are selected for analysis:

```
   1 EGO Permanent ID      * 3 D1S515 Genotypes        5  Sample No.
 * 2 D1S1656 Genotypes     * 4 D2S1329 Genotypes
--------------------------------------------------------------------------------
  GENEFREQ    Select loci                     File - merge.out
--------------------------------------------------------------------------------
Enter numbers corresponding to loci for which gene and genotype frequencies
are to be calculated, or [C] to continue.
>
```

Here, three loci have been chosen. GENEFREQ makes two initial assumptions about genetic loci. First, it assumes that data are *phenotypic* unless one of the last three mnemonic words associated with the Code File entry for a locus is "**GENO**" or "**G'TYPE**", in which case the data are interpreted as *genotypic*. Second, the program assumes that phenotypes are autosomal codominant and follow conventions listed in Appendix C, Rules for Genotype and Phenotype Nomenclature, unless specified otherwise in a Genotype-Phenotype Map File **GENOMAP.<EXT>** (see Appendix A, Definitions, Data, and File Types). GENEFREQ first searches the Data Directory for map file **GENOMAP.<EXT>**. If the file is not found, or if the selected phenotype is not included in it, the following display appears:

```
No entries for selected loci found in GENOMAP.XMP

Enter RETURN if all loci are autosomal co-dominant. Otherwise enter name of
alternate genotype-phenotype map file (loci not listed in a map file are
assumed to be autosomal co-dominant).
>
```

A Phenotype-Genotype Map is not required for autosomal codominant loci that adhere to the conventions listed in Appendix C. Presence of a map file eliminates the need for the tallying step and can significantly speed execution. If a Phenotype-Genotype Map File is found and it contains the selected phenotype, an alternate display appears:

```
GENOMAP.XMP contains entries for

     ABO PHENO TYPE (AUDR)

Enter RETURN to use GENOMAP.XMP.  Otherwise enter name of alternate
genotype-phenotype map file (loci not listed in a map file are assumed
to be autosomal co-dominant).
>
```

In some cases genotypes or phenotypes can be inferred from the types of other family members (see program INFER). Control over inclusion or exclusion of inferred types in gene frequency calculations is given in the next instruction:

```
INFERRED alleles and genotypes are designated by enclosure in parentheses.

Enter - E to EXCLUDE inferred types in allele frequency calculations.
        I to INCLUDE inferred types in allele frequency calculations.
>
```

When this choice has been made, the program begins execution.

2. A single locus chosen from the LOCUS Sub-directory

When the LOCUS Sub-directory is selected (choice **4** in the Introductory Display), the following display showing all files listed in its Directory File (**DBFILES**) appears:

```
   1. AllLoci
   2. CandLoci
   3. D1S1656
   4. D1S515
   5. D2S1329
   6. D2S326
   7. D3S1229
   8. D4S1582
   9. D5S1466
  10. D6S1036
  11. D7S23
  12. GGAT1A4
  13. ABO
  14. ADA
  15. APOBpvu2a
  16. CA1
  17. ErrorDemo
  18. G6PD

Enter a file number or name.
>
```

Here, any single locus may be selected by entering its corresponding file number (3 - 18). Command sequences that follow are the same as those outlined above for unlinked files in the main Data Directory.

3. Multiple loci listed in a special-purpose file in the LOCUS Sub-directory

Alternately, a pre-defined set of loci (file numbers 1 and 2) may be chosen. The content of file 1 (**AllLoci**) is shown below:

```
FILELIST
D1S1656
D1S515
D2S1329
D2S326
D3S1229
D4S1582
D5S1466
D6S1036
D7S23
GGAT1A4
ABO
ADA
APOBpvu2a
CA1
ErrorDemo
G6PD
```

Note that the first record in the file must contain only the word FILELIST, and that this record is followed by a list of file names exactly as they appear in the sub-directory DBFILES. In the example above, we have included all locus files, but the list may be constructed to include any subset as shown below:

```
FILELIST
ABO
ADA
APOBpvu2a
CA1
G6PD
```

Command sequences that follow are also the same as those outlined above for unlinked files in the main Data Directory.

- **Additional requirements for calculating X-linked gene frequencies**

When an X-linked is designated in the Genotype-Phenotype Mapping File, GENEFREQ compares the Code File of the input locus file with entries in **CODE.STD** for standard mnemonics identifying Ego and Ego's parents. If the Code File does not contain the mnemonic **SEX**, a pedigree file containing this information (along with Ego PIDs that match those in the input file) must be selected:

```
G6PD PHENO TYPE  is listed as an X-linked locus which requires that sex be
known for gene frequency calculations.

Enter - [P] to select an appropriate pedigree file
         Q  to quit program.
>
```

Notes:

- **Important cautions:** GENEFREQ calculations are approximate, and should be used only for rough estimates of the overall genetic composition of the population included in a given file. This is because

  a. Calculations are made without regard to generational position or kinship relations between individuals.

  b. The program uses a simple counting procedure to calculate gene frequencies for codominant loci. But for dominant/recessive phenotypes, the frequency of the recessive allele is estimated as

  $$Freq(q)=\sqrt{R}$$

  where *R* is the frequency of the homozygous recessive phenotype in the population. Frequencies of the dominant alleles are estimated as

  $$Freq(p_i)=\sqrt{Di+R}-\sqrt{R}$$

  where *Di* is the frequency of the *ith* homozygous dominant phenotype in the population. These estimates depend on non-zero frequencies of homozygous dominant phenotypes. To guarantee this, the count of homozygotes is changed from *0* to *1* when alleles are observed in heterozygotes, but are missing in their homozygous form in the population.

  These estimates also depend on the assumption of random mating and large population size.

  c. Gene and genotype frequencies may be adjusted slightly so that they sum to *1.0*.

- A Phenotype-Genotype Map need not be created for autosomal codominant loci that adhere to the conventions listed in Appendix C, Rules for Genotype and Phenotype Nomenclature. Nonetheless, is it frequently helpful to generate a map for autosomal codominant loci, because GENEFREQ will otherwise be forced to read through the entire Data File twice for each locus (once to tally phenotypes, and again to do the analysis). Presence of a map file eliminates the need for the tallying step and can significantly speed execution.

- Calculations made from *genotypes* take into account the use of a dash (**-**) to designate a partially known genotype (for example, **A-**). This situation arises when it is known that an untyped individual must have received or transmitted a particular allele (allele **A** in the example) because of the known genotypes of relatives, but where inferences cannot be made about the identity of the individual's other allele (shown as **-** here). In this case, the known allele is counted in calculations of gene frequency, while the dash is ignored.

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: genefreq subset.out.

Output:

Shown below are contents of output file **genefreq.tab** and **genefreq.out** which include computations done for ABO, ADA, CA1 and G6PD phenotypes and APOB Pvu2a genotype. ABO and G6PD phenotypes are listed in a Genotype-Phenotype Map File, while the remaining phenotypes are controlled by autosomal codominant loci that follow default rules for loci not in a map file.

1. genefreq.tab

An example ot output file **genefreq.tab** is shown below:

```
-------------------------------------------------------------------------------
ABO PHENO TYPE       N  Typed  Phenotype   Count    Freq.     Gene    Freq.
-------------------------------------------------------------------------------
(AUDR)              32     30  A              12    0.4000     a       0.3484
                               AB              4    0.1333     b       0.3033
                               B              10    0.3333     o       0.3483
                               O               4    0.1333
                               UNT             0
                               BLANKS          2
(Inferred types included)      INFERRED        0
                               UNINTERP        0
-------------------------------------------------------------------------------
ADA PHENO TYPE       N  Typed  Phenotype   Count    Freq.     Gene    Freq.
-------------------------------------------------------------------------------
(AUCD)              32     27  A               8    0.2963     A       0.4444
                               AB              8    0.2963     B       0.5186
                               B              10    0.3704     C       0.0370
                               C               1    0.0370
                               UNT             0
                               BLANKS          5
(Inferred types included)      INFERRED        0
                               UNINTERP        0
-------------------------------------------------------------------------------
APOB Pvu2a GENO      N  Typed  Genotype    Count    Freq.     Gene    Freq.
-------------------------------------------------------------------------------
(AUCD)              32     27  A1A1            5    0.1852     A1      0.3704
                               A1A2           10    0.3704     A2      0.6296
                               A2A2           12    0.4444
                               UNT             0
                               BLANKS          5
(Inferred types included)      INFERRED        0
                               UNINTERP        0
-------------------------------------------------------------------------------
CA1 PHENO TYPE       N  Typed  Phenotype   Count    Freq.     Gene    Freq.
-------------------------------------------------------------------------------
(AUCD)              32     30  A              10    0.3333     A       0.4833
                               AB              9    0.3000     B       0.5167
                               B              11    0.3667
                               UNT             0
                               BLANKS          2
(Inferred types included)      INFERRED        0
                               UNINTERP        0
-------------------------------------------------------------------------------
G6PD PHENO TYPE      N  Typed  Phenotype   Count    Freq.     Gene    Freq.
-------------------------------------------------------------------------------
(XLCD)              32     26  A               3    0.1154     a       0.2195
                               AB              5    0.1923     b       0.7805
                               B              18    0.6923
                               UNT             0
                               BLANKS          6
(Inferred types included)      INFERRED        0
                               UNINTERP        0
```

The first line of each segment of the table is a header describing columns below it. Mode of inheritance is shown in parentheses in the first column, as is a note to the effect that inferred alleles have been included in frequency calculations. Symbols and numbers of phenotypes found in the input file are shown in the **Phenotype** and **Count** columns, respectively. Below these are four categories with counts that are not actually included in computations:

UNT This is a reserved phenotypic symbol that indicates phenotypes that are permanently untestable, usually because the individual or its blood or tissue sample is no longer available.

BLANKS Following the usual PEDSYS convention, blank fields represent missing data.

INFERRED This is simply an enumeration of the inferred phenotypes found in the input file, and has no bearing on whether or not inferred types are included in frequency calculations.

UNINTERP This is an error category which is incremented whenever a phenotypic symbol is (a) not found in the Phenotype-Genotype Map File, or (b) does not follow the rules described above for autosomal codominant loci.

2. genefreq.out

An example ot output file **genefreq.out** is shown below:

```
ABO     PHENO   TYPE            a       0.3484
ABO     PHENO   TYPE            b       0.3033
ABO     PHENO   TYPE            o       0.3483
ADA     PHENO   TYPE            A       0.4444
ADA     PHENO   TYPE            B       0.5186
ADA     PHENO   TYPE            C       0.0370
APOB    Pvu2a   GENO    TYPE    A1      0.3704
APOB    Pvu2a   GENO    TYPE    A2      0.6296
CA1     PHENO   TYPE            A       0.4833
CA1     PHENO   TYPE            B       0.5167
G6PD    PHENO   TYPE            a       0.2195
G6PD    PHENO   TYPE            b       0.7805
```

The first four items on records in this file consist of mnemonics identifying the locus in question. These are followed by an allele descriptor and the frequency of that allele. There is a separate record for each allele.

Files:

Input: A Data File having records containing genotypic or phenotypic data, and an associated Code File.

Optionally, a Phenotype-Genotype Map File

Output: **genefreq.tab** contains summary tabulations of gene and genotype (or phenotype) frequencies for each locus selected.

genefreq.err contains messages about incorrectly formulated symbols.

Algorithm: R. Polich, L. Freeman-Shade

GENOMAP

Program GENOMAP is used to create and/or edit Genotype-Phenotype Map Tables, required to specify relationships between phenotypes and genotypes of (a) loci that are not autosomal codominant, or (b) autosomal codominant loci whose nomenclature does not conform to the PEDSYS conventions given in Appendix C, Rules for Genotype and Phenotype Nomenclature. Tables for one of more loci are kept in a Genotype-Phenotype Map File, generated by the program.

Introductory Display:

```
PEDSYS program GENOMAP - Copyright 1994, 1999 SFBR

GENOMAP is used to create and/or edit a Genotype-Phenotype Map File.

Enter - [C] to CREATE a new Genotype-Phenotype Map File
        E to EDIT a previously created Phenotype-Genotype Map File
        Q to QUIT program
>C
```

1. Creating a new Genotype-Phenotype Map File

Creation of a new Genotype-Phenotype Map File (choice **C** above) results in the following display in which mode of inheritance is specified for generation of a Genotype-Phenotype Map Table:

```
 1. Autosomal codominant         (AUCD)
 2. Autosomal dominant/recessive (AUDR)
 3. X-linked codominant          (XLCD)
 4. X-linked dominant/recessive  (XLDR)
 5. Y-linked                     (YL)

Enter number corresponding to mode of inheritance.
>1
```

If autosomal codominant mode (choice **1**) is selected, the following display appears:

```
   Enter - [F] to read Genotype-Phenotype Map automatically from locus file
           K  to enter Genotype-Phenotype Map Table manually from keyboard
>f
```

Information for a Genotype-Phenotype Map Table can be assembled from a file containing phenotypic or genotypic data, and/or from keyboard entry.

Choice F - Genotype-Phenotype Map Data Read from File

If choice **F**, selection of a file containing an autosomal codominant locus is selected, a File Selection display sequence appears from which a locus file is chosen:

```
Select file containing locus

  1.  Unlinked File
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter number, file name, [R] to reselect mode of inheritance, "Q" to quit.
>2
```

Here, the Example Directory has been selected,

```
Linked Files:

    1.  Example Master File
    2.  Markers + Quant. P'types
    3.  Blood Sample Inventory
    4.  Hemoglobin Levels

Specify file containing genotypic/phenotypic data
>2
```

the Markers + Quant. P'types file,

```
   1 Ego ID                 3 APOB Pvu2a Genotype     5 APOAI Level
   2 ADA  Phenotype         4 Tot. Ser. Cholestero
--------------------------------------------------------------------------------
  GENOMAP     Select locus                    File - QUANTGEN.XMP
--------------------------------------------------------------------------------
Enter number above corresponding to locus, or enter RETURN to select another
locus file.
>2
```

and the ADA Phenotype chosen:

```
 * 1 ADA  Phenotype
--------------------------------------------------------------------------------
  GENOMAP     Select locus                    File - QUANTGEN.XMP
--------------------------------------------------------------------------------

Is this correct?  [Y]/N
>Y
```

If this is not correct (entry of **N**), the following display appears, giving options to delete (**D**), replace (**R**), or add a table entry:

```
   1 ADA  Phenotype
--------------------------------------------------------------------------------
  GENOMAP     Source of alleles              File - QUANTGEN.XMP
--------------------------------------------------------------------------------

Enter - D  to DELETE an item
        R  to REPLACE an item
        A  to ADD an item
       [C] to CONTINUE to next step
>
```

If the table entry is correct, the following display appears:

```
Enter - [A] to create table automatically from genotypes/phenotypes in FILE
         M  to enter genotypes/phenotypes MANUALLY for this locus
>
```

At this point, locus mnemonics have been entered into a new table, and the next step is to add phenotypes and their corresponding genotypes:

Option A - Automatic creation of map table (locus data read from file)

If option **A** is chosen, the following confirmation display appears, showing correspondences between phenotypes found in the file and their genotypes deduced by program GENOMAP:

```
ADA PHENO TYPE Genotype-Phenotype Map (AUCD)

   Phenotype     :Genotype

   1. A          : A/A
   2. AB         : A/B
   3. AC         : A/C
   4. B          : B/B
   5. BC         : B/C
   6. C          : C/C

Enter [C] to continue if this is correct, "E" to edit, "R" to restart with
new locus.
>C
<<Note that you can't quit if you select 'R' and then change your mind>>
```

Here, entry of **E** switches to Edit Mode (see below); **R** aborts the current entry and starts the beginning sequence for entry of a new locus; **C** produces the following display:

```
 Genotype-Phenotype Map Table

   1. ADA PHENO TYPE

Enter "V" to view locus, "E" to edit locus, "A" to add locus, "D" to delete
locus, [C] to continue, generating and saving output file.
>
```

Here, entry of **V** returns to the confirmation view of correspondences between phenotypes and genotypes shown above; **E** switches to Edit Mode, **A** starts the beginning sequence for entry of a new locus; **D** is followed by a request for the locus to be deleted, **C** begins execution of the program. See Section 2 below (**Editing a previously created Genotype-Phenotype Map File**) for more details on Edit commands

Option M - Manual creation of map table (locus data read from file)

If option **M** is chosen, the following display appears, in which phenotype symbols are entered:

```
Enter symbols for PHENOTYPES in any order. Enter RETURN alone when done.
>A
>B
>C
>AB
>AC
>BC
>

Are these correct?  [Y]/N
>y
```

Next, correspondences between phenotypes entered above and the possible genotypic combinations derived from them must be established, also manually. Note that locus mnemonics are displayed since data for the ADA locus has been read from a file:

```
Locus: ADA PHENO TYPE,  Mode of inheritance: AUCD

 Phenotypes: ================================================  Page  1/ 1

  1 A             3 C             5 AC
  2 B             4 AB            6 BC

 Genotypes: =================================================  Page  1/ 1

  1 A/A = A                 3 A/B = AB              5 B/C =
  2 A/C = AC                4 C/C =                 6 B/B =

Enter number corresponding to GENOTYPE to be assigned to a phenotype.  Enter
"AP" to add, "DP" to delete a phenotype; "AA" to add, "DA" to delete, or "RA"
to replace allele symbol. Enter "M" to change locus mnemonics, "I" to change
mode of inheritance, [C] to continue.
>4

Enter number of PHENOTYPE for A/C
>3
```

In the display above, correspondence between the first three genotypes listed (**A/A**, **A/C** and **A/B**) and their respective phenotypes (**A**, **AC** and **AB**) have been entered, while matching of genotype 4 (**C/C**) with phenotype 3 (**C**) has been specified. This is essentially the same display as is used for Edit Mode (shown below).

When this process is finished the program reverts to the same sequence of operations as shown above for option A, automatic creation of the map table (confirmation display, etc.).

Choice K - Genotype-Phenotype Map Data Entered from Keyboard (no data read from file)

When choice **K** is selected, all data must be entered manually. First to be entered are locus mnemonics. Although not illustrated here, autosomal dominant/recessive (AUDR) mode of inheritance has been selected for the following example).

```
Enter one or more mnemonics for the locus.  NOTE: These should be identical to
mnemonics found in the Code File of the corresponding Data File.  Enter RETURN
alone to continue to next step.

>ABO   <  Enter mnemonic 1
>PHENO <  Enter mnemonic 2
>TYPE  <  Enter mnemonic 3
>      <  Enter mnemonic 4
```

Here, mnemonics for the ABO phenotype have been entered. Next, allele symbols are entered, followed by phenotype symbols. Note that all symbols are case-sensitive, and that alleles and phenotypes need not have the same case:

```
Enter symbols for ALLELES in any order. Enter RETURN alone when done.
>a
>b
>o
>

Are these correct?  [Y]/N
>y

Enter symbols for PHENOTYPES in any order. Enter RETURN alone when done.
>A
>B
>AB
>O
>

Are these correct?  [Y]/N
y
```

Next, correspondences between phenotypes, and the possible genotypic combinations derived from the alleles entered above must be established in the same manner as was illustrated above for the case in which data were read from a locus file:

```
Locus: ABO PHENO TYPE,  Mode of inheritance: AUDR

 Phenotypes: ================================================  Page  1/ 1

  1 A            2 B            3 AB           4 O

 Genotypes: =================================================  Page  1/ 1

  1 a/a = A                3 a/o =                  5 b/o =
  2 a/b = AB               4 b/b =                  6 o/o =

Enter number corresponding to GENOTYPE to be assigned to a phenotype.  Enter
"AP" to add, "DP" to delete a phenotype; "AA" to add, "DA" to delete, or "RA"
to replace allele symbol. Enter "M" to change locus mnemonics, "I" to change
mode of inheritance, [C] to continue.
>3

Enter number of PHENOTYPE for a/o
>1
```

In the display above, correspondence between the first two genotypes listed (**a/a** and **a/b**) and their respective phenotypes (**A**, **AB**) have been entered, while matching of genotype 3 (**a/o**) with phenotype 1 (**A**) has been specified. When this process is finished the program reverts to the same sequence of operations as shown above for other map table entry options (confirmation display, etc.).

2. Editing a previously created Genotype-Phenotype Map File

Entry of choice **E** in the introductory screen results in the following display:

```
Enter name of Genotype-Phenotype Map File to edit.
>genomap.tab
```

When the name of the Genotype-Phenotype Map File has been entered, the following display appears:

```
genomap.tab Genotype-Phenotype Map File

   1. ADA PHENO TYPE
   2. ABO PHENO TYPE
   3. G6PD PHENO TYPE
   4. DYS391 PHENO TYPE

Enter "V" to view locus, "E" to edit locus, "A" to add locus, "D" to delete
locus, [C] to continue, generating and saving output file.
>
```

Commands:

V Entry of **V** produces the confirmation display of correspondences between phenotypes and genotypes. First, the number of map table to be viewed must be entered:

```
Enter number of locus to view
>1
```

Here, the ADA locus (choice **1**) has been selected, which produces the following display:

```
       ADA PHENO TYPE Genotype-Phenotype Map (AUCD)

          Phenotype     :Genotype

          1. A          : A/A
          2. AB         : A/B
          3. AC         : A/C
          4. B          : B/B
          5. BC         : B/C
          6. C          : C/C

       Enter [C] to continue
```

Entry of RETURN returns to the initial Edit Menu.

E Entry of **E** switches to Edit Mode, first requiring entry of the number of the map table to be edited:

```
Enter number of locus to edit
>1
```

When the map table has been selected, the Edit Menu Display appears:

```
Locus: ADA PHENO TYPE,  Mode of inheritance: AUCD

 Phenotypes: ================================================  Page  1/ 1

  1 A             3 AC            5 BC
  2 AB            4 B             6 C

 Genotypes: =================================================  Page  1/ 1

  1 A/A = A                 3 A/C = AC              5 B/C = BC
  2 A/B = AB                4 B/B = B               6 C/C = C

Enter number corresponding to GENOTYPE to be assigned to a phenotype.  Enter
"AP" to add, "DP" to delete a phenotype; "AA" to add, "DA" to delete, or "RA"
to replace allele symbol. Enter "M" to change locus mnemonics, "I" to change
mode of inheritance, [C] to continue.
>
```

Edit Options

Genotype - phenotype assignment change - Assignment of a genotype to a particular phenotype may be changed by entering a number corresponding to the genotype, which produces a request for the number of s corresponding phenotype:

```
Enter number corresponding to GENOTYPE to be assigned to a phenotype.  Enter
"AP" to add, "DP" to delete a phenotype; "AA" to add, "DA" to delete, or "RA"
to replace allele symbol. Enter "M" to change locus mnemonics, "I" to change
mode of inheritance, [C] to continue.
>2

Enter number of PHENOTYPE for A/C
>1
```

Here, genotype **2** (symbols **A/B**) will be reassigned phenotype **1** (symbol **A**). Note that such a change would require substantial further alterations to the table if it were to made permanent (see option **AA** below).

AP - When the **AP** (Add phenotype) option is entered in the Edit Menu, the following display appears, requesting entry of a symbol for the new phenotype:

```
Enter new phenotype symbol
>D
```

Here, symbol **D** has been entered. This symbol is added (numbered **7**) to the list of phenotypes shown in the top half of the Edit Menu:

```
Locus: ADA PHENO TYPE,  Mode of inheritance: AUCD

 Phenotypes: ================================================  Page  1/ 1

  1 A             3 AC            5 BC            7 D
  2 AB            4 B             6 C
```

If this phenotype is to remain in the table, it must be associated with one or more appropriate genotypes, which may entail adding allele symbols (see option **AA** below).

DP - When the **DP** (Delete phenotype) option is entered, the following display appears, requesting entry of a symbol to be deleted from the list:

```
Enter phenotype to be deleted.
>D
```

This option would return the list of phenotypes to the six shown originally above.

AA - When the **AA** (Add allele) option is entered, the following display appears, requesting entry of a symbol for the new allele:

```
Enter new allele symbol
>D
```

Here, symbol **D** has been entered. This symbol is incorporated into a series of new genotype combinations (numbered **7 - 10**) in the list of genotypes shown in the bottom half of the Edit Menu:

```
 Genotypes: ================================================  Page  1/ 1

  1 A/A = A               5 B/C = BC              9 A/D =
  2 A/B = AB              6 C/C = C              10 B/D =
  3 A/C = AC              7 D/D =
  4 B/B = B               8 C/D =
```

Note that the new genotypes have not been assigned to their corresponding phenotypes, which must be done with the first Edit option described above.

DA - When the **DA** (Delete allele) option is entered, the following display appears, requesting entry of a symbol to be deleted from the list:

```
Enter allele to be deleted
>C
```

This would reduce the list of genotypes shown in the bottom half of the Edit Menu :

```
Genotypes: ================================================  Page  1/ 1

 1 A/A = A                 2 A/B = AB              3 B/B = B
```

Here again, such a change would require further alterations to the table if it were to made permanent (see option **DP** above).

RA - When the **RA** (Replace allele) option is entered, the following display appears, requesting entry first of a symbol to be replaced in the list, and then its replacement:

```
Enter allele symbol to be replaced
>A

Enter new allele symbol
>a
```

This would result in substituting a lower case **a** for all occurrences of upper case **A** in the list of genotypes shown in the bottom half of the Edit Menu :

```
 Genotypes: ================================================  Page  1/ 1

  1 a/a = A               3 C/a = AC              5 B/C = BC
  2 B/a = AB              4 B/B = B               6 C/C = C
```

Care must be taken that such replacements actually reflect the form and content of symbols in the locus file to which the Genotype-Phenotype Map Table applies.

M - Entry of **M** (Change locus mnemonics) produces the following display:

```
Enter one or more mnemonics for the locus. NOTE: These should be identical
to mnemonics found in the Code File of the corresponding Data File. Enter
RETURN alone to continue to next step.

>ADA    <  Enter mnemonic 1
>P'TYPE<  Enter mnemonic 2
>       <  Enter mnemonic 3
```

Here, the mnemonics **ADA PHENO TYPE** are changed to **ADA P'TYPE**. This change will be reflected in the first line of the map table for the locus in question.

I - The **I** option (Change mode of inheritance) produces the following display:

```
Locus: ADA P'TYPE,  Mode of inheritance: AUCD

 1. Autosomal codominant         (AUCD)
 2. Autosomal dominant/recessive (AUDR)
 3. X-linked codominant          (XLCD)
 4. X-linked dominant/recessive  (XLDR)
 5. Y-linked                     (YL)

Enter number corresponding to mode of inheritance.
>
```

This change will also be reflected in the first line of the map table for the relevant locus

Notes:

- Any printable ASCII characters may be used for phenotype or genotype symbols except for a blank space (ASCII Code 32), the open or closed parenthesis (Codes 40 and 41), and the slash (Code 47).
- Maximum length of allele symbols is 5 characters; maximum genotype length is 10 characters.
- Minimum width of genotype (and phenotype) fields is always twice the width of the longest character string representing an allele.
- Genotypes must be represented by pairs of symbols representing both alleles.
- There must be at least one genotype entry for each phenotype. A given genotype may be mapped to only one phenotype. None of the $na(na + 1)/2$ autosomal genotypes, or $na(n +3)/2$ X-linked genotypes (where *na* is the number of alleles) may remain unmapped.
- Many of GENOMAP's editing features frequently require that action be taken after making initial changes. For example, adding or deleting a phenotype symbol typically requires changes in genotype assignment, and possibly deletion or addition of new genotypes. Changing between X-linked and autosomal mode of inheritance has significant effects because GENOMAP assigns a symbol denoting the Y chromosome to the hemizygous type in place of the second allele symbol always present in autosomal representation (**a/Y** in contrast to **a/a**, for example). Such changes are likely to require substantial modification of genotype assignments.
- Mode of inheritance is coded mnemonically as follows:

AUCD	Autosomal codominant
AUDR	Autosomal dominant/recessive
XLCD	X-linked codominant
XLDR	X-linked dominant/recessive
YL	Y-linked

- A Genotype-Phenotype Map need not be created for autosomal codominant loci that adhere to the conventions given in Appendix C, Rules for Genotype and Phenotype Nomenclature. Nonetheless, is it frequently helpful to generate a map for autosomal codominant loci using program GENOMAP, because PEDSYS programs GENEFREQ, GENCHECK, INFER and TRANSLAT will otherwise be forced to read through the entire Data File twice for each locus (once to tally phenotypes, and

again to do the analysis or transformation). Presence of a map file eliminates the need for the tallying step and can significantly speed execution.

- To be accessible to PEDSYS programs that use it, output file **genomap.tab** should be renamed **GENOMAP.<EXT>**, and moved to the appropriate Data Directory.

Output:

```
ADA     PHENO  TYPE            3   6   6 AUCD
C     A     B

               |A|A| |B| |
              A|B|C|B|C|C|
A     A      |+| | | | | |
A     B      | |+| | | | |
A     C      | | |+| | | |
B     B      | | | |+| | |
B     C      | | | | |+| |
C     C      | | | | | |+|
-------------------------------------------------------------------------------

ABO     PHENO  TYPE            3   4   6 AUDR
A     B     O

               | | |A|
              A|B|O|B|
A     A      |+| | | |
A     B      | | | |+|
A     O      |+| | | |
B     B      | |+| | |
B     O      | |+| | |
O     O      | | |+| |
-------------------------------------------------------------------------------

G6PD    PHENO  TYPE            2   3   5 XLCD
B     A

               |A| |
              A|B|B|
A     A      |+| | |
A     Y      |+| | |
A     B      | |+| |
B     B      | | |+|
B     Y      | | |+|
-------------------------------------------------------------------------------

DYS391  PHENO  TYPE             2   2   2 YL
B     A

              A|B|
A     X      |+| |
B     X      | |+|
```

Record 1 contains 1 - 4 locus mnemonics (here, ADA), followed by integers denoting the number of alleles (3), phenotypes (6), and genotypic combinations (6), respectively, and the symbol denoting mode of inheritance (AUCD).

Record 2 contains symbols for each of the alleles .

Record 3 is blank.

Records 4 - 5 contain phenotype labels.

Records 6 - 11 contain genotype symbols and markers (+) designating phenotypes they express.

This pattern is repeated for the ABO, G6PD and Y-linked loci that follow. A more detailed description of formats of the records in this file can be found in Appendix A.

Files:

Input: Console and/or a Data File containing phenotypic or genotypic data, and optionally, a previously constructed Genotype-Phenotype Map File.

Output: File **genomap.tab**.

Program limits:

Tables for up to 50 loci, each with up to 13 alleles may be constructed.

INDEX

The chief function of program INDEX is to create a PEDSYS Master File by assigning a Sequential ID number (pointer) to selected individuals in a pedigree file, and arranging data items in a predetermined order on each record. Other functions are to compute numbers of individuals' offspring, and to create a Master Pointer File (see Appendix A. File Types).

Introductory Display:

```
PEDSYS program INDEX - Copyright 1992, 1999 SFBR

INDEX assigns a Sequential ID number (file pointer) to individuals in a
file.  Individuals may be indexed as follows:

    [1]  EGO, FA and MO (adding pointers to offspring, sibs, etc.)*
     2.  Same as above, plus another individual (such as a mate)*
     3.  EGO, FA and MO (without adding pointers to offspring, etc.)
     4.  Any one individual
                                              * Produces Master File

Enter number above corresponding to choice, "H" for HELP, "Q" to QUIT.
>
```

Once the indexing format has been selected, a pedigree file is chosen from the following display:

```
 1.  Unlinked Pedigree File
 2.  Example Master File

Enter the number or name of the file to index.
>1
```

Each of the four choices for indexing format shown in the Introductory Display produces a slightly different result:

Choice 1. (Index of EGO, FA and MO, producing Master File)

Choice 1 indexes EGO, FA and MO, adding Sequential IDs (pointers) of first offspring; next oldest paternal, maternal and full sibs; number of offspring; paternal, maternal and full sibship sizes; Pedigree Identification Number; and Generation Number, to produce a standard PEDSYS Master File. When this option is selected, a modified File List appears from which a pedigree file may be chosen. No Item List is displayed. INDEX searches the Code File of the pedigree file selected, and if the required items are found (see **Notes** below), the following queries appear in sequence:

Query 1

```
Create record(s) for missing known parent(s)? [Y]/N
>
```

Many analysis programs require that the pedigree file contain a record for each PID that appears in the file. Sometimes this requirement is not met when pedigrees imported from other databases have been assembled from birth registry data in which the identity of parents is known, but parental birth records are not available, and thus are not entered into the file. If Ego's record contains one or more parental PIDs for which individual records are not present in the pedigree file, answering **y** to the first query above instructs INDEX to create a record for each missing parent, and assign a pointer appropriately to Ego's record.

Query 2

```
Create record for unknown parent when only one is known? [Y]/N
>
```

A similar problem can occur when only one parent is known, that is, when one of the parental PIDs is blank. Answering **y** to the second query above instructs INDEX to (a) create a record for the unknown parent, and (b) place a PID for that parent in the appropriate place in Ego's record. Normally, Permanent Identification Numbers are kept as Data Type **C**, in which case PIDs of newly created individuals start with the character **?**, followed by a sequential number equal to one greater than the number of previously created records. If PIDs are kept as Data Type **I** *(not recommended)*, newly created integer PIDs are incremented in the range <i> through <n>, where <i> is one greater than the largest integer representing a PID in the file. In some cases fields width of newly created PIDs may exceed the original width specified in the input Code File. When this occurs, INDEX halts execution with an instruction to increase PID field widths appropriately. Note that records for fictitious offspring also may be created when needed, as described below.

Query 3

```
Create an offspring record for mated pairs without progeny? [Y]/N
>
```

Some genetic analysis programs will not process individuals that have mated into a pedigree, but do not have offspring listed in the pedigree file. INDEX will create a fictitious offspring for pairs without progeny if the following conditions are met:

- PIDs for EGO, FA and MO must be included in the input pedigree file. SEX code is required.
- Records must include an item containing the Permanent ID of a mate. The input Code File must contain an Item Descriptor Record with the code **MATE** (or **SPOUSE**) as the first and only mnemonic word.
- Any individual listed as a mate must also appear as Ego in the pedigree file.

Entry of **y** results in creation of a single fictitious offspring for each pair. PIDs of newly created offspring are formed as described above for missing parents.

Query 4

Some genetic analysis programs interpret **0** as a legitimate pedigree number and thus misinterpret assignment of unrelated single individuals to this category. This can be avoided by choosing option **2** below, which assigns such individuals their own separate pedigree numbers:

```
Assign individuals that have neither parents nor offspring

 1. All to the same pedigree (Pedigree 0)
 2  Each to a separate pedigree.

Enter number corresponding to choice
>
```

Query 5

```
Create a Master Pointer file? [Y]/N
>
```

The Master Pointer File is an important component of the PEDSYS file system, and must be created if a Master File is being created for the first time. In fact, however, it is usually a good idea to create a new Master Pointer File any time INDEX is used to create a Master File.

Choice 2. (Master File with additional indexed individual)

Choice 2 also creates a standard PEDSYS Master File, but adds an index for another individual appearing on Ego's record. This feature may be used to create Sequential IDs of mates (usually for monogamous species or single-pair caging). The command sequence is the same as for Choice 1, except that the following Item List Display appears after file selection (a pedigree file **examped** has been created for this example):

```
 * 1 EGO Permanent ID      * 4 Dam's Permanent ID      7 Exit Code
   2 Birth Date (YYYYMMDD    5 Sex (M, F or U)         8 Mate's Permanent ID
 * 3 Sire's Permanent ID     6 Exit Date (YYYYMMDD)
--------------------------------------------------------------------------------
  INDEX                                File - examped
--------------------------------------------------------------------------------
Individuals marked with an asterisk will be indexed and the first eleven items
above replaced. To index an individual other than EGO, FA or MO, enter the
corresponding number above. Enter [C] to continue to next step.
>8
```

EGO, FA and MO are pre-selected, and another individual may be indexed, in which case a new Sequential ID will be inserted just after the individual selected. Here, Mate's PID has been selected for indexing. Subsequent queries and commands are the same as in Choice 1.

Choice 3. (Index of EGO, FA and MO without producing Master File)

Choice 3 automatically indexes EGO, FA and MO, but does not generate derived items 4 - 11, Pedigree Identification Numbers, or Generation Numbers, normally included in Master File records, and a Master Pointer File is not generated. When this mode is selected, a modified File List appears from which a pedigree file may be chosen. When an appropriate pedigree file and its Code File have been entered, the following screen is shown:

```
 * 1 EGO SEQUENTIAL ID       6 Sire's Permanent ID    11 Mate's Permanent ID
 * 2 SIRE'S SEQ              7 Dam's Permanent ID     12 PEDIGREE NUMBER
 * 3 DAM'S SEQ               8 Sex (M, F or U)        13 GENERATION NUMBER
   4 EGO Permanent ID        9 Exit Date (YYYYMMDD)
   5 Birth Date (YYYYMMDD   10 Exit Code
--------------------------------------------------------------------------------
  INDEX       Items to be included            File - examped
--------------------------------------------------------------------------------
Enter numbers corresponding to items shown above in the order they are to
be included in indexed record. Enter "A" to list all items shown above,
[C] to continue to next step.
>
```

EGO, FA and MO are pre-selected for indexing, and only their Sequential IDs will be included in the output file unless other items are selected at this point. Subsequent queries and commands are the same as in Choice 1.

If the input file is in Master File format, Item List Display contains the following warning:

```
 * 1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
 * 2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ   16 Sex (M, F or U)
 * 3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE   12 EGO Permanent ID       19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE   13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  INDEX       Items to be included            File - MASTER.XMP
--------------------------------------------------------------------------------
WARNING.  Input is in Master File format. EGO, FA and MO will be re-indexed,
but sibs and offspring will retain their original Sequential IDs. Enter
numbers corresponding to items shown above in the order they are to be
included in indexed record. Enter "A" to list all items shown above, [C] to
continue to next step.
>
```

This warning indicates the potential danger of a mismatch between Sequential IDs of EGO, FA and MO (which will be re-indexed) and those of sibs and offspring (which will not). Such a mismatch could occur if changes have been made to the input file after it was originally indexed. Under these circumstances, it is usually best (a) to eliminate the original Items 4 - 11 for inclusion in output records, or (b) to produce a complete new Master File by restarting the program and selecting Choice 1 or 2.

Indexing EGO, FA, and MO without producing a Master File requires an input file containing minimally the IDs of ID/EGO, FA/SIRE, and MO/DAM. These may be either Permanent or Sequential IDs. In either case, the sequence of numbers need not be in order and may include gaps. Mnemonic codes for required entries must be used exactly as shown above for production of Master Files.

Choice 4. (Index of any one individual without producing Master File)

Choice 4 automatically indexes a single individual, but does not generate derived items 4 - 11, Pedigree Identification Numbers, or Generation Numbers, normally included in Master File records, and a Master Pointer File is not generated. A Standard Item List (with no individuals pre-selected for indexing) is displayed when this choice is selected:

```
   1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ   16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE   12 EGO Permanent ID       19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE   13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 GENERATION NUMBER

--------------------------------------------------------------------------
INDEX                                FILE - MASTER.XMP
--------------------------------------------------------------------------
Enter number corresponding to the individual to be indexed.
>19
```

Here, we have chosen to add an index to Ego's mate. Indexing a single individual requires an Input File containing the Permanent ID of Ego, and of the individual to be indexed (if other than Ego).

Notes on Input Record Format

If the input Code File contains parental mnemonics **FA** and **MO**, the population is assumed to be human. If the Code File contains mnemonics **SIRE** and **DAM**, the population is assumed to be nonhuman.

- Choice 1 -- INDEX requires an input Data File with records containing items specified by the table below to index of EGO, FA and MO and produce Master File. Order of input Code File entries is not fixed, and will depend on the order of the data in the Input File. Required entries may be interspersed with entries describing other items in data records. Mnemonic codes for required entries must be used exactly as shown (see description of the Standard Mnemonics File in Appendix A). INDEX will terminate with an error message if any of the required items are missing from the input Code File. Exit Date and Exit Code are optional items for all PEDSYS programs, but may be required for some demographic analyses (see Notes on Master File Record Format in Appendix A).

Length	Item name	Mnemonic	Type***
3 - 9	Ego Permanent ID	EGO (or ID)	C
3 - 9	Father's Permanent ID	FA (or SIRE)	C
3 - 9	Mother's Permanent ID	MO (or DAM)	C
1	Sex	SEX	C
3 - 8 *	Birth Date (Yr,Mo,Da)	BIR (or BIRTH)	D
3 - 8 *	Exit Date (Yr,Mo,Da) **	EXIT	D
1 - 3	Exit Code **	EXCODE	C

... Code File may contain additional items...

* PEDSYS date format, eight-digits preferred
** Optional
*** Character Data Type preferred

- Choice 2 -- The same input items as given for Choice 1, plus an item containing a permanent ID to be indexed are required to produce a Master File with additional indexed individual.

- Choice 3 -- Only the first three items in the table are required to index EGO, FA and MO without producing a Master File

- Choice 4 -- Only an item containing a permanent ID to be indexed is required in this case.

Notes on Output Record Format

- Choice 1 -- INDEX produces Master File records formatted as shown in the table below. The first 12 items of the record are fixed in position, while the remainder follow in an arbitrary order, largely dependent on the order of items specified by the input Code File. Items 1-11, 16, 19 and 20 are invariant in length, while others may vary within limits specified in the table. Items 1 - 11 as well as Pedigree Identification Number and Generation Number (Items 19 and 20 in the table) are family data derived during execution of the program. A valid Master File can be created without specifying an Exit Date, but this practice should be avoided if possible (see Notes on Master File Record Format in Appendix A). Pedigree Identification Number and Generation Number are added automatically. Output records will also include additional items present in the input file. Twenty standard items of a Master File are formatted as shown below.

Item No.	Length	Item name	Mnemonic	Type
1	5	Ego Sequential ID	SEQ	I
2	5	Father's SEQ	FSEQ (or SSEQ)	I
3	5	Mother's SEQ	MSEQ (or DSEQ)	I
4	5	First Offspring SEQ	KID1	I
5	5	Next Paternal Sib SEQ	PSIB	I
6	5	Next Maternal Sib SEQ	MSIB	I
7	5	Next Full Sib SEQ	FSIB	I
8	3	Number of Offspring	NKID	I
9	3	Paternal Sibship Size	PSIBSZ	I
10	3	Maternal Sibship Size	MSIBSZ	I
11	3	Full Sibship Size	FSIBSZ	I
12	3 - 9	Ego Permanent ID	EGO (or ID)	C
(13)	3 - 8 *	Birth Date (Yr,Mo,Da)	BIRTH (or BIR)	D
(14)	3 - 9	Father's Perm. ID	FA (or SIRE)	C
(15)	3 - 9	Mother's Perm. ID	MO (or DAM)	C
(16)	1	Sex (M,F,U or 1,2,3)	SEX	C
(17)	3 - 8 *	Exit Date (Yr,Mo,Da) **	EXIT	D
(18)	1 - 3	Exit Code **	EXCODE	C
(19)	5	Pedigree ID Number	PEDNO	I
(20)	5	Generation Number	GEN	I

... Code File may contain additional items...

( ) position not fixed * Eight-digit dates preferred ** Optional

- **Choice 2** -- Master File records containing the index of an additional individual include all items shown in table above, with new field of Sequential IDs appearing as an item immediately following the PID of the individual selected for indexing, as well as additional items included in the input file.

- **Choice 3** -- Minimally, only Sequential IDs of EGO, FA and MO, are included on output records if no items are selected in this mode. Additional items are included only if selected from input records.

- **Choice 4** -- In this mode, all items present on input records are included on output records, with the new field of Sequential IDs appearing as an item immediately following the PID of the individual selected for indexing.

Assignment of Pedigree ID and Generation Numbers

INDEX identifies genealogical links in a pedigree file and assigns numbers to each individual corresponding to the pedigree to which he or she belongs, as well as the generational position of that individual in the pedigree. Singletons may be assigned either to Pedigree Number 0 or to their own individual pedigree; Founders are assigned to Generation 0, with numbers increasing through more recent generations. This function is automatic, except for Choices 3 and 4, where no assignment of these variables is done. INDEX maintains the record position of items containing Pedigree ID and Generation Numbers if mnemonics PEDNO and GEN are found in the input Code File. Otherwise the newly derived variables are placed at the end of the output record.

Order of Records in the Output File

INDEX automatically sorts output records in the following order: Pedigree ID Number, Generation Number, Father (Sire), Mother (Dam), Birth Date (if present), and EGO PID. This groups members of the same pedigree in adjacent positions, and maintains the correct parent-offspring order (parents always precede their offspring). As a consequence of this sorting sequence, records in **index.out** may not be in the same order as those in the input file. Also, if the option is chosen to create records for offspring or missing parents, there may be more records in **index.out** than in the input file.

Creating Records for Unknown and Missing Parents

PEDSYS convention makes a distinction between missing and unknown parents.

• A *missing* parent is one for which no record exists in the pedigree file, but whose PID appears as one of Ego's parents on one or more records in the file. References to missing parents on Ego's record generate a warning message written to the **index.err** file.

WARNING: When the option to create records for missing parents is chosen, INDEX issues a warning if a parental PID is listed as **0** (integer zero). This is done to help avoid problems with imported foreign pedigree data where missing parents may sometimes be designated by zeros. PEDSYS designates missing data as a blank field, and interprets **0** as a legitimate Permanent ID (in contrast to the Sequential ID). In this case, if INDEX cannot find a record with Ego PID listed as "**0**", it will create one. If more than one record contains zero PIDs for one same-sexed parent, the process may create unwanted half-sibships.

• An *unknown* parent is one for which no record exists in the pedigree file, and whose PID does not appear as FA or MO on any record in the file. Unknown parents are designated by blank parental fields in PEDSYS pedigree files.

WARNING: Creation of a record for one unknown parent requires that a record for the other, known parent be present in the file.

Creating Records for Fictitious Offspring

Only one fictitious offspring is created for a given pair; no other checks for reciprocal mate references are done. Records created are then indexed with the rest of the file.

A note on the use of integers as Permanent IDs

• PEDSYS Permanent IDs are normally in character format (Data Type C) and left-justified. This convention may be problematic if PIDs consist solely of integers that must sort in numerical order, because the collating sequence for characters is different from that for integers.

It is strongly recommended that PIDs be kept as Data Type C, and contain non-integer characters (preferably in the leading position) to avoid confusion with Sequential IDs.

Error Messages and Displays:

Because INDEX plays such an important part in the organization of pedigree databases, considerable effort has gone into preventing errors from being incorporated into the data. Protection from permanent damage is provided at three levels:

a. Errors that affect single records, or that lead to an incomplete pedigree structure, are displayed on the console screen during program execution, and are recorded in file **index.err**. The program produces output files and otherwise terminates normally.

b. Most errors that lead to incorrect pedigree structure are displayed on the console screen during program execution, and are recorded in file **index.err**. In addition, these errors cause the program to suppress production of output data files, and to terminate with a FATAL ERROR message.

c. INDEX checks human populations (indicated by parental mnemonics **FA** and **MO**) for nuclear family consanguinity (father-daughter, mother-son, full- and half-sib matings). Occurrence of such matings results in a warning in **index.err**. This check is done only when pointers to offspring and sibs are generated (initial Choice 1 and Choice 2 shown in the Introductory Display). Original input files remain unchanged, making it possible to return to earlier versions of the data.

Output:

Choices 1 or 2 (Master File produced):

If the input pedigree file is not already in Master File format, individuals selected will be indexed, and the first eleven items of a Master File record will be created. If the input pedigree file is in Master File format, individuals selected will be re-indexed, and the first eleven items of the Master File, plus PEDNO and GEN will be replaced.

The newly generated index item for the additional individual selected in Choice 2 will immediately follow that individual's Permanent ID in the output record. All other items will be placed in the same order as they appeared in the input record.

Choice 3 (Master File not produced):

Newly generated indices (for ID/EGO, SIRE/FA and DAM/MO) will appear as the first three items in the output record. Additional items will be placed in the order they are selected.

Choice 4 (single individual):

The newly generated index item for the individual selected will immediately follow that individual's Permanent ID in the output record. All other items will be placed in the same order as they appeared in the input record.

Files:

Input: 1. For Choice 1 - 3, any pedigree file (having records containing a unique identifier for an individual and his or her parents, sex and birth date). For Choice 4, any file containing a unique identifier for each record.

2. An associated Code File.

3. The Standard Mnemonics File (CODE.STD).

Output: **index.out** contains all records of the input file plus additional records created to represent missing or unknown parents and hypothetical offspring of mated pairs without progeny, with items (indices, etc.) added as described above.

index.cde defines the record structure of **index.out**.

index.err contains information on individuals that fail to pass checks of internal consistency at the individual and family level (duplicate Ego PIDs, wrong sex of parent, offspring birth date listed as earlier than that of parent, etc.).

index.mpt contains records in Master Pointer File format.

Program limits:

A maximum of four individuals (ID/EGO, SIRE/FA, DAM/MO plus one additional individual) may be indexed at one time. Input files must be formatted as described above. INDEX will not run with a pedigree file of more than 8,000 individuals on the IBM-PC, or 60,000 individuals on the Macintosh and Unix machines.

INFER

Program INFER infers unknown phenotypes and genotypes from phenotypes (or genotypes) of relatives. The program examines each offspring-father-mother triplet and when possible, adds missing alleles and genotypes according to the Mendelian laws of transmission. An error is reported when an inconsistency between Ego and a parent is found. When all triplets have been examined, the program starts through the pedigree again, making assignments on the basis of genotypes inferred in the previous iteration. The entire pedigree file is re-examined repeatedly until no more assignments can be made by examining triplets. The file is again re-examined, this time for inconsistencies revealed by including the grand-parental generation, and by checking the distribution of alleles within entire sibships.

Introductory Display:

```
PEDSYS program INFER - Copyright 1992, 1999 SFBR

INFER infers unknown genotypes from phenotypes (or genotypes) of relatives.

  1.  Unlinked File
  2.  merge.out
  3.  EXAMPLE
  4.    LOCUS Sub-directory

Enter number of the file or directory to use.
>3
```

Loci may be selected for analysis in several ways:

1. An unlinked file containing genotypes or phenotypes for one or more loci

It is often convenient to organize multiple genotypes and/or phenotypes on the same record. An example of this organization is given by file **merge.out** (choice **2** above) in which genotypes for three loci have been assembled on records for each individual (display from program BROWSE):

```
 merge.out                              Items: 5       Top record: 1/32
  (1)  (2)  (3)  (4)   (5)
  ID   D1S  D1S  D2S  SAMPLE
       1656 515  1329 NUMBER

 ID015 BB   BB   DE    W236
 ID019 BC   AA   AA    W240
 ID034 AE   AC   DF    W238
 ID038 AE   AC   AF    W216
 ID042 CD   AC   EF    W218
 ID045 BE   BD   BD    W230
 ID047 BB   AB   BE    W232

Enter [+]<n> or -<n> for new page; <n> (1-32) to set top row; ">" or "<"<n>
to shift window; I<n> (1-5) to set window; L to lock columns; T to change
instructions; N for new file; F to find; H for help; Q to quit.
>
```

When an unlinked file has been selected, INFER compares its Code File with entries in **CODE.STD** for standard mnemonics identifying Ego and Ego's parents. If the Code File does not contain mnemonics **FA** (or **SIRE**) and **MO** (or **DAM**), a pedigree file containing this information (along with Ego PIDs that match those in the input file) must be selected:

```
Pedigree information is not present in file merge.out

 1.  Unlinked File
 2.  Example Master File

Enter number of file containing pedigree.
>2
```

Here, the Example Master File has been designated. Next, loci are selected for analysis:

```
   1 EGO Permanent ID      * 3 D1S515 Genotypes        5  Sample No.
 * 2 D1S1656 Genotypes     * 4 D2S1329 Genotypes
--------------------------------------------------------------------------------
  INFER       Select loci               File - merge.out
--------------------------------------------------------------------------------
Enter number corresponding to the loci for which types are to be inferred,
or [C] to continue.
>
```

Here, three loci have been chosen. INFER makes two initial assumptions about genetic loci. First, it assumes that the data are *phenotypic* unless one of the last three mnemonic words associated with the Code File entry for a locus is "**GENO**" or "**G'TYPE**", in which case the data are interpreted as *genotypic*. Second, the program assumes that phenotypes are autosomal codominant and follow conventions listed in Appendix C, Rules for Genotype and Phenotype Nomenclature, unless specified otherwise in a Genotype-Phenotype Map File **GENOMAP.<EXT>** (see Appendix A, Definitions, Data, and File Types). INFER first searches the Data Directory for map file **GENOMAP.<EXT>**. If the file is not found, or if the selected phenotype is not included in it, the following display appears:

```
No entries for selected loci found in GENOMAP.XMP

Enter RETURN if all loci are autosomal co-dominant. Otherwise enter name of
alternate genotype-phenotype map file (loci not listed in a map file are
assumed to be autosomal co-dominant).
>
```

A Genotype-Phenotype Map is not required for autosomal codominant loci that adhere to the conventions listed in Appendix C. Nonetheless, is it frequently helpful to generate a map for autosomal codominant loci using program GENOMAP, because INFER will otherwise be forced to read through the entire Data File twice for each locus (once to tally phenotypes, and again to do the analysis). Presence of a map file eliminates the need for the tallying step and can significantly speed execution. If a Genotype-Phenotype Map File is found and it contains the selected phenotype, an alternate display appears:

```
GENOMAP.XMP contains entries for

     ABO PHENO TYPE (AUDR)

Enter RETURN to use GENOMAP.XMP.  Otherwise enter name of alternate
genotype-phenotype map file (loci not listed in a map file are assumed
to be autosomal co-dominant).
>
```

Next, the format of inferred types is specified:

```
Normally, inferred genotypes and phenotypes are enclosed in parentheses,
and incompletely known genotypes are represented by a single allele symbol
and a dash.

Enter - [P] to use parentheses and to display partial genotypes
         D  to delete parentheses and partial genotypes from output
>
```

Finally, output file types are selected:

```
 1. Full input records with appended inferred types, plus INFERERR.OUT.
 2. Individuals with known prior and/or inferred types, plus INFERERR.OUT.
 3. Both output file types, plus INFERERR.OUT
 4. INFERERR.OUT only.

Choose number corresponding to output format shown above.
>
```

2. A single locus chosen from the LOCUS Sub-directory

When the LOCUS Sub-directory is selected (choice **4** in the Introductory Display), the following display showing all files listed in its Directory File (**DBFILES**) appears:

```
  1. AllLoci
  2. CandLoci
  3. D1S1656
  4. D1S515
  5. D1S3721
  6. D2S1329
  7. D2S326
  8. D3S1229
  9. D4S1582
 10. D5S1466
 11. D6S1036
 12. D7S23
 13. GGAT1A4
 14. ABO
 15. ADA
 16. APOBpvu2a
 17. CA1
 18. ErrorDemo
 19. G6PD

Enter a file number or name.
>
```

Here, any single locus may be selected by entering its corresponding file number (3 - 18). Command sequences that follow are the same as those outlined above for unlinked files in the main Data Directory.

3. Multiple loci listed in a special-purpose file in the LOCUS Sub-directory

Alternately, a pre-defined set of loci (file numbers 1 and 2) may be chosen. The content of file 1 (**AllLoci**) is shown below:

```
FILELIST
D1S1656
D1S515
D1S3721
D2S1329
D2S326
D3S1229
D4S1582
D5S1466
D6S1036
D7S23
GGAT1A4
ABO
ADA
APOBpvu2a
CA1
ErrorDemo
G6PD
```

Note that the first record in the file must contain only the word FILELIST, and that this record is

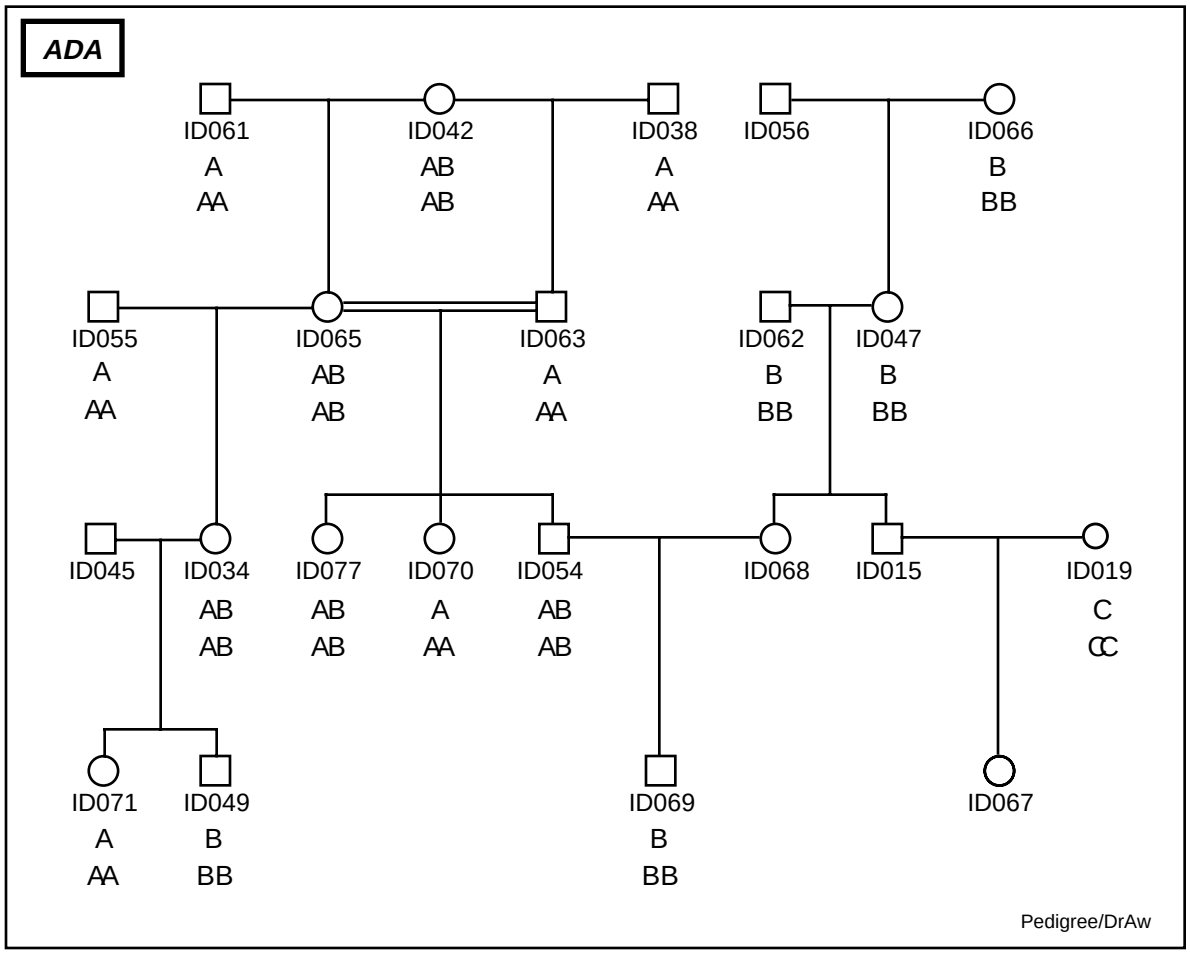
followed by a list of file names exactly as they appear in the sub-directory DBFILES. In the example above, we have included all locus files, but the list may be constructed to include any subset as shown below

```
FILELIST
ABO
ADA
APOBpvu2a
CA1
G6PD
```

Command sequences that follow are also the same as those outlined above for unlinked files in the main Data Directory.

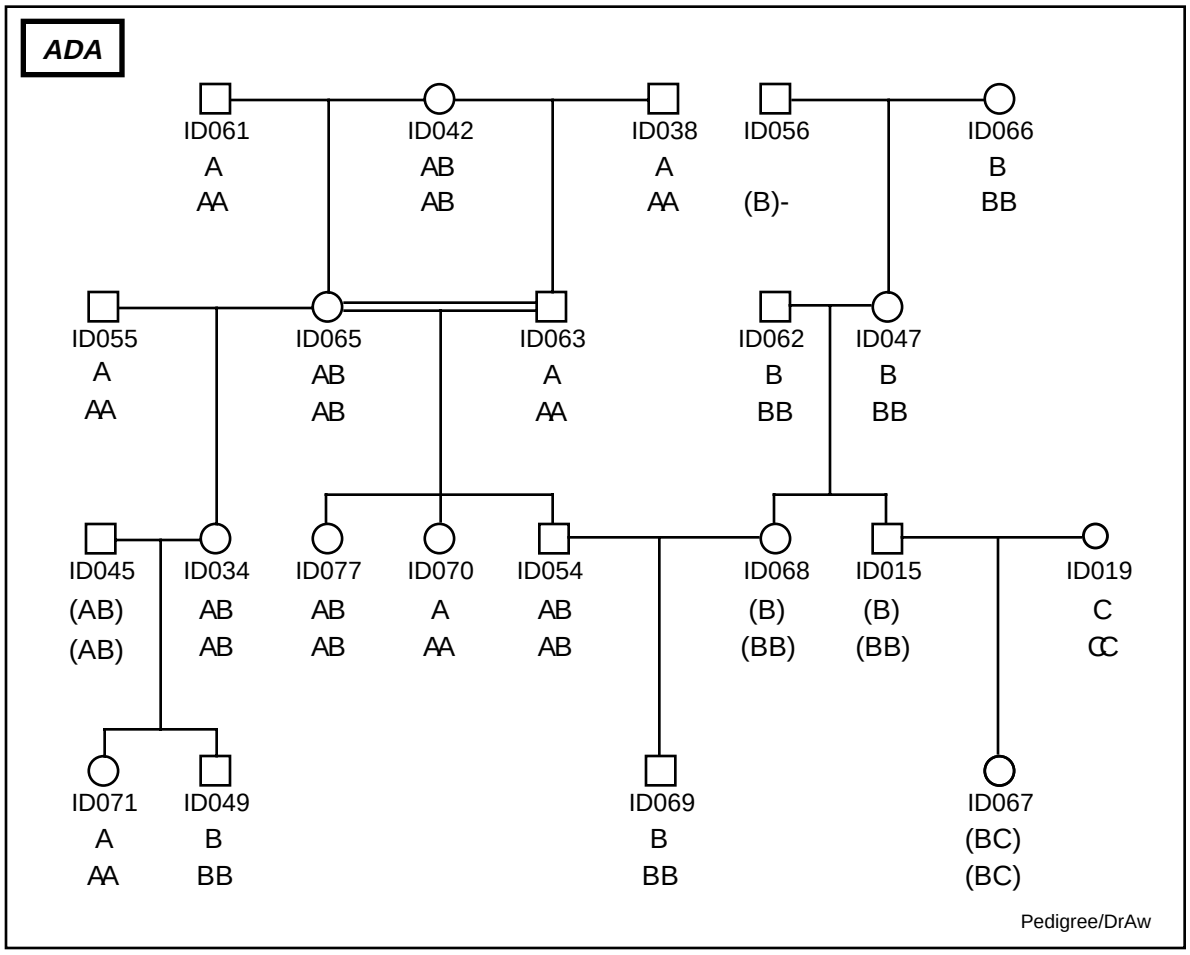
Example 1. Inferring Codominant Phenotypes (ADA locus)

ADA Phenotype is an autosomal codominant locus, so that it is unnecessary to create a Genotype-Phenotype Map File. INFER proceeds by assigning genotypes on an *a priori* basis, as illustrated in the diagram below:



Directly below each individual's PID is shown the ADA phenotype exactly as it appears in the ADA locus file. Note that there is no entry for individuals ID015, ID045, ID056, ID067, or ID068. Be-

low each phenotype is an *a priori* genotype assignment: since the locus is codominant, genotypes can be deduced directly from phenotypes. Note also that alleles and phenotypes are represented by identical symbols (here upper case), which illustrates the convention followed when a Phenotype-Genotype Map is not used. INFER starts with this information and iteratively attempts to fill in the missing types according to the Mendelian laws. The results can be seen in the next diagram:



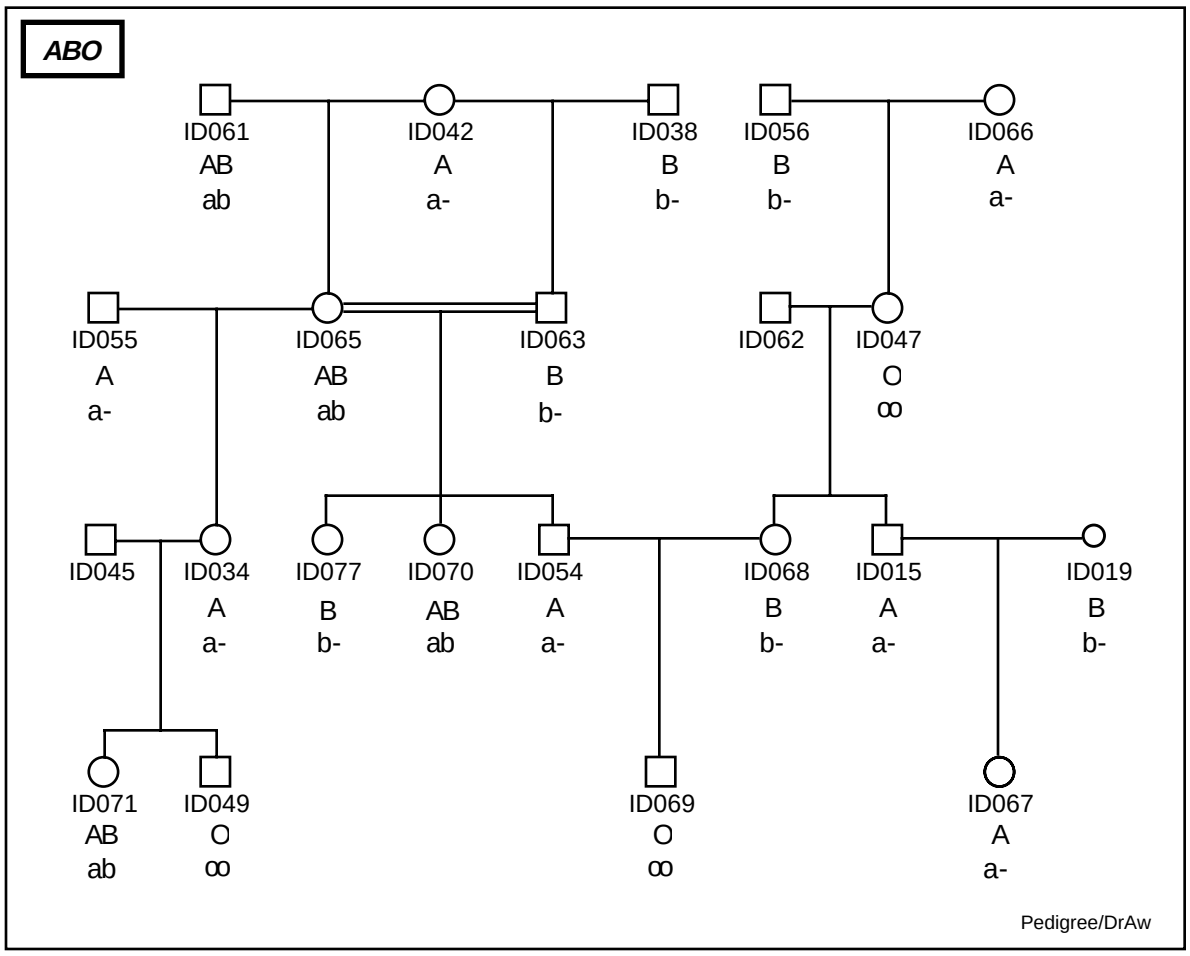
Here, inferred allele types, genotypes, and/or phenotypes are surrounded by parentheses. Inferences have been made from both parents and offspring: Since we know *a priori* that parents ID062 and ID047 are both **BB** homozygotes, we infer that their offspring ID068 and ID015 must share the same genotype and phenotype, assuming correct assignment of paternity. In contrast, we know *a priori* that offspring ID071 has received an **A** an allele from each parent and that ID049 has inherited a pair of **B** alleles from the same parents. Their mother ID034 is known to have phenotype **AB**, and so we infer that father ID045 is also phenotype **AB** of which he has inherited from ID034.

The iterative nature of the process is implied in the case of individual ID067, whose genotype (**BC**) could not be inferred completely until her father's genotype (**BB**) had been determined.

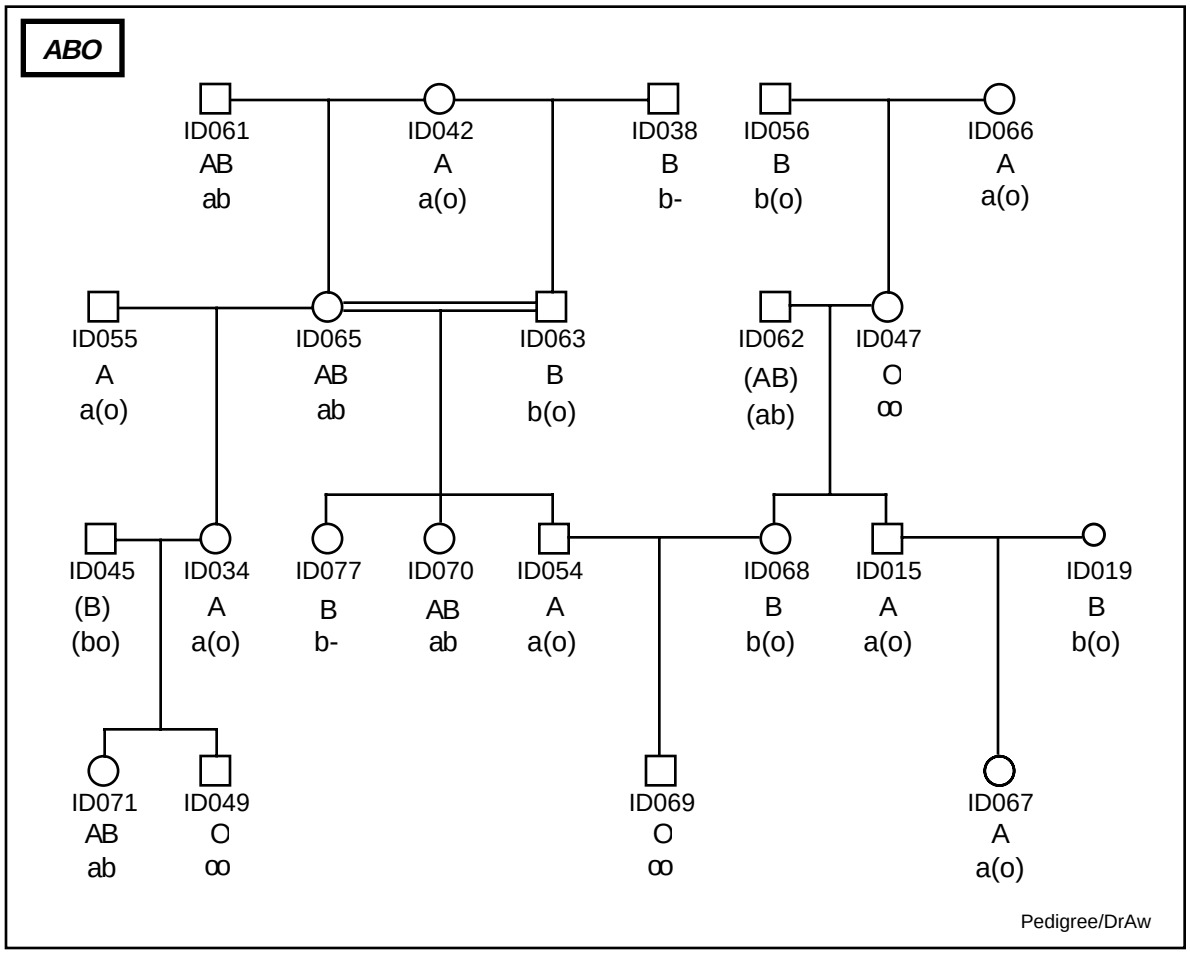
Note that in this pedigree it has not been possible to infer the phenotype of individual ID056. However, since his offspring (ID047) is a **BB** homozygote, we can infer that ID056 has at least one **B** allele, as is indicated by the symbol enclosed in parentheses. The other allele is unknown, however, and is represented by a dash.

Example 2. Inferring Dominant/Recessive Phenotypes (ABO locus)

A Phenotype-Genotype Map is required for correct inferences to be made for phenotypes that show dominance. If the selected phenotype is included in this file, INFER proceeds in the same fashion as it does when making inferences from codominant phenotypes. In the example that follows, the ABO Phenotype has been selected. INFER proceeds by assigning genotypes on an *a priori* basis, as illustrated in the diagram below:



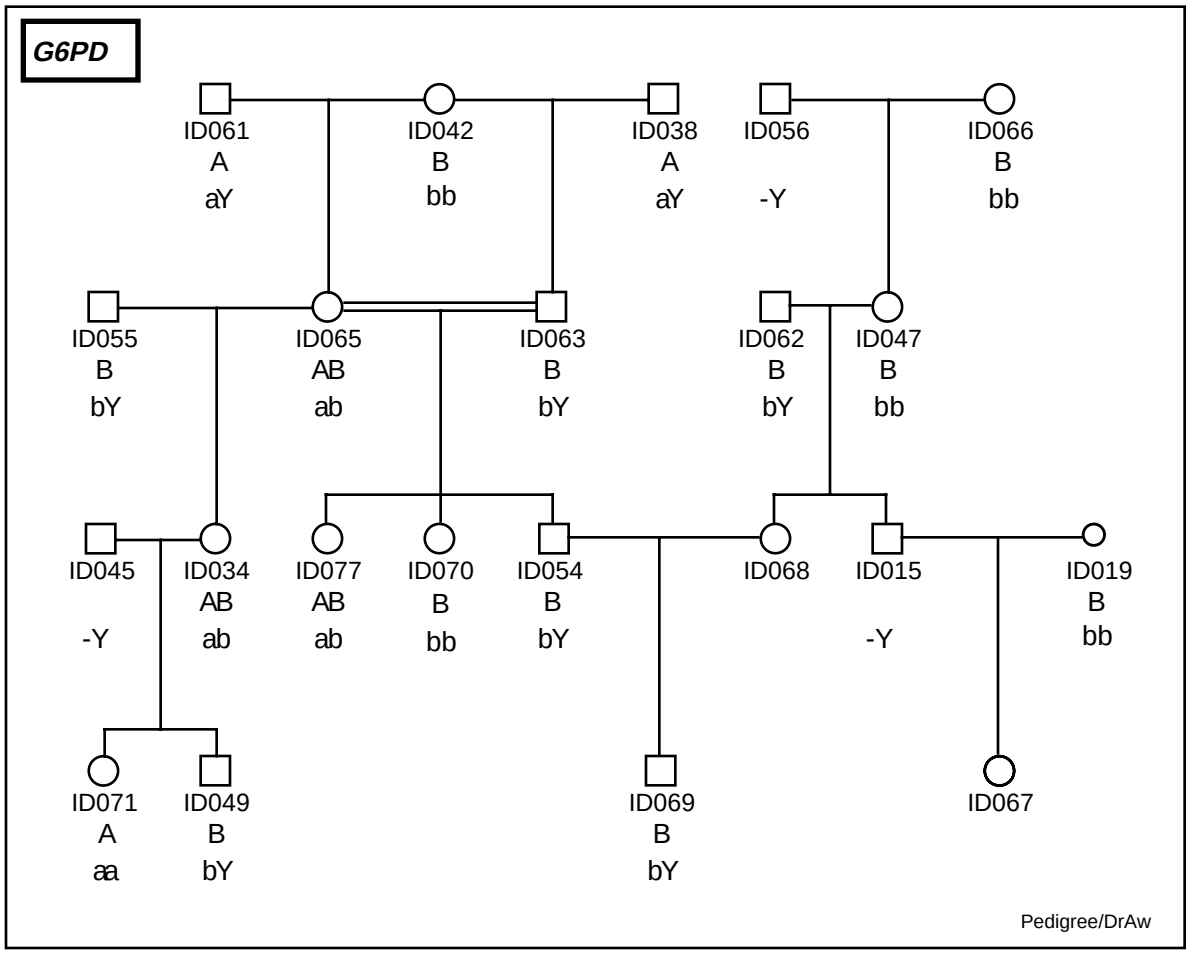
Directly below each individual's PID is shown the ABO phenotype exactly as it appears in the ABO locus file. Note that there is no entry for individuals ID045 or ID062. Below each phenotype is an *a priori* genotype assignment: recessive homozygotes **ab** and **oo** can be deduced unambiguously from phenotypes **AB** and **O**, respectively, and the presence of at least one **a** or **b** allele is known when phenotype **A** or **B**, respectively, is found. The latter case is designated by a single allele symbol followed by a dash, where the dash indicates that the second allele has as yet not been determined. INFER starts with this information and iteratively attempts to replace missing types according to the Mendelian laws, as illustrated in the next diagram:



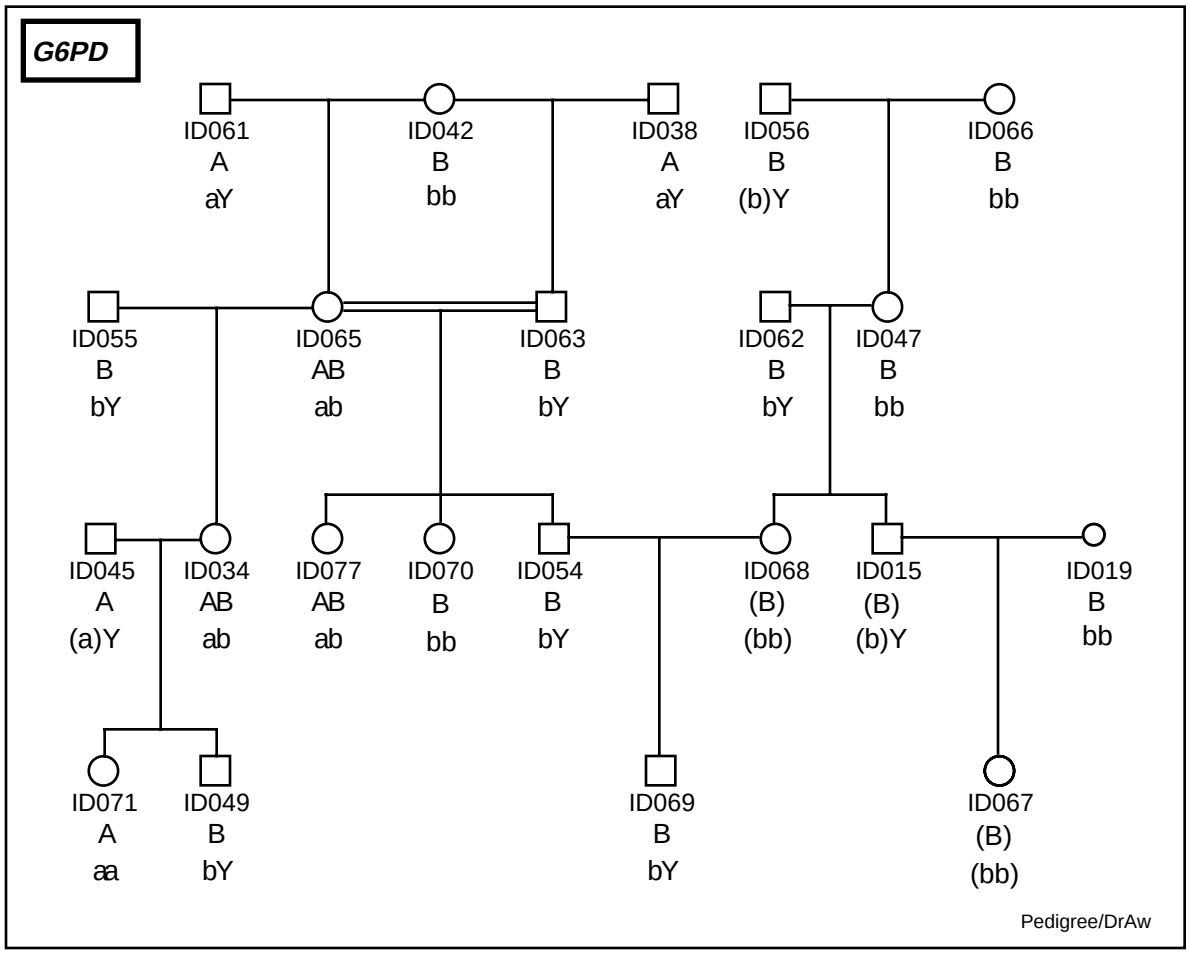
As before, inferred allele types, genotypes, and/or phenotypes are surrounded by parentheses. In this pedigree it has been possible to infer complete genotypes for all but two individuals (ID038 and ID077).

Example 3. Inferring X-linked Phenotypes

A Phenotype-Genotype Map is required for correct inferences to be made for X-linked phenotypes. INFER proceeds by assigning genotypes on an *a priori* basis, as illustrated in the diagram below:



Directly below each individual's PID is shown the G6PD phenotype exactly as it appears in the G6PD locus file. Note that there is no entry for individuals ID015, ID045, ID056, ID067, or ID068. Below each phenotype is an *a priori* genotype assignment: many genotypes can be deduced unambiguously since phenotypes are codominant in females and hemizygous in males. The symbol **Y** has been assigned to designate the Y chromosome in males. This has been paired with a dash in individuals ID015, ID045 and ID056, indicating that the X-linked G6PD allele determining their phenotype is not yet known. INFER starts with this information and iteratively attempts to fill in the missing types according to the Mendelian laws. The results can be seen in the next diagram:



Inferring from Genotypes

DNA marker types are typically referred to as genotypes, rather than phenotypes. INFER interprets marker data as genotypic only if the mnemonic word **GENO** or **G'TYPE** appears as one of the last three mnemonic words associated with the Code File entry. The program proceeds in the same fashion as it does when making inferences from codominant phenotypes.

Output:

Output of INFER may be directed optionally to files **infer1.out** and/or **infer2.out**. File **infer1.out** consists of all records from the input locus file in their original order. Each record contains input items in their original order, with the inferred phenotype (or genotype) appended at the end. An example of a display from program BROWSE, showing contents of **infer1.out** for the ABO locus file is shown below:

```
infer1.out                              Items: 4       Top record: 1/32
 (1)   (2)   (3)        (4)
 ID   ABO   ABO        ABO
      PHENO SAMPLE     PHENO

ID015 A      R236  A
ID019 B      R240  B
ID034 A      R238  A
ID038 B      R216  B
ID042 A      R218  A
ID045        R230  (B)
ID047 O      R232  O
ID049 O      R248  O
ID054 A      R234  A
ID055 A      R222  A
ID056 B      R214  B
ID061 AB     R212  AB
ID062        R224  (AB)

Enter [+]<n> or -<n> for new page; <n> (1-32) to set top row; ">" or "<"<n>
to shift window; I<n> (1-4) to set window; L to lock columns; T to change
instructions; N for new file; F to find; H for help; Q to quit.
>
```

In contrast, file **infer2.out** consists of records containing Ego PIDs from the input locus file, initial phenotype (that is, as it appears in the locus file), the inferred phenotype, the *a priori* genotype, the inferred genotype, and two mnemonics identifying the locus. Phenotype fields are left blank if inferences are made from genotypic data. The file contains a record for each individual for which an original type is available, or for which a phenotype or genotype can be inferred. An example of the contents of **infer2.out**, again with the ABO locus as input, is shown below:

```
infer2.out                                   Items: 6       Top record: 1/32
 (1)       (2)          (3)          (4)          (5)          (6)
 ID       INITAL       INFERD       PRIOR        INFERD       LOCUS
          PHENO        PHENO        GENO         GENO

ID015 A            A            a/-          a/(o)        ABO    PHENO
ID019 B            B            b/-          b/(o)        ABO    PHENO
ID034 A            A            a/-          a/(o)        ABO    PHENO
ID038 B            B            b/-          b-           ABO    PHENO
ID042 A            A            a/-          a/(o)        ABO    PHENO
ID045              (B)                       (b/o)        ABO    PHENO
ID047 O            O            o/o          o/o          ABO    PHENO
ID049 O            O            o/o          o/o          ABO    PHENO
ID054 A            A            a/-          a/(o)        ABO    PHENO
ID055 A            A            a/-          a/(o)        ABO    PHENO
ID056 B            B            b/-          b/(o)        ABO    PHENO
ID061 AB           AB           a/b          a/b          ABO    PHENO
ID062              (AB)                      (a/b)        ABO    PHENO

Enter [+]<n> or -<n> for new page; <n> (1-32) to set top row; ">" or "<"<n>
to shift window; I<n> (1-6) to set window; L to lock columns; T to change
instructions; N for new file; F to find; H for help; Q to quit.
>
```

Error Checking:

INFER identifies discrepancies between Ego and any one of Ego's parents. When a discrepancy is found, a record is written to file **infererr.out** containing PIDs and genotypes (including inferred types) of EGO, FA and MO, and mnemonics of the locus. Another type of error may occur when genotypes of parents are incompletely known and (a) there are more than 4 alleles represented in the offspring sibship, or (b) there are too many combinations of alleles represented in the sibship given what is known about the parents and grandparents. Here is an example of a display from program BROWSE, showing contents of **infererr.out**:

```
infererr.out                          Items: 8      Top record: 1/4
  (1)   (2)   (3)      (4)          (5)          (6)          (7)        (8)
  EGO   FA    MO       EGO          FA           MO           LOCUS      DSCRPT
                       G'TYPE       G'TYPE       G'TYPE                  PARENT

 ID063 ID038 ID042  B2    E       D     E       C     E      ERROR  DEMO   SorD
 ID068 ID062 ID047  B2    D                     E     E      ERROR  DEMO   D
 ID069 ID054 ID068  B2    C       E     E       B2    D      ERROR  DEMO   S
 ID071 ID045 ID034  D     D       C     E       E     E      ERROR  DEMO   SandD

Enter [+]<n> or -<n> for new page; <n> (1-4) to set top row; ">" or "<"<n> to
shift window; I<n> (1-8) to set window; L to lock columns; T to change
instructions; N for new file; F to find; H for help; Q to quit.
>
```

Discrepancy codes (Item 8 in the display above) are as follows:

S indicates that the father's genotype is inconsistent with the genotypes of Ego and Ego's mother.

D indicates that the mother's genotype is inconsistent with the genotypes of Ego and Ego's father.

SorD indicates that a discrepancy exists, but that it is not possible to determine which is the discrepant parent.

SandD indicates that both the father's and mother's genotypes are inconsistent with Ego's genotype.

FMLY indicates that discrepancies are found when grandparents and entire sibships are considered, even though single EGO-FA-MO triplets are consistent with Mendelian laws.

Notes:

- Enclosure of a discrepancy code in parentheses, for example **Sor(D)** or **(M)** indicates that the parental genotype has been inferred from genotypes of other family members.

- INFER automatically substitutes the symbols **S** for sire and **D** for dam in nonhuman animal populations, **F** for father and **M** for mother in human populations.

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **infer subset.out**.

Files:

1. Single files containing genotypes or phenotypes for one or more loci

Input: A Master File and a Data File having records containing genotypic or phenotypic data, with associated Code Files. The Data File need not be linked if it contains pedigree data, or if the pedigree can be supplied by another file.

GENOMAP.<EXT>, the Phenotype-Genotype Map File, contains information on mode of inheritance, forms of allele and phenotype symbols, and the relationship between genotypes and phenotypes for dominant and/or X-linked phenotypes.

An alternate Phenotype-Genotype Map File may be specified as input.

Output: **infer1.out** consists of a record for each individual represented in the input locus data file. Each record contains all the items of the input locus file in their original order, with the inferred phenotype (or genotype) appended at the end.

infer2.out contains a record for each individual in the Master File or pedigree file for which an original phenotype is available, or for which a phenotype or genotype can be inferred.

infer1.cde and **infer2.cde** define the record structures of **infer1.out** and **infer2.out**, respectively.

infererr.out contains a record for each individual for which a genotype cannot be inferred because of inconsistencies between Ego and one parent.

infererr.cde defines the record structure of **infererr.out**.

2. Multiple loci from file in the LOCUS Sub-directory

Input: A special-purpose file containing selected names of locus files in the LOCUS Sub-directory, plus the locus files and their associated Code Files.

A file (such as a Master File) containing pedigree data.

GENOMAP.<EXT> or an alternate Phenotype-Genotype Map File.

Output: **I<locusfilename>** contains records for each individual represented in the input locus data file. Each record contains all the items of the input locus file in their original order, with the inferred phenotype (or genotype) appended at the end.

infer2.out contains a record for each individual in the Master File or pedigree file for which an original phenotype is available, or for which a phenotype or genotype can be inferred.

infererr.out, **I<locusfilename>.cde**, **infer2.cde**, **infererr.cde**.

Recursive use of output files is permitted, that is, file **infer1.out** or **infer2.out** may be used as input to program INFER.

Algorithm: R. H. King, L. Freeman-Shade

ITEMIZE

ITEMIZE is a program used (a) to determine the PEDSYS format of records in files imported from foreign database systems, (b) to modify this format, and (c) to create a PEDSYS Code File and/or a Data File from the original data. The program shares some features of programs CODE and REFORMAT, but does not replace them.

Initial Display:

```
PEDSYS program ITEMIZE - Copyright 1992, 1999 SFBR


ITEMIZE determines the PEDSYS format of foreign ASCII data records.

Enter the name of the file for which PEDSYS format is to be defined.
>MASTER.XMP

Enter the name of the Code File for MASTER.XMP, or enter RETURN to create a
new Code File.
>
```

When the name of the input data file and the source for its Code File (if specified) has been entered, the following instruction appears (here, we will treat the Example Master File as if its record format is unknown by not specifying its Code File):

```
Some imported files keep data for a single individual on two or more contig-
uous records.  These data must be assembled into a single record in PEDSYS
format.

Enter the number of input records that must be assembled into a single
PEDSYS record, or RETURN if the input file contains only a single record
for each individual.
>
```

This feature makes it possible to transform multiple contiguous records into a single PEDSYS record for each individual. ITEMIZE assumes that each in the sequence of multiple input records has the same format. For example, although it is not necessary for the first and second records in a group to have the same format, all first records must be formatted identically, etc.

Next, the PEDSYS output file type is selected:

```
ITEMIZE will automatically assign Standard Mnemonics if the imported file is
to be used as input for program INDEX to create a new Master File.

Enter -  H to assign human mnemonics
         N for nonhuman mnemonics
        [C] if neither applies
>
```

Responses:

H or **N** If the foreign file contains pedigree information and is to be submitted to program INDEX to create a PEDSYS **Master File**, entry of **h** (or **n**) should be entered at this point. This command has the effect of directing ITEMIZE to check that pedigree data required in a Master File are present by searching for Standard Mnemonics (EGO or ID, BIRTH, FA or SIRE, MO or DAM, and SEX). An opportunity to assign Standard Mnemonics is provided if they are not found initially.

C Entry of **C** (or default RETURN) at this point begins the process of creating a PEDSYS **Data File**.

The parsing procedure described below is identical for both Data Files and Master Files, since checking for Standard Mnemonics is not done until the record structure of the foreign file has been determined (see the section on **Checking for Standard Mnemonics** below).

- **Starting without an initial Code File**

As an example, we will use ITEMIZE to determine the format of Example Master File records, as can be seen from the file name entered in the display above. We will assume for purposes of this example that no Code File exists for **MASTER.XMP**. An **n** has been entered in the display above, indicating that the output is to conform to Master File specifications. This is followed by the next display:

```
MASTER.XMP (Record 1 of 32, Length = 95)                          Page  1/ 1




::::|::::|::::|::::|::::|::::|::::|::::|::::|::::|::::|::::|::::|::::|:::
    1    0    0   12    0    0    0  1  0  0  0ID06119720309          M1998042



:|::::|::::|::::|
01ID042    1    0

Enter -  A to ASSIGN first item           [R] to display an alternate RECORD
         X to SAVE and EXIT
>a
```

The Status Line appears at the top, showing the file name, number of the input record displayed below, number of the last record in the input file, record length and display page number. Next, the first page of the input record is displayed as two lines of text (each up to 78 characters long) below an Input Ruler. The colon (**:**) symbol in the Input Ruler designates a character on the input record that has not yet been assigned to a PEDSYS data item; as a visual aid, every fifth unassigned character is marked with a stroke (**|**). Characters that have been assigned to a PEDSYS item are marked with an underline character (**_**), while characters on the input record that are not to be included as PEDSYS items are marked with a skip symbol (**.**).

In this example, the record requires only one page, and its second line of text does not extend across the screen. Below the data display are instruction and command lines (which may vary, depending on which operations are available). The consequence of choosing the options shown is as follows:

A (ASSIGN characters to Item)

Defining the first (or any) item consists of (a) determining the number of characters (starting at the beginning of the unassigned portion of the input record) that constitute a PEDSYS data item, and (b) assigning a Data Type and single mnemonic to the newly defined item. From the example above it can be seen that an **a** has been entered at the command line, which produces the following display:

```
MASTER.XMP (Record 1 of 32, Length = 95)                          Page  1/ 1



::::|::::|::::|::::|::::|::::|::::|::::|::::|::::|::::|::::|::::|::::|:::
    1    0    0   12    0    0    0  1  0  0  0ID06119720309          M1998042


:|::::|::::|::::|
01ID042    1    0

Item 1:
>5 <     Enter number of characters to be assigned to this item (1-95).
>i<      Enter the Data Type of the item (C, I, or R), or "S" to skip.
>SEQ   < Enter a single mnemonic for this item.
```

The current Item Number is displayed just below the input record display. This is followed by a request for the number of characters in the record that are to be assigned to the item, starting with the first unassigned character. At this point, the unassigned portion starts at the beginning of the record, and it can be seen that the first non-blank character in the record (**1**) is five spaces from this point. This suggests that the first item is a five-character integer (which would be a reasonable guess, even without prior knowledge of the record format). Hence, 5 characters have been assigned to this item. In the next command, its Data Type is set at **i**. (Entry of **s** here defines the item as one to be skipped when writing output records to file **itemize.out**.) Finally, the mnemonic **SEQ** is assigned to the new item. When the mnemonic has been entered, the full command display appears:

```
MASTER.XMP (Record 1 of 32, Length = 95)                          Page  1/ 1


1
SEQ
_____ ::::|::::|::::|::::|::::|::::|::::|::::|::::|::::|::::|::::|::::|::::|::
    1     0    0   12    0    0    0  1  0  0  0ID06119720309          M199804


::|::::|::::|::::|
201ID042    1    0

Enter -  A to ASSIGN next item              I to INSERT new Item
         S to SUBDIVIDE Item                D to DELETE numbered item
         M to change MNEMONIC               T to change Data TYPE
         W to change WIDTH of Item         [R] to display an alternate RECORD
         X to SAVE and EXIT
>a
```

Here, the first item has been defined at the left-most side of the data display: The item number **1** appears first, with mnemonic **SEQ** just below it. Below this is a row of five underline characters (_) forming an Item Ruler. This pattern is repeated for subsequent assignments (using command **A**), with new items appended to the new row left-to-right, while the row of unassigned characters recedes correspondingly. With each new item defined, the Input Ruler is repositioned so that its leading character is located at the start of the unassigned portion of the input record.

The next display shows the result of having defined two additional items, and the start of designating the next as a **skip** field:

```
MASTER.XMP (Record 1 of 32, Length = 95)                          Page  1/ 1


1     2     3
SEQ   SSEQ  DSEQ
_____ _____ _____ ::::|::::|::::|::::|::::|::::|::::|::::|::::|::::|::::|::::|
    1     0     0    12    0    0    0  1  0  0  0ID06119720309          M1998



::::|::::|::::|::::|
04201ID042    1    0

Item 4:
>32<     Enter number of characters to be assigned to this item (1-80).
>s<      Enter the Data Type of the item (C, I, or R), or "S" to skip.
```

Here, the next 32 characters of the input record (starting at the blank character four spaces prior to the **7**, and ending with the **0** just before the characters **ID061**...) will be assigned to a single item with the mnemonic **SKIP**. The result of this operation are shown below:

```
MASTER.XMP (Record 1 of 32, Length = 95)                          Page  1/ 1


1     2     3     4
SEQ   SSEQ  DSEQ  SKIP
_____ _____ _____ ................................ ::::|::::|::::|::::|::::|::
    1     0     0    12    0    0    0  1  0  0  0 ID06119720309          M199



::|::::|::::|::::|:::
804201ID042    1    0

Enter -  A to ASSIGN next item              I to INSERT new Item
         S to SUBDIVIDE Item                D to DELETE numbered item
         M to change MNEMONIC               T to change Data TYPE
         W to change WIDTH of Item         [R] to display an alternate RECORD
         X to SAVE and EXIT
>
```

The Item Ruler of the fourth item is now made up of **.** symbols, and labeled with the mnemonic **SKIP**. The functional consequence of this is that the 32 characters in this field need not be included in records of the output file.

M (Change MNEMONIC)

T (Change Data TYPE)

Entering **M<n>** or **T<n>**, where <n> is a number corresponding to a previously entered item, produces a request for a new Mnemonic or Data Type, respectively. Changing the Mnemonic to **SKIP** or the Data Type to **S** designates the item chosen as a **skip** field.

I (INSERT new Item)

D (DELETE numbered Item)

Entry of **I** produces a request for the number of a previously entered item **prior** to which a new item is to be inserted. Entry of **D<n>** where <n> is a number (or range of numbers) corresponding to a previously entered item (or contiguous items), results in deletion of those items. Either of these operations results in a downstream frame shift, that is, all item boundaries to the right of the modified item are shifted right (in the case of Insertion) or left (in the case of Deletion) by the number of characters in the modified item. Characters included in the unassigned portion of the record are shifted accordingly. The width of an inserted item is limited to **1** through **<unchars>**, where <unchars> is either the number of characters as yet unassigned to in the input record, or 99, the maximum number of characters permitted in a PEDSYS Data Item. The upper limit implies that an insertion may not be made if its width exceeds the number of unassigned characters remaining in the input record. If an item of a certain width cannot be inserted because there are insufficient characters remaining, it will be necessary to decrease the width or delete one or more items so as to accommodate the frame shift.

W (Change WIDTH of Item)

Entering **W<n>**, where <n> is a number corresponding to a previously entered item, produces a request for a new width of the item. Changing the width of an item shifts its right-hand boundary in either direction by a number of characters equal to the difference between the new and old widths (old minus new). This change produces a downstream frame shift, that is, all item boundaries to the right of the modified item are also shifted left or right by the difference in old and new widths of the modified item. Characters included in the unassigned portion of the record are shifted accordingly. Limits on width are **1** through **<unchars>**, where <unchars> is either the number of characters as yet unassigned to in the input record, or 99, the maximum number of characters permitted in a PEDSYS Data Item. These limits imply that an item cannot be deleted by changing its width, and that its width cannot be increased unless there is a sufficient number of characters in the input record that have as yet been unassigned to data items. If the width of an item cannot be increased because there are not enough unassigned characters remaining, it will be necessary to delete or decrease the width of one or more items so as to accommodate the frame shift.

S (SUBDIVIDE an Item)

The **S** command is used to subdivide an existing item without changing its width. Entry of **S<n>**, where <n> is a number corresponding to the item to be subdivided, produces the following display:

```
Subdivide SKIP



::::|::::|::::|::::|::::|::::|::
   12    0    0    0  1  0  0  0



Item 4:
>5  <   Enter number of characters to be assigned to this item (1-32).
>i<     Enter the Data Type of the item (C, I,  or R), or "S" to skip.
>KID1  < Enter a single mnemonic for this item.
```

Here, subdivision of Item 4, (initially designated as a skip field) has been selected, and is displayed alone on the screen as an unassigned string. Definition of items proceeds in the same way as described for the original assignment of data characters. Here also, the first five characters are defined as integers and given the mnemonic **KID1**, which results in the following display

```
Subdivide SKIP


4
KID1
_____ ::::|::::|::::|::::|::::|::
   12     0    0    0  1  0  0  0



Item 5:
>   <   Enter number of characters to be assigned to this item (1-27).
```

This pattern is repeated for subsequent assignments. Item numbers are incremented and displayed as items are defined. When all characters have been assigned, the screen returns to a full command display with all items numbered consecutively.

R<n> (Display RECORD <n>)
RETURN (Display next record)

ITEMIZE starts with a display of the first record in the input file selected. However, other records in the file may contain additional information (such as non-blank items) that clarify the format. Other records in the file may be displayed in one of two ways. **RETURN** alone moves the display to the next record in sequence; entry of **R<n>** displays the contents of record <n>, where <n> is a number corresponding to a number in the range **1** through **<nrecs>** and <nrecs> is the number of records in the file. All other screen characteristics (ruler, instruction and command lines, etc.) remain unchanged.

X (EXIT)

Entry of **X** causes the program to write output in files **itemize.out** and **itemize.cde**. If any items have been assigned the mnemonic **SKIP**, however, the following instruction is displayed before files are saved:

```
Enter -  E  to EXCLUDE all skipped fields.
         B  to exclude skipped fields only if BLANK in all records.
        [I] to INCLUDE skipped fields.
         Q  to QUIT program without saving output.
>
```

Usually, skipped fields are not wanted in program output (option **E**). However, their inclusion (option **I**) is sometimes useful because an unreduced **itemize.cde** can be used as a Code File for the original input Data File. It is frequently desirable to skip fields that are blank on every record of the input file. Unless every record is examined, however, it is sometimes possible to miss an infrequent non-blank entry. Choice of option **B** avoids this possibility by preserving skipped fields if they contain a non-blank character on any record in the file.

If upon EXIT there are characters remaining in the input Data Record that have not been assigned to Data Items, the following instruction is displayed before files are saved:

```
Enter - D  to DISCARD unassigned portion of input record.
       [S] to SAVE unassigned portion of input record.
        Q  to QUIT program without saving output.
>
```

Saving the unassigned portion of the input Data Records (along with skipped fields as described above) makes it possible to interrupt execution of the program, and to return to assignment of characters to PEDSYS Data Items at a later time.

Input Data Records, like PEDSYS records, are limited to 1024 characters in length. Unlike PEDSYS records, however, they may contain more than 100 variables. Thus, it may be necessary to use ITEMIZE on the same input file more than once, assigning as many characters as one can to 100 PEDSYS Items, and then saving the unassigned portion of the record as described above. Program REFORMAT can then be used to divide **itemize.out** records into two files, and the file containing the unassigned portion may then be run through ITEMIZE again.

In order for PEDSYS programs to access the unassigned portion of the record, however, ITEMIZE automatically assigns it PEDSYS format. That is, it is given the mnemonic **UNASG<n>** (where $1 \leq n \leq 9$), and assigned Data Type **C**. If the unassigned portion is greater than 99 characters (the maximum length of a PEDSYS Item), it is automatically divided into segments no longer than 99 characters, with all segments assigned a numbered mnemonic (UNASG1 through UNASG9). Output Code File **itemize.cde** will then contain descriptor records for each of these automatically created items, making it possible for programs like REFORMAT, etc., to process **itemize.out** even though it has not been put into its final PEDSYS format.

- **Starting with an initial Code File**

A Code File may be named in response to the second request in the Initial Display shown above, in which case ITEMIZE displays records with characters assigned to PEDSYS Data Items as specified by the Code File, rather than as a string of unassigned characters. There is no requirement that this Code File in fact fit the structure of the record, but having an approximate pattern may be an efficient way to get started in determining the PEDSYS format of records in the file.

Once an input file has been selected, the program examines the entire file to find the length of its longest record. If the length of this record , and record length as specified by the Code File are not the same, the program uses by default the longer of the two as the length of the output record. This leads to two possibilities:

a. If the length specified in the Code File is greater than the maximum record length found in the input Data File, all items listed in the Code File are displayed on the screen with their numbers, mnemonics and ruler. As many characters as there are in the Data File are assigned to these items and are shown in the row beneath them. Code File items for which there are no corresponding data are displayed with blank characters in the row beneath them. *Unless explicitly skipped, these blanks are included in the output file* **itemize.out** and are defined in **itemize.cde**.

b. The length of the longest record found in the file is greater than that specified by the Code File. In this case, characters from the input Data Record are assigned to as many Code File items as there are, and the remainder are displayed as a single unformatted text string.

Checking for Standard Mnemonics

If the option to prepare the foreign file for use as a PEDSYS Master File, has been selected, ITEMIZE checks to see that Standard Mnemonics (EGO or ID, BIRTH, FA or SIRE, MO or DAM, and SEX) have been designated. If all are present, ITEMIZE proceeds to write its output files. However, if one or more are missing, an opportunity to assign Standard Mnemonics is provided. As an example of the latter operation, note that Item **2** in the display below (**DOB**) does not conform to the standard BIRTH:

```
examped (Record 1 of 32, Length = 48)                              Page  1/ 1

1     2        3     4     5   6        7      8     9     10
ID    DOB      SIRE  DAM   SEX EXIT     EXCODE MATE  PEDNO GEN
_____ ________ _____ _____ _   ________ _      _____ _____ _____
ID061 19720309             M   19980420 1      ID042     1     0



Enter -  S to SUBDIVIDE Item               D to DELETE numbered item
         M to change MNEMONIC              T to change Data TYPE
         W to change WIDTH of Item        [R] to display an alternate RECORD
         X to SAVE and EXIT
>
```

When an **x** is entered under these conditions, the following display appears:

```
   1 EGO Permanent ID        5 Sex (M, F or U)         9 PEDIGREE NUMBER
   2 Birth Date (YYYYMMDD    6 Exit Date (YYYYMMDD)   10 GENERATION NUMBER
   3 Sire's Permanent ID     7 Exit Code
   4 Dam's Permanent ID      8 Mate's Permanent ID
--------------------------------------------------------------------------------
  ITEMIZE     Select Standard Mnemonics       File - examped
--------------------------------------------------------------------------------
A Standard Mnemonic required for Master File is missing. Enter number
corresponding to BIRTH, RETURN alone to skip field.
>2
```

Item 2 has been selected here, and the correct mnemonic will be substituted automatically. The program proceeds through any other missing Standard Mnemonics, and when all corrections have been made, writes output files **itemize.out** and **itemize.cde**.

ITEMIZE employs some simplifying conventions in generating its output Code File.

a. Only one mnemonic word may be entered for each data item defined. If an input Code File has been used, any change in mnemonics results in the new single mnemonic completely replacing all the original mnemonics.

b. When a new Code File is generated, any mnemonic entered is used also as the item descriptor label. When an input Code File is used, the original item descriptor labels remain unchanged even when a new mnemonic is entered.

c. The item descriptor label remains unchanged when a missing Standard Mnemonic is added.

To bypass these limitations, use program CODE to modify file **itemize.cde**.

Notes:

- To reduce file storage requirements, some computers and programs ignore trailing blank characters at the end of records. Because most PEDSYS programs require that all Data Records be exactly the same length, it is safer to allow ITEMIZE to write **itemize.out** in which trailing blanks are always included, even when inclusion of skipped fields means that it is simply a copy of the input Data File.

Files:

Input: Any formatted ASCII text file, with or without a Code File.

Output: **itemize.out** contains a copy of the input file, with or without the inclusion of fields designated to be skipped.

itemize.cde defines the structure of **itemize.out**.

Program Limits:

A maximum of 500 items may be defined for any one PEDSYS record. Width of a record displayed on the screen may not exceed 1024 characters

KINSHIP

Program KINSHIP calculates coefficients of kinship (ϕ) for pairs of individuals, or inbreeding coefficients (F) by either the Stevens-Boyce, or the Quaas-Henderson algorithm (Boyce 1983). The file from which pedigrees are read may be either a Master File, or an indexed pedigree file consisting of records containing minimally IDs for Ego, Ego's parents and (if required) Ego's sex. Individuals for which coefficients are to be calculated may be entered from either the keyboard or from a Proband Input File (a file consisting of a list of either Sequential or Permanent IDs), or may consist of all members of the input pedigree. Output consists of a file containing IDs of Ego and Ego's parents (if inbreeding has been calculated) or Ego and another individual (for kinship), plus the appropriate coefficient.

Introductory Display:

```
PEDSYS program KINSHIP - Copyright 1992, 1999 SFBR

KINSHIP calculates coefficients of kinship (phi) for pairs of individuals or
inbreeding coefficients (F) for single individuals by the Stevens-Boyce or
Quaas-Henderson algorithm. Stevens-Boyce is usually more efficient for ped-
igrees of lower, Quaas-Henderson for higher than average kinship.

 1. Inbreeding (Stevens-Boyce)
 2. Kinship    (Stevens-Boyce)
 3. Inbreeding (Quaas-Henderson)
 4. Kinship    (Quaas-Henderson)

Enter number corresponding to choice of algorithm, or "Q" to quit.
>
```

Any method chosen above produces a request for an input file containing one or more pedigrees that include individuals whose coefficients are to be calculated:

```
 1.  Unlinked pedigree file
 2.  Example Master File

Enter number, file name, or "Q" to quit.
>1
```

KINSHIP allows the use of input only from an indexed pedigree file (that is, one whose records contain minimally IDs for Ego and Ego's parents). These conditions are fulfilled automatically if a Master File is selected, but an unlinked Data File requires preprocessing with program INDEX.

Calculating inbreeding coefficients

Inbreeding coefficients are calculated for individuals when choice **1** or **3** is selected in the command line of the Introductory Display. Once a Master File or indexed pedigree file with an appropriate Master Pointer File has been selected, the following display appears from which the source of probands for which inbreeding coefficients are to be calculated may be chosen:

```
Coefficients may be calculated for individuals entered from

  1.  Keyboard
  2.  Unlinked File
  3.  EXAMPLE
  4.    LOCUS Sub-directory

Enter number above corresponding to the appropriate source of probands
>1
```

1. Proband IDs entered from the keyboard

Choice **1** (shown above) specifies that Ego IDs are to be entered one-by-one from the keyboard:

```
Enter [P] if individuals are identified by Permanent IDs, or "S" if by
Sequential IDs
>

Enter Ego Permanent ID.  RETURN alone terminates entry.

>ID034< ID
>ID015< ID
>     < ID

 [1] Save all output records in file kinship.out.
  2. Discard output records with zero coefficients.
  3. Save records with non-zero coefficients in kinship.out, zero
     coefficients in nokinship.out.

Enter number corresponding to choice
>2
```

Here, two PIDs have been entered, and a choice has been made to skip records of any individuals with inbreeding coefficients of zero (that is, they not written to the output file).

2. Proband IDs read from a file

A proband file is often a subset of the input pedigree file (typically a Master File), but may be any Data File whose records contain IDs in the appropriate form (that is, Permanent or Sequential). Selection of a single item containing the proband ID is made from a Standard Item List Display:

```
   1 EGO Permanent ID        5 Sex (M, F or U)         9 PEDIGREE NUMBER
   2 Birth Date (YYYYMMDD    6 Exit Date (YYYYMMDD)   10 GENERATION NUMBER
   3 Sire's Permanent ID     7 Exit Code
   4 Dam's Permanent ID      8 Mate's Permanent ID
--------------------------------------------------------------------------
  KINSHIP     Identify PROBAND                File - examped
--------------------------------------------------------------------------
Enter a number above corresponding to Proband ID, or [C] to continue to
next step.
>8
```

Here, inbreeding coefficients of individuals identified by PIDs as mates have been specified. Calculations will be done for mates found on all records of file **examped**.

Calculating kinship coefficients

Coefficients of kinship are calculated between pairs of individuals when choice **2** or **4** is selected in the command line of the Introductory Display. The *source* of these pairs is controlled in the next display:

```
Calculate kinship between pairs of individuals

 1. Both members of which are selected from the same file (or from the same
    list entered from the keyboard).
 2. One member of which is selected from one file (or list), and the other
    member from a second file (or list).

Enter number above corresponding to the source of pairs, or "Q" to quit.
>1
```

1. Both members of pairs selected from the same source.

A **1** entered in the command line specifies that both members of the pair are to be selected from the same source. This choice is followed by the following display, from which the *composition* of selected pairs is specified (opposite-sex, same-sex, etc.):

```
Kinship coefficients may be calculated between individuals specified as

 1. All opposite-sexed pairs
 2. All same-sexed pairs
 3. All pairs regardless of sex
 4. Specified pairs of IDs

Enter number corresponding to choice
>1
```

- **Options 1 - 3** -- Any of the first three options selected from the pair composition menu above results in a request for the source of pairs for which calculations will be made. Here, a **1** has been entered, specifying that all opposite-sexed pairs will be formed and analyzed.

```
Coefficients may be calculated for pairs entered from

  1.  Keyboard
  2.  Unlinked File
  3.  EXAMPLE
  4.    LOCUS Sub-directory

Enter number above corresponding to the appropriate source of pairs
>
```

Proband IDs entered from the keyboard

Prior to entry, the form of the IDs (Permanent or Sequential) must be specified. This is followed by a request for a list of IDs:

```
Enter [P] if individuals are identified by Permanent IDs, or "S" if by
Sequential IDs.
>p

Enter Ego Permanent ID. RETURN alone terminates entry.

>ID071< ID
>ID049< ID
>ID069< ID
>ID067< ID
>     < ID
```

Here, Permanent IDs of four individuals have been entered.

Proband IDs read from a file

Selection of proband ID is made from a Standard Item List Display:

```
  1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
  2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZE  16 Sex (M, F or U)
  3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZE  17 Exit Date (YYYYMMDD)
  4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
  5 NEXT PATERNAL SIB SEQ *12 EGO Permanent ID       19 Pedigree ID Number
  6 NEXT MATERNAL SIB SEQ  13 Birth Date (YYYYMMDD)  20 Generation Number
  7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 Mate's Permanent ID

--------------------------------------------------------------------------------
  KINSHIP      Identify PROBAND             File - MASTER.XMP
--------------------------------------------------------------------------------

Enter number above corresponding to Proband ID.  Enter "C" to continue.
>
```

Here, measurement of kinship coefficients between individuals identified by Ego Permanent ID has been specified.

- **Option 4** -- Choice of fixed pairs of IDs from the pair composition menu limits kinship calculations to specific pairs of individuals whose IDs are either entered in pairs from the keyboard, or read in pairs from records in a file. This option likewise results in a request for the source of pairs for which calculations will be made. Here, a **1** has been entered, specifying that all opposite-sexed pairs will be formed and analyzed.

Proband IDs entered from the keyboard:

Prior to entry, the form of the IDs (Permanent or Sequential) must be specified. This is followed by requests for pairs of IDs:

```
Enter RETURN if individuals are identified by Permanent IDs, or "S" if by
Sequential IDs.
>

Enter pairs of Permanent IDs. RETURN alone terminates entry.

>ID034< ID 1
>ID054< ID 2

>ID069< ID 1
>ID067< ID 2

>     < ID 1
```

Here, two pairs of PIDs have been entered.

Proband IDs read from a file:

Selection of a pair of items containing the proband IDs is made from a Standard Item List Display:

```
   1 EGO Permanent ID        5 Sex (M, F or U)         9 PEDIGREE NUMBER
   2 Birth Date (YYYYMMDD    6 Exit Date (YYYYMMDD)   10 GENERATION NUMBER
   3 Sire's Permanent ID     7 Exit Code
   4 Dam's Permanent ID      8 Mate's Permanent ID
--------------------------------------------------------------------------
 KINSHIP      Identify PROBANDS            File - subset.out
--------------------------------------------------------------------------

Enter a single pair of numbers above corresponding to Proband Ids, [C] to
continue.
>
```

Here, measurement of kinship coefficients between Ego and mate (both designated by Permanent IDs) has been specified. Calculations will be done for Ego-mate pairs found on all records of file **examped**.

2. Each member of pair selected from a different source.

Choice **2** entered for pair formation specifies that all possible pairs will be formed by matching the first member selected from one source (a file or list entered from the keyboard) with the second member from a second source. Each of the two sources must be chosen as follows:

```
Select the FIRST member of each pair from

  1.  Keyboard
  2.  Unlinked file
  3.  subset.out
  4.  EXAMPLE

Enter number above corresponding to the appropriate source of individuals
>1

Select the SECOND member of each pair from

  1.  Keyboard
  2.  Unlinked file
  3.  subset.out
  4.  EXAMPLE

Enter number above corresponding to the appropriate source of individuals
>3
```

Here, the first member of each pair will be entered from the keyboard and the second from file **subset.out**. Next, the type of ID by which pairs are to be identified is selected, after which IDs of the first member of each pair is entered:

```
Enter RETURN if individuals are identified by Permanent IDs, or "S" if by
Sequential IDs.
>

Select FIRST member of each pair:
Enter Ego Permanent ID. RETURN alone terminates entry.

>ID071< ID
>ID067< ID
>ID045< ID
>     < ID
```

Here, the first member of each of three pairs has been selected. The second member of each pair is accessed automatically from file **subset.out** (which in this example contains records for individuals ID049 and ID069). Altogether, six coefficients are calculated:

```
kinship.out                          Items: 5       Top record: 1/6
  ( 1)  ( 2)  ( 3)  ( 4)     ( 5)
  SEQ1  SEQ2  ID1   ID2      Phi

    17    18 ID071 ID049  0.250000000
    17    19 ID071 ID069  0.039062500
    20    18 ID067 ID049  0.000000000
    20    19 ID067 ID069  0.062500000
    11    18 ID045 ID049  0.250000000
    11    19 ID045 ID069  0.000000000

Enter +<n> or -<n> for new page; <n> (1-4) to set top row; ">" or "<"<n> to
shift window; I<n> (1-5) to set window; L to lock columns; T to change
instructions; N for new file; F to find; H for help; X to exit.
>
```

Notes:

- KINSHIP requires that records of parents appear prior to those of their offspring in the input file. Master Files and pedigree files produced by program INDEX meet this requirement. It is a good idea to run INDEX on a pedigree file if there is any doubt about the order of its records. The Master Pointer File option in INDEX is required in this case.

- The Stevens-Boyce algorithm uses a path-searching method based on the relationship

$$f = \sum_{i=1}^{l} \left(\left(\tfrac{1}{2}\right)^{n_i} (1 + f_a) \right)$$

  where f is the inbreeding coefficient of an individual, l is the number of loops of relationship formed by following paths to and from Ego and an ancestor common to both parents, and n_i is the number of individuals in each loop (excluding Ego).

- The Quaas-Henderson algorithm depends on the relationship

$$f_{XY} = 0.5(f_{XS} + f_{XD})$$

  where f_{XY} is the coefficient of kinship between individuals X and Y, and f_{XS} and f_{XD} are the kinships measured between X and the father and mother, respectively, of Y (see Boyce 1983 for details of the algorithm itself).

- The number of pairs for which kinship coefficients are to be computed can become very large, even with modest pedigree sizes. For example, the total number of pairs (disregarding sex) that can be formed from a Master File containing n = 1,500 individuals is $n(n - 1)/2$ = 1,124,250. In order for KINSHIP to compute coefficients for so many pairs, it automatically subdivides the input file and sequentially processes it in smaller groups of individuals. This requires the program to intersperse calculations with repeated reads and writes to a temporary disk file. Consider carefully the need to compute kinship for large samples, and be prepared for extended execution times in these cases.

- Choice of algorithm depends on the number of coefficients to be calculated and the complexity and generational depth of the pedigree. The Stevens-Boyce method must count the number of individuals in each loop connecting Ego to a common ancestor. Computations are minimal if Ego is not inbred, but the task may be time-consuming if Ego's parents are related in complex ways, making the number of loops very large, as may be the case for some domestic animals that are produced by repeated backcrossing. The Quaas-Henderson method starts at the top of a pedigree, and computes kinships for individuals in subsequent generations on the basis of coefficients already computed. This approach avoids repeated counting of complete loops, but requires computations even for unrelated parents. Thus, Stevens-Boyce is usually more efficient for calculations of coefficients in pedigrees where average kinship is low, while Quaas-Henderson is better for large, deep pedigrees of higher average kinship. The table below gives comparisons of the two methods used to calculate inbreeding coefficients for individuals in four separate populations from Southwest Foundation for Biomedical Research pedigree databases. Both methods yield exactly the same numerical results.

Population	Colony Size	No. of Pedigrees	Max. Generations	Mean F	Stevens-Boyce	Quaas-Henderson
*Monodelphis domestica**	30,162 50**	2 1	22 22	.2781 .2253	---- 6:35:48	0:25:32 0:00:14
Baboons	11,450	285	4	.0046	0:10:33	0:31:42
Humans 1	6,462	41	6	.0000	0:03:18	0:06:05
Humans 2	21,857	46	11	.0344	0:24:53	1:36:14

*Gray short-tailed opossum ** Random sample of 50 drawn from colony

- The inbreeding coefficient of an individual is identical to the kinship coefficient measured between that individual's parents.

- The coefficient of kinship calculated for an individual (that is, for *self*) is 1.0.

- It is sometimes useful to separate individuals or pairs for which coefficients are zero, from those whose coefficients are non-zero. This option is provided for in the following display that appears before execution of the program begins:

```
 1. Save all output records in file KINSHIP.OUT.
 2. Discard output records with zero coefficients.
 3. Save records with non-zero coefficients in KINSHIP.OUT, zero
    coefficients in NOKINSHP.OUT

Enter number corresponding to choice
>1
```

- To conserve computing time, KINSHIP skips calculations and assigns a zero coefficient of kinship between two individuals if they do not belong to the same pedigree, that is, if they do not have the same pedigree number. KINSHIP checks to see if the mnemonic PEDNO appears in the pedigree Code File, indicating that the file has been processed by program COUNTPED. If the mnemonic is present, the following query appears as a reminder that COUNTPED output may not be valid if records have been added or removed from the pedigree file after the program has been run:

```
Execution speed can be increased by limiting calculations to pairs belonging
to the same pedigree. This should be done only if records have not been add-
ed or removed from the Master File after program COUNTPED was last run,
and current pedigree identification numbers (PEDNOs) are certain to be valid.

Enter "L" to limit calculations to pairs in the same pedigree using current
PEDNOs, RETURN to calculate kinship without regard to PEDNO.
>
```

The advantage of this restriction is illustrated by computation of kinship coefficients between opposite-sexed pairs in the Humans 2 population listed in the table above. Calculations for 13806 of these pairs using the Stevens-Boyce method took 7:00 minutes. Using PEDNO to identify pairs in which both individuals were members of the same pedigree limited calculations to only 478 pairs, which took 2:56 minutes.

Algorithms: A. J. Boyce

Reference:

Boyce AJ 1983 Computation of inbreeding and kinship coefficients on extended pedigrees. *Journal of Heredity* 74:400-404.

Files:

Input: Any Master File or indexed pedigree file and Master Pointer File
Input Data File or console keyboard.

Output: **kinship.out** (inbreeding) contains Sequential and Permanent IDs of Ego, Ego's parents, and Ego's inbreeding coefficient.

kinship.out (kinship) contains Sequential and Permanent IDs of the two individuals for which kinship was calculated, and the kinship coefficient measured between them.

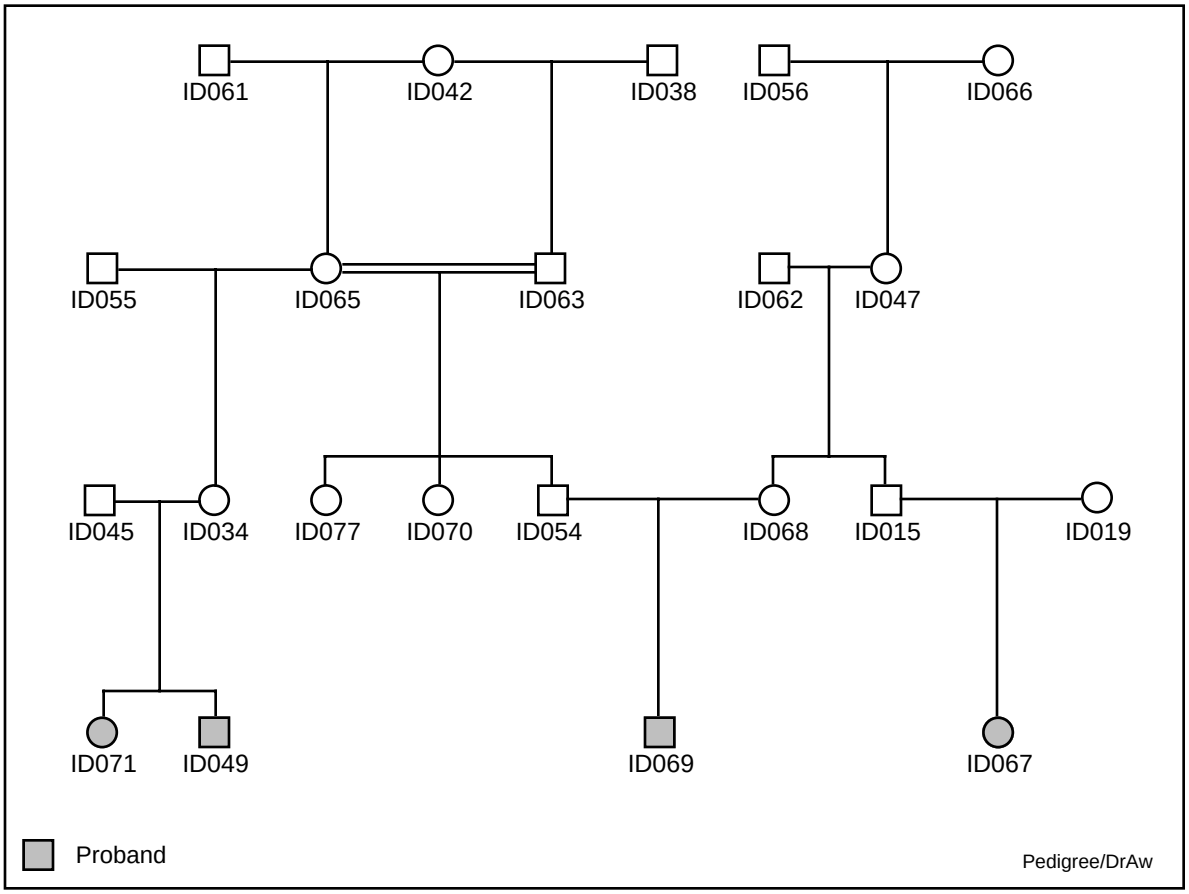
kinship.cde defines the record structure of **kinship.out**

nokinshp.out is an optional output file with the same format as **kinship.out**, but containing records for individuals or pairs with zero coefficients.

nokinshp.cde defines the record structure of **nokinshp.out**

kinship.err contains IDs of individuals for which calculations are requested, but which were not found in the pedigree file.

The pedigree diagram below shows these individuals and their relationships:



The following table gives all of the combinations specified by choices **1** - **3** in the pair composition menu, and identifies the respective pairs in the pedigree for which calculations are done:

Menu Choice	Pair combination	Pairs for which kinship coefficients computed
1	All opposite-sexed pairs	ID071-ID049 ID071-ID069 ID049-ID067 ID069-ID067
2	All same-sexed pairs	ID071-ID067 ID049-ID069
3	All pairs	ID071-ID049 ID071-ID069 ID071-ID067 ID049-ID069 ID049-ID067 ID069-ID067

LIST

Program LIST produces listings in a standardized format of any file for which an appropriate Code File has been constructed. Designation and setup of printers for PEDSYS listings are controlled by the Device Control File **DEVICES** (see Appendix A).

Introductory Display:

```
PEDSYS program LIST - Copyright 1992, 1999 SFBR

LIST produces listings in a standardized format of any file for which an
appropriate Code File has been constructed.

  1.  Unlinked File
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>
```

Because LIST does much of its formatting to conform to a particular output device, the first action required is to select one of a series of output devices (specified in file **DEVICES**). After a file has been selected whose contents are to be listed, the following display appears:

```
Output from LIST may be sent to the following output devices:

    [1]  Main Lab LaserWriterII
     2.  Main Lab Line Printer
     3.  Printer Room Sun LaserWriter
     4.  Console Screen
     5.  Output file "list.tab"

Enter number of device, or "Q" to quit without printing.
>
```

Next, items to be listed are selected:

```
   1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ   16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE   12 EGO Permanent ID       19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE   13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  LIST        Items to be listed              File - MASTER.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to items shown above in the order they are to
appear across the page. Precede the number with ">" or "<" for RIGHT or LEFT
justification. Enter "A" to list all items, "B" to insert blanks, [C] to
continue to next step.
>
```

Commands:

Item number Selection of items within records is accomplished by entering <n>, the number corresponding to the item in the list above the instruction line, in the order that items are to appear in the output listing.

><*n*> *or* **<<*n*>** where <n> is an item number as described above. Entry of this sequence right or left justifies the character string in the field. If the justification feature is used, the usual asterisk marking items chosen for inclusion will be replaced by directional markers: **>** marks items to be right justified, **<** marks items to be left justified.

B A field of blanks may be entered with this command, which produces a request for entry of field width, item label and mnemonics. Data type is automatically set at **C**.

K This command inserts a field containing a fixed string of characters. The string must be surrounded by single or double quotes, and its entry followed with entry of an item label and mnemonics. Data Type is automatically set at **C**.

Output Width Adjustment:

Maximum printer width (in characters) is set in the **DEVICES** file. If the width of the output record selected exceeds this maximum, the following display appears :

```
The total width of the items chosen exceeds the record length of the file.
The limit is 133 characters.

The selected items when formatted, add to 138 characters.
Please delete one or more items.

Press RETURN to continue.
>
```

Deletion of items is done with the standard record editing cycle (see Appendix A. SETTING DIRECTORIES and RUNNING PEDSYS PROGRAMS).

Requests for additional format information are displayed before the program is run:

```
 [1] LIST - conventional single-spaced, continuous print-out
  2. LINELIST - single-spaced groups separated by blank line
  3. DASHLIST - single-spaced groups separated by dashed line
  4. PAGELIST - single-spaced groups starting on new page
  5. FORMS-3 - continuous, three spaces per record
  6. FORMS-5 - continuous, five spaces per record
  7. PAGEFORM-3 - three spaces per record, groups start on new page
  8. PAGEFORM-5 - five spaces per record, groups start on new page

Select number corresponding to format of listing.
>
```

If grouped listings are selected (choices 2,3,4,7 or 8 above), the Item List is again displayed:

```
   1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ   16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE   12 EGO Permanent ID       19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE   13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  LIST        Item(s) to define groups        File - MASTER.XMP
--------------------------------------------------------------------------------
Enter number of one or more items to make up a key that defines record
groups (assumes file has been sorted on this key). Enter [C] to continue
to next step.
>
```

If PAGELIST (Choice 4) has been selected, the following display appears which permits further subgrouping of records, separated by dashed lines within pages:

```
Define subgroups on page (will be separated by dashed lines)?  [Y]/N
>
```

This is followed by another Item List Display similar to the one from which page grouping key items were selected.

Note that the grouping feature requires that the file be sorted by one or more key items by which groups are defined.

Once items (and groups) have been selected, the next instruction appears:

```
Enter number of copies to be printed: [1]-5, or "S" to select range of
records to be printed.
>
```

It is not necessary to list all records in the file, and printed output may be restricted to a specified range of records. If **S** is chosen, the following display is shown:

```
Enter range of records to be printed [1-32].
>1-10


Enter number of copies to be printed: [1]-5
>
```

Notes:

- LIST prints page and column headings at the top of each page. The page heading includes the current date, file title and listing page number. The current date is taken from the computer's hardware clock, and the file title is usually taken from the Code File Label Record. However, if files **sort.out** or **subset.out** are listed, the file title is derived additionally from file **SORTIN.TMP** or **SUBIN.TMP**, respectively (see Temporary System Files, Appendix A). This makes it possible to include information about processing history in the page heading. Examples of page headings might be:

```
"Example Master File"
"SORTED Example Master File"
"SUBSET OF reformat.out"
"SORTED SUBSET OF Blood Inventory File"
```

- Column headings are numbered, and consist of one to four rows of mnemonics taken from appropriate Code File Descriptor Records.

- Output directed to disk file **list.tab** is formatted as if it were to be printed except that control characters normally sent to the printer are not preserved in the file. Record length of the output file is fixed at 133 characters.

- When LIST output is directed to the console screen, record length and height of screen output is determined by the console setting in the **DEVICES** file. Since console width is limited, it is usually necessary to delete items before output can be displayed on the screen. Consecutive screen pages are displayed each time a carriage return is entered from the keyboard.

```
Example Master File                                    Top record: 1/15

  (1 )  (2 )  (3 )  (4 )  (5 )  (6 )  (7 ) (8 ) (9 )   (10)   (11)    (12)
  SEQ   SSEQ  DSEQ  KID1  PSIB  MSIB  FSIB NKID PSIBSZ MSIBSZ FSIBSZ  ID

    1     0     0    12     0     0     0   1     0      0      0   ID061
    2     0     0    11     0     0     0   1     0      0      0   ID056
    3     0     0    10     0     0     0   1     0      0      0   ID038
    4     0     0    10     0     0     0   2     0      0      0   ID042
    5     0     0    11     0     0     0   1     0      0      0   ID066
    6     0     0    13     0     0     0   1     0      0      0   ID055
    7     0     0    14     0     0     0   2     0      0      0   ID062
    8     0     0    20     0     0     0   2     0      0      0   ID045
    9     0     0    19     0     0     0   1     0      0      0   ID019
   10     3     4    16     0    12     0   3     1      2      1   ID063
   11     2     5    14     0     0     0   2     1      1      1   ID047
   12     1     4    16     0    10     0   4     1      2      1   ID065
   13     6    12    20     0    17     0   2     1      4      1   ID034
   14     7    11    19    15    15    15   1     2      2      2   ID015
   15     7    11    22    14    14    14   1     2      2      2   ID068

Enter RETURN for next page, number (1-32) to set top record, "Q" to quit.
>
```

The starting record of the page sequence may be reset by entering the appropriate record number prior to the RETURN, as indicated by the instructions at the bottom of the screen shown above.

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **list subset.out**.

Format options:

- Program LIST permits records with a common attribute to be printed in distinct groups. For example, all animals having the same dam may be printed contiguously, separated from other maternal half sibships by a blank line, a dashed line, or a page boundary. This requires that the file be sorted by DAM Permanent ID or DAM Sequential ID prior to running LIST.

- The Forms option produces "boxes" for handwritten entry of data or comments by enclosing records and item entries with dashed lines. This format may be combined with the sorted group option described above.

- The number of items that will fit across a page is limited by paper width, printer type size, and item field widths. The approximate number of columns available is the product of nominal paper width (typically 8 or 14 inches) and width of printed characters (1/10, 1/12, 1/15, 1/16.6 inch, etc.). LIST computes the number of characters required to print a record from information taken from the Code File. Specifically, this is the sum of the field length parameter, or the width of largest mnemonic (minimum 4, maximum 6 characters), whichever is longer, plus a single blank space per item, for all items selected for listing. This number is then compared with the printer line length parameters in file **DEVICES** (see The Device Control File, Appendix A.) to determine the largest type size that can be used to print the full line. If the number of characters is too great to be printed on a single line using the smallest available font, a warning message appears with a request to reduce the number of items to be printed (see Output Width Adjustment, above).

- The number of lines that can be printed on a single page is variable, but we have found that 50 records on an 11-inch-long page in continuous, single-spaced format is convenient. With number of records set at 50 in the **DEVICES** file, FORMS-3 format gives 17 records per page, and the larger FORMS-5 boxes permit 11 records per page.

Files:

Input: Any Master File or Data File, and an associated Code File.
The Device Control File, **DEVICES**.

Output: Printer, console screen, or output file **list.tab**. Records selected from the input file will be printed in their original order.

MAKEPED

Program MAKEPED constructs pedigrees connecting one or more individuals with their relatives. The program constructs pedigrees in two ways: (a) by assembling ascending and/or descending genealogical trees of specified depth for each proband selected, or (b) assembling family units (parents, full sibs, mates and offspring) for a single proband and for specified relatives. The Output File generated by both methods is ordered by Permanent ID, and duplicates are eliminated so that individuals appear only once, which may merge pedigrees when multiple probands are selected. The file from which pedigrees are generated may be either a Master File, or an indexed pedigree file consisting of records containing IDs for Ego and Ego's parents, Sex, etc. Pedigrees may be constructed for probands entered from the keyboard, or from a Proband Input File (consisting of a list of either Sequential or Permanent IDs).

Introductory Display:

```
PEDSYS program MAKEPED - Copyright 1992, 1999 SFBR

MAKEPED constructs pedigrees (a) by extending genealogical links upward
and/or downward from one or more probands, or (b) by starting with a single
proband's nuclear family and extending successive links both vertically and
laterally to relatives of the proband.

IMPORTANT: The file from which the pedigree is generated must be an indexed
           pedigree file with an associated Master Pointer File.

 1.  Unlinked File
 2.  index.out
 3.  Example Master File

Enter number, file name, or "Q" to quit.
>
```

If an unlinked file is used (choice **1** above), it must contain PIDs of Ego and Ego's parents, and Ego's sex. This file must then be used as input to program INDEX with the option to create a Master Pointer File selected. When an unlinked file has been selected, the program compares its Code File with entries in **CODE.STD**, for standard mnemonics **EGO** (or **ID**), **FA** (or **SIRE**), **MO** (or **DAM**), and **SEX**. If the Code File contains these mnemonics, the following instruction appears:

```
Use of an unlinked pedigree file requires that the file be in Master File
format; that is, it must be indexed and have associated with it a Code File
and a Master Pointer File (such as INDEX.MPT)

Enter the name of the pedigree file to use.
>index.out

Enter the name of the code file for index.out, or enter RETURN for index.cde.
>

Enter the name of the Master Pointer File or enter RETURN for index.mpt.
>
```

Failure to find a Master Pointer File results in the option to enter another file name or to return to the Initial File List display. When a Master File or indexed pedigree file with an appropriate Master Pointer File has been selected, an Item List Display appears, from which items may be selected for inclusion in the output files generated by the program. Once this is done, the following display appears, from which the method of constructing pedigrees is selected:

```
MAKEPED will construct pedigrees in two ways:

Method 1 - Ascending and/or descending genealogical trees of specified
depth are assembled for each proband. Optionally, probands' full sibs
and/or mates may be included.

Method 2 - Initially, a nuclear family unit (parents, full sibs, mates and
offspring) is assembled for a single proband.  In subsequent steps family
units are formed for each relative of the proband newly added in the
previous step.

The Output File generated by both methods is ordered by Permanent ID, and
duplicates are eliminated so that individuals appear only once, which may
merge pedigrees when multiple probands are selected.

   1. METHOD 1 (Ascending/descending trees; 1-500 probands)
   2. METHOD 2 (Assembly of family units; Single proband)
   Q. QUIT without executing program

Enter number above corresponding to choice.
>
```

Method 1

If Method **1** has been selected, the following display appears from which the source of proband IDs may be chosen:

```
Pedigrees may be constructed for 1-500 probands entered from

  1.  Keyboard
  2.  Unlinked File
  3.  EXAMPLE
  4.    LOCUS Sub-directory

Enter number above corresponding to the appropriate source of probands.
>1
```

Choice 1: Proband IDs entered from the keyboard.

Choice 1 (shown above) specifies that proband IDs are to be entered from the keyboard. Before keyboard entry begins, the form of the IDs (Permanent or Sequential) must be specified:

```
Enter - [P] if probands are identified by Permanent ID
         S  if by Sequential ID
>p
```

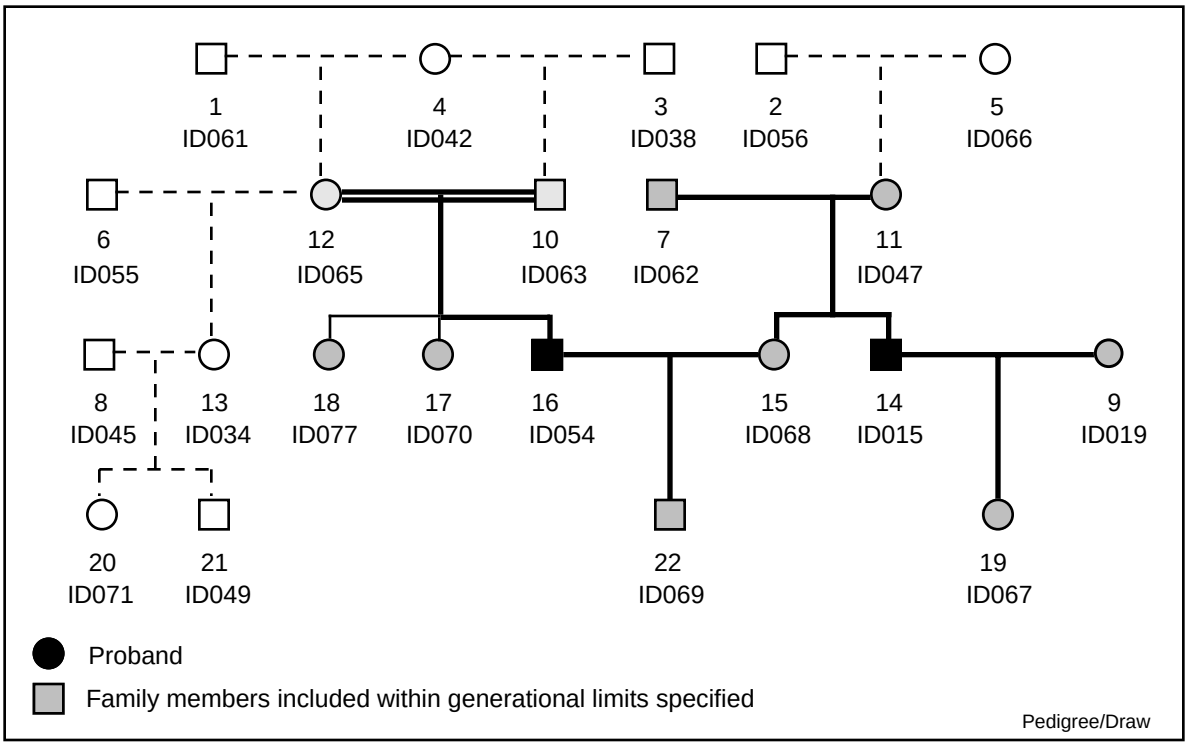
Here, Permanent IDs have been selected. Next, limits must be set on the number of ascending generations (starting from each proband) from which common ancestors will be sought:

```
Output File will include direct ancestors, and both parents of direct
descendants of each proband (including probands' mates).  Output File may
contain more than one pedigree if multiple probands are selected.  Limits of
genealogical depth set below apply independently to each proband selected.

Enter number of ASCENDING generations of probands' kin to be included in
pedigree ([0]-20).
>1

Enter number of DESCENDING generations of probands' kin to be included in
pedigree ([0]-20).
>1

Include probands' FULL SIBS?  [Y]/N
>n

Include MATES?  [Y]/N
>y
```

Here, sibs have been excluded, with mates included in single ascending and descending generations from the probands have been selected. Next, proband IDs are entered:

```
 Enter proband's Permanent ID.  (RETURN alone terminates entry.)

>ID054< ID
>ID015< ID
>     < ID
```

The diagram below shows probands ID054 and ID015 and their relatives included in the pedigree:



Choice 2, 3 or 4: Proband IDs read from a Proband Input File.

A proband file is often a subset of the input pedigree file (typically a Master File), but may be any Data File whose records contain IDs in the appropriate form (that is, Permanent or Sequential). Selection of a single item containing the proband ID is made from a Standard Item List Display:

```
   1 EGO Permanent ID        5 Sex (M, F or U)         9 PEDIGREE NUMBER
   2 Birth Date (YYYYMMDD    6 Exit Date (YYYYMMDD)   10 GENERATION NUMBER
   3 Sire's Permanent ID     7 Exit Code
   4 Dam's Permanent ID      8 Mate's Permanent ID
--------------------------------------------------------------------------------
  MAKEPED     Identify PROBAND                File - subset.out
--------------------------------------------------------------------------------
Enter a number above corresponding to Proband ID, or "C" to continue
to next step.
>8
```

Here, the pedigrees of the mates of individuals found in file **subset.out** have been specified, and the program proceeds to selection generational depth, etc.

Method 2

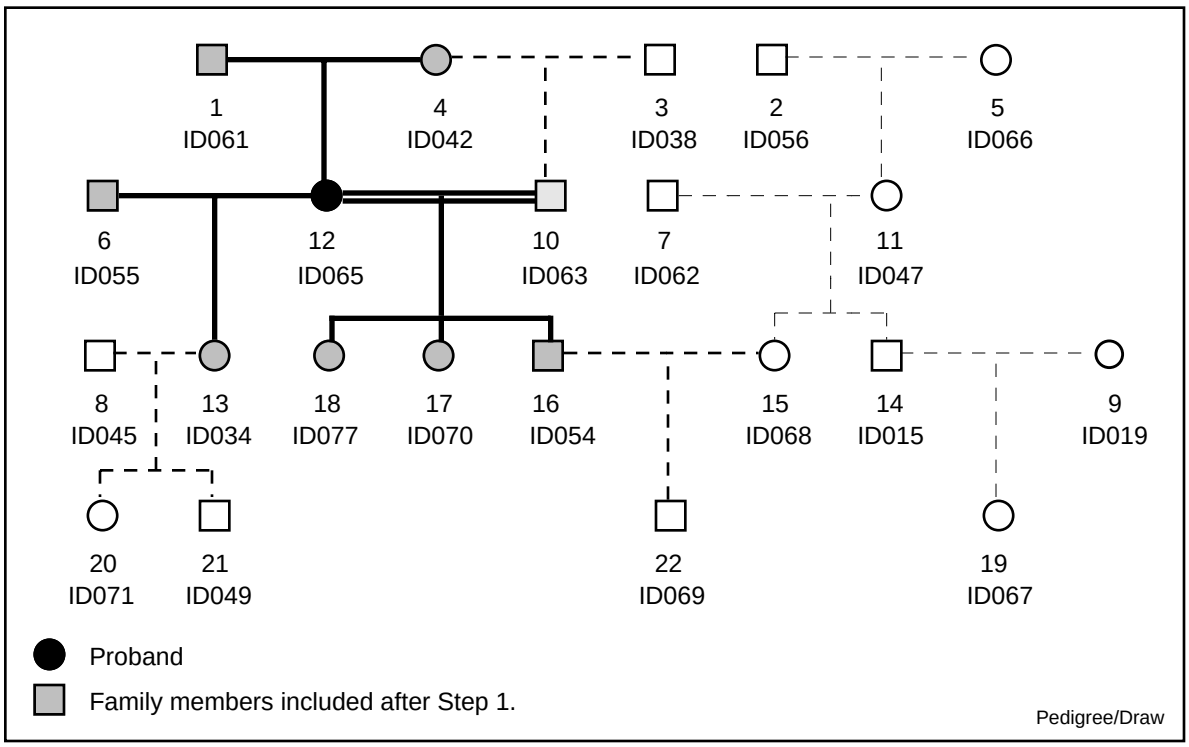
Method 2 begins with the following sequence:

```
Enter - [P] if probands are identified by Permanent ID
         S  if by Sequential ID
>

Enter number of steps in assembly of family units ([1]-6), or enter "H" for an
explanation of the process.
>1

Enter proband's Permanent ID.

>ID065< ID
```

First, the form of the IDs (Permanent or Sequential) used to identify pedigree members must be specified. This is followed by a request for the number of "steps" to extend the pedigree outward from the proband, and then for the ID of the proband. Once a proband ID has been entered, Method 1 proceeds as follows:

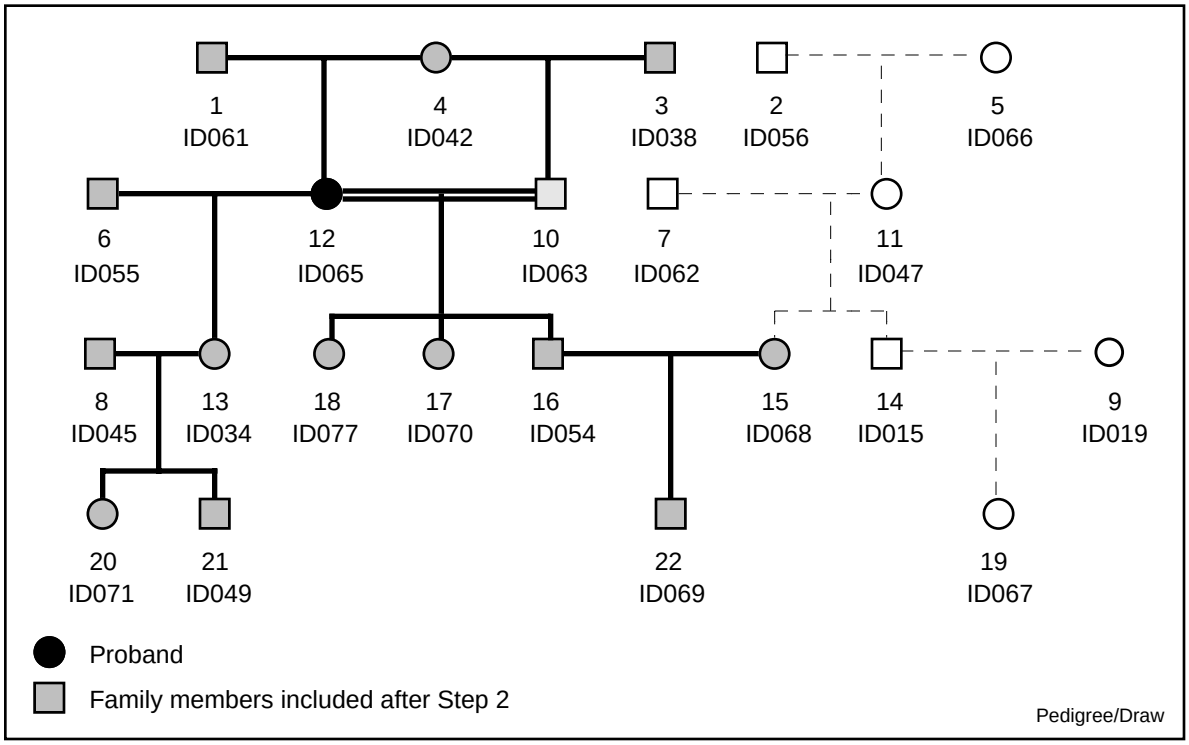
Step 1. A nuclear family unit consisting of the proband, the proband's parents, full sibs, one or more mates, and all offspring. This family unit forms the basis of a pedigree stored in the Output File, and its members (excluding the proband and his or her mates) are used as input to subsequent steps, if specified. Mates are included in the unit, but are not used as the basis for successive steps of family unit construction. In the absence of inbreeding, kin identified in Step 1 are related to the proband with a coefficient of kinship $\phi = 0.25$.

The diagram below shows relatives of proband ID065 included in the pedigree after Step 1:



Step 2. If this step is specified, family units are constructed for each of the proband's parents, sibs and offspring, and individuals not previously identified in Step 1 are added to the pedigree. Kin identified in this step (grandparents, half sibs, aunts/uncles, nieces/nephews and grandchildren) are related to the proband with $\phi = 0.125$.

The diagram below shows relatives of proband **ID065** included in the pedigree after Step 2:



Step 3. In this and subsequent steps (if specified), family units are formed for each relative newly identified in the previous step, etc. In general, ϕ measured between the proband and kin newly identified in each step will be equal to (1/2)**(STEP + 1), providing the pedigree is not inbred.

Files:

Input: Any Master File or indexed pedigree file and Master Pointer File
Input Data File or console keyboard.

Output: **makeped.out** contains one or more pedigrees connecting probands to their relatives according to limits specified on input.

makeped.ego contains records identifying probands for which pedigrees were constructed.

makeped.cde defines the record structure of **makeped.out** and **makeped.ego**.

makeped.err contains (a) proband IDs entered from an input file that are not found in the input pedigree file, and (b) IDs of individuals whose sex is unknown.

MERGE

Program MERGE assembles single records for individuals from information kept in two separate Data Files. The principal output file is created by adding to records in a **Target file** (the first file accessed) selected items copied from records in a **Donor file** (the second file accessed).

Introductory Display:

```
PEDSYS program MERGE - Copyright 1992, 1999 SFBR

MERGE assembles single records for individuals from information kept in two
separate Data Files.

  1.  Unlinked File
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter number or name of file (TO which data will be added), "Q" to quit.
>2
```

A Standard File List appears next, with an instruction for selecting the Target file:

```
    1.  Example Master File
    2.  Markers + Quant. P'types
    3.  Blood Sample Inventory
    4.  Hemoglobin Levels

Enter number of TARGET file (TO which data will be added).
>1
```

Here, the Example Master File has been selected. Note that the Target file need not be a Master File -- in fact, records from any two files may be merged, so long as they have the same Merge Key Item format (see **Selecting Key Items**, below).

The procedure for selecting the Donor file follows the same pattern as described above:

```
    1.  Example Master File
    2.  Markers + Quant. P'types
    3.  Blood Sample Inventory
    4.  Hemoglobin Levels

Enter number of DONOR file (FROM which data will be copied).
>2
```

In this example, the **Markers + Quant. P'types** file will be merged to the Example Master File. Selection of Key Items that govern matching of records in the two files is done next.

Selecting Key Items:

Records are merged only when each contains an identical **Merge Key**. Merge Keys consist of one or more Data Items selected from Standard Item Lists that define records of both the Target and the Donor files. The Target file Item List appears first:

```
   1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ   16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE  *12 EGO Permanent ID       19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE   13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  MERGE                                TARGET File - MASTER.XMP
--------------------------------------------------------------------------------
Enter number(s) identifying key item(s) on which merge will be based. Enter
[C] to continue to next step.
>
```

In the example above, a single Key Item (Ego PID) has been selected. The same Key Item is next selected for the Donor file:

```
 * 1 Ego ID                  3 APOB Pvu2a Genotype     5 APOAI Level
   2 ADA  Phenotype          4 Tot. Ser. Cholestero
--------------------------------------------------------------------------------
  MERGE                               DONOR  File - QUANTGEN.XMP
--------------------------------------------------------------------------------
Enter number identifying key item(s) on which merge will be based. Enter
[C] to continue to next step.
>1
```

A confirmation step follows:

```
Merge will be based on

EGO Permanent ID      in MASTER.XMP (TARGET file) and
Ego ID                in QUANTGEN.XMP (DONOR file).

Is this correct?  [Y]/N
>
```

If, however, Key Item formats do not match, the following message appears:

```
Key Items from MASTER.XMP and QUANTGEN.XMP do not have the same format. (For
details, compare formats in CODE.XMP and QUANTGEN.CDE.)

Enter RETURN to re-select Key Items.  Enter "Q" to quit program.
>
```

If re-selection of Key Items is chosen, the program returns to the Item List Display shown above.

The Merge Key may be made up of more than one Data Item, in which case the confirmation display appears as follows:

```
Merge will be based on

MASTER.XMP (TARGET file):
   FIRST OFFSPRING SEQ
   Sex (M, F or U)
MASTER.XMP (DONOR file):
   EGO SEQUENTIAL ID
   Sex (M, F or U)

Is this correct?  [Y]/N
>
```

This example also shows that it is possible for the same file to be both Target and Donor. This particular merge produces records containing information about parents and their same-sexed first offspring on the same record.

Note also that the Donor file need not be a Linked Data File; records from any two files may be merged, so long as they have the same Key Item format (see **Selecting Key Items** below). If the Donor file is not linked (as in the example displayed below), the following query appears:

```
Does examped (DONOR file) contain more than one record for the
same Key Item(s)?  Y/[N]
>
```

If the answer is **Y**, the following message is displayed:

```
WARNING! There should be only one record in examped (DONOR file) for each
key in MASTER.XMP; otherwise the record selected for merging is unpredictable.
Switching order of MASTER.XMP and examped may be helpful in this case. Enter
"Q" to exit program, RETURN to continue to next step.

>
```

Although this is not a fatal error, it is usually best to exit the program and switch the order of files to be merged in this case.

Selecting Items to be Included in Merged Records:

If Key Items are correct (that is, are identical in width and Data Type), items to be included in the merged record are selected from Standard Item Lists, first for the Target file:

```
 * 1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING   *15 Dam's Permanent ID
 * 2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ  *16 Sex (M, F or U)
 * 3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE  *12 EGO Permanent ID        19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE  *13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ     *14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  MERGE       Items from TARGET file   TARGET File - MASTER.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to items shown above in the order they are to
be included in merged record. Enter "A" to list all items, [C] to continue
to next step.
>
```

and then for the Donor file:

```
   1 Ego ID                * 3 APOB Pvu2a Genotype   * 5 APOAI Level
 * 2 ADA  Phenotype        * 4 Tot. Ser. Cholestero
--------------------------------------------------------------------------------
  MERGE       Items from DONOR file    DONOR  File - QUANTGEN.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to items shown above in the order they are to
be appended to merged record. Enter "A" to list all items, [C] to continue
to next step.
>
```

After Standard Confirmation, two requests for additional information are displayed in sequence before the program is run:

```
Include records from MASTER.XMP (TARGET file) that have no
corresponding entry in QUANTGEN.XMP (DONOR file)?  [Y]/N
>Y

In addition to saving merged records in file merge.out,

1. Save records from MASTER.XMP (TARGET file) that have
   no corresponding entry in QUANTGEN.XMP (DONOR file).
   These records will be placed in file nomerge1.out.
2. Save records from QUANTGEN.XMP (DONOR file) that have
   no corresponding entry in MASTER.XMP (TARGET file).
   These records will be placed in file nomerge2.out.
3. Save unmatched records as described in both 1. and 2. above.

Enter number above corresponding to choice, or press RETURN to continue
to next step.
>
```

The first query controls the content of output file **merge.out**. The second makes it possible to save records from either the Target or the Donor files that cannot be merged, that is, records that do not share Key Items. The next display appears briefly before merging begins:

```
Merged records will contain the following items:

From MASTER.XMP (TARGET file):
   1 EGO SEQUENTIAL ID       4 EGO Permanent ID        7 Dam's Permanent ID
   2 SIRE'S SEQ              5 Birth Date (YYYYMMDD    8 Sex (M, F or U)
   3 DAM'S SEQ               6 Sire's Permanent ID
--------------------------------------------------------------------------------
From QUANTGEN.XMP (DONOR file):
   1 ADA  Phenotype          3 Tot. Ser. Cholestero
   2 APOB Pvu2a Genotype     4 APOAI Level
--------------------------------------------------------------------------------
```

Notes:

- Program MERGE operates by reading records of the Target file one by one and for each record, searching through the Donor file for a record with a matching Key Item. The searching algorithm (a binary search) requires that the Donor file be sorted by Key Item. The program does this sort automatically.

- A merged record contains information from no more than one record from each of the contributing files.

- The number of records in MERGE.OUT is always less than or equal to the number of records in the Target file.

- The outcome of merging linked and unlinked Single-entry files is the same. However, differences occur in the case of Multiple-entry files: A *linked* Multiple-entry Donor file will merge EGO's most recently entered record with records in the Target file, whereas the particular record selected for merging from an *unlinked* Multiple-entry Donor file is unpredictable. Items selected from records in a Donor file will be repeatedly merged with each of Ego's records that appear in a Multiple-entry Target file, whether the latter is linked or unlinked. In most cases it will be best to define an unlinked Multiple-entry file as the Target, rather than the Donor file.

- The command line shortcut with *two* input files (program name, followed by a blank, the Target filename, another blank and the Donor filename) may be used to start this program. Example: **merge sort.out subset.out**.

Files:

Input: Any two PEDSYS files having associated Code Files. Recursive use of output files is permitted, that is, file **merge.out**, **nomerge1.out** or **nomerge2.out** may be used as input to program MERGE.

Output: **merge.out** contains records in which items have been assembled from both Target and Donor files. Optionally, the output file may also contain records from the Target file for which no entries were found in the Donor file.

merge.cde defines the record structure of **merge.out**.

nomerge1.out contains records from the Target file that have no corresponding entry in the Donor file. The Key Item is added automatically to records in this file.

nomerge2.out contains records from the Donor file that have no corresponding entry in the Target file. The Key Item is added automatically to records in this file.

nomerge1.cde and **nomerge2.cde** define the record structure of **nomerge1.out** and **nomerge2.out,** respectively.

Program limits:

Records in Target and Donor files must contain identical Key Items for a merge to occur.

The maximum field width of the Key Item is 99 characters. The maximum number of records that may be merged is 60,000 for Unix and the Macintosh, 8,000 for MS-DOS.

NEWITEM

Program NEWITEM makes changes in selected items of data records as specified in a conversion table. Changes in one item may be specified so that they are either independent of, or dependent on values found in other items on the same record. A previously created table may be used, or a new table may be entered from the keyboard.

Introductory Display:

```
PEDSYS program NEWITEM - Copyright 1995, 1999 SFBR

NEWITEM makes changes in selected items of data records as specified in a
conversion table.

  1.  Unlinked File
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>
```

When the Master File or Data File containing items to be converted has been selected, the following display appears:

```
Existing items may be changed, and/or new items may be created and added
to the end of each record.  Enter "N" to create new items, or RETURN to
continue without creating new items.
>n
```

If one or more items is to be created, the following sequence appears (once for each new item):

```
Enter item to right of pointer (>), enter RETURN to advance cursor, or "!"
to abort and re-enter item.  Enter RETURN at first position when done.

>Kids/Sibs Ratio      <- Label 22
>KidSib<- Mnemonic 1
>Ratio <- Mnemonic 2
>      <- Mnemonic 3
>      <- Mnemonic 4
>6 <- Width
>r<- Data type
>2<- Places to right of decimal point
```

Next, the source of conversion data is selected:

```
A previously created conversion table may be used, or a new table may be
entered from the keyboard.

Enter - [S] to START a new Conversion Table
         T  to USE an existing Conversion Table
         Q  to QUIT program.
>
```

STARTING a new Conversion Table:

When the START option is selected a Standard Item List appears:

```
   1 EGO SEQUENTIAL ID       9 PATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   2 SIRE'S SEQ             10 MATERNAL SIBSHIP SIZ   18 Exit Code
   3 DAM'S SEQ              11 FULL SIBSHIP SIZE      19 Mate's Permanent ID
   4 FIRST OFFSPRING SEQ    12 EGO Permanent ID       20 PEDIGREE NUMBER
   5 NEXT PATERNAL SIB SE   13 Birth Date (YYYYMMDD   21 GENERATION NUMBER
   6 NEXT MATERNAL SIB SE   14 Sire's Permanent ID    22 Kids/Sibs Ratio
   7 NEXT FULL SIB SEQ      15 Dam's Permanent ID
   8 NUMBER OF OFFSPRING   *16 Sex (M, F or U)
--------------------------------------------------------------------------------
  NEWITEM     Items to be converted    MASTER.XMP
--------------------------------------------------------------------------------
After IF/AND/OR:  Enter <Item No.> <Operator> <Test>.
      THEN/ELSE:  Enter <Item No.> = <Change> (may be compound).
EXAMPLES:     IF  > 16 eq "M"     AND > 8 lt [9,10,11]     OR  > 21 ne
              THEN> 18 = k3       THEN> 22 = 8 / 11        IF  > 13 = 75*
Enter "H" for explanation, shortcuts, examples; RETURN alone to continue
to next step.

Last entry: IF 16 EQ 'F' THEN 16 = '2'
IF  > 16 = 'M'
AND >
OR  >
THEN> 16 = '1'
ELSE>
```

This example shows an entry sequence required to make changes to SEX coding independently of values found in other items on the record (conversion from **M**, **F** and **U** to **1**, **2** and **3**, respectively). Item 16 has been marked with an asterisk , indicating that at least one Conversion Table entry has been made for SEX. The most recent of these, specifying the change from '**F**' to '**2**', is shown in the line labeled **Last entry:**, just above the five command lines at the bottom of the screen display. Entries have been made in the command sequence that will specify the change of '**M**' to '**1**'. These will be transferred to the **Last entry** line when entry of this conversion is done, as can be seen below:

```
   1 EGO SEQUENTIAL ID       9 PATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   2 SIRE'S SEQ             10 MATERNAL SIBSHIP SIZ   18 Exit Code
   3 DAM'S SEQ              11 FULL SIBSHIP SIZE      19 Mate's Permanent ID
   4 FIRST OFFSPRING SEQ    12 EGO Permanent ID       20 PEDIGREE NUMBER
   5 NEXT PATERNAL SIB SE   13 Birth Date (YYYYMMDD   21 GENERATION NUMBER
   6 NEXT MATERNAL SIB SE   14 Sire's Permanent ID    22 Kids/Sibs Ratio
   7 NEXT FULL SIB SEQ      15 Dam's Permanent ID
   8 NUMBER OF OFFSPRING   *16 Sex (M, F or U)
--------------------------------------------------------------------------------
  NEWITEM     Items to be converted    MASTER.XMP
--------------------------------------------------------------------------------
After IF/AND/OR:  Enter <Item No.> <Operator> <Test>.
      THEN/ELSE:  Enter <Ttem No.> = <Change> (may be compound).
EXAMPLES:     IF  > 16 eq "M"     AND > 8 lt [9,10,11]     OR  > 21 ne
              THEN> 18 = k3       THEN> 22 = 8 / 11        IF  > 13 = 75*
Enter "H" for explanation, shortcuts, examples; RETURN alone to continue
to next step.

Last entry: IF 16 EQ 'M' THEN 16 = '1'
IF  > 11 <> k0
AND >
OR  >
THEN> 22 = 8/11
ELSE> 22 = 8
```

Shown here is an example of a change to a compound value that is dependent on values of other items on the record. Item 22, **Kids/Sibs Ratio**, is a new item added prior to starting the table. The value of this item is dependent on the values of Item 8, **NUMBER OF OFFSPRING** and Item 11, **FULL SIBSHIP SIZE**. If Item 11 is non-zero, Item 22 is set to the value of Item 8 divided by the value of Item 11. Otherwise, Item 22 is set to the undivided value of Item 8.

When all items have been chosen for conversion, the following display appears, showing entries in the Conversion Table, and providing an opportunity to edit them:

```
 1. IF SEX EQ 'F' THEN SEX = '2'
 2. IF SEX EQ 'M' THEN SEX = '1'
 3. IF SEX EQ 'U' THEN SEX = '3'
 4. IF FSIBSZ NE K0 THEN KidSib Ratio = NKID / FSIBSZ ELSE KidSib Ratio =
    NKID
--------------------------------------------------------------------------------
  NEWITEM     Conversion Table         newitem.tab
--------------------------------------------------------------------------------

Enter - Number of table entry above to EDIT
        A to ADD an entry to end of list
        D to DELETE an entry
        Q to QUIT program (with option to save Conversion Table)
       [C] to CONTINUE (Save table, execute program)
>3
```

Entry of a number corresponding to one of the Conversion Table entries (here entry **3**) results in the following EDIT screen display:

```
   1 EGO SEQUENTIAL ID      9 PATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   2 SIRE'S SEQ            10 MATERNAL SIBSHIP SIZ   18 Exit Code
   3 DAM'S SEQ             11 FULL SIBSHIP SIZE      19 Mate's Permanent ID
   4 FIRST OFFSPRING SEQ   12 EGO Permanent ID       20 PEDIGREE NUMBER
   5 NEXT PATERNAL SIB SE  13 Birth Date (YYYYMMDD   21 GENERATION NUMBER
   6 NEXT MATERNAL SIB SE  14 Sire's Permanent ID    22 Kids/Sibs Ratio
   7 NEXT FULL SIB SEQ     15 Dam's Permanent ID
   8 NUMBER OF OFFSPRING   16 Sex (M, F or U)
--------------------------------------------------------------------------------
  NEWITEM     EDIT Conversion Table    newitem.tab
--------------------------------------------------------------------------------
Re-enter Item Number-Operator-Test/Value sequence to right of pointer (>).
Enter one or more spaces to remove sequence, enter RETURN alone to advance
to next sequence, "]" to go to next step.

Last entry:
IF  > 16 EQ 'U'
AND >
OR  >
THEN> 16 = b
ELSE>
```

Here, the third Conversion Table entry will be modified so that the SEX code will be changed from **U** to **b** (blank) instead of **3**, as originally specified.

When the **ADD** function is selected from the previous screen, characteristics of the new item are specified as shown previously, after which the table entry sequence display appears. Entry of **D** results in the following instruction:

```
Enter number corresponding to entry to be deleted.
```

Exit from the program prior to data conversion is possible at this point. Command **S** creates and/or saves the Conversion Table in output file **newitem.tab**; command **Q** exits without execution. Entry of **C** results in program execution.

Using NEWITEM with an existing Conversion Table:

If option **T** has been chosen from the Introductory Display, the following request appears on the screen:

```
Enter name of file containing conversion table.
>
```

When name of this file has been entered, a Conversion Table display appears (with edit, delete, save, etc. commands) as described above.

Notes:

- **Shortcuts**:

Two shortcuts may be used to speed entry of NEWITEM command sequences. The first of these makes it possible to enter a full command sequence on a single line, rather than in segments following the > prompts at the bottom of the command entry screen. For example, entering the following sequence after the **IF** > prompt

```
IF  > 16 ne F and 16 ne U then 16 = '1'
AND >
OR  >
THEN>
ELSE>
```

is exactly equivalent to the following sequence:

```
IF  > 16 ne F
AND > 16 ne U
OR  >
THEN> 16 = '1'
ELSE>
```

NEWITEM recognizes the full sequence, skips the remaining prompts, marks the item to be changed, and waits for the next entry when RETURN is entered at the end of the line.

The second shortcut reduces the number of separate command sequences required for testing a single item against multiple test values. That is, multiple test values enclosed in square brackets and separated by commas may follow a logical operator. A single sequence may be written for this case:

```
IF 12 EQ [ID071,ID049,ID045,ID034]
```

When the initial operator is **EQ**, commas separating test values are interpreted as **OR** operators, so that the statement above is true when Item 12 (EGO PID) is the same as any one of the PIDs in the list. For any other initial operator, commas separating test values are interpreted as **AND** operators. Thus, the following statement

```
IF 8 LT [9,10,11]
```

is true when Item 8 (NUMBER OF OFFSPRING) is less than all the values of Items 9, 10 and 11 (PATERNAL, MATERNAL and FULL SIBSHIP SIZE, respectively).

This shortcut applies only to test values, that is, multiple item numbers may not be tested in this way. Data types of test items are determined by the data types listed in the input Code File.

The constant designator can also be used with multiple test values. For example:

```
IF 11 eq k[1,2,3]
```

and

```
IF 11 eq [9,k2]
```

are legitimate expressions.

Substring values of character data (input Data Type **C**) may be tested in multiples, as well:

```
IF 12 EQ [*071,*049,*045,*034]
```

Here, the numerical portion of EGO PID is tested, with the result (in this example data set) being the same as the sequence testing PIDs shown above. Substring searches are slow, and should be avoided when rapid output is needed.

- **Equivalences:**

As a convenience, several command elements have been given equivalent alternate forms. These are:

EQ = **=** (equal sign)	**GT** = **>**	**GE** = **>=** or **=>**
NE = **<>** or **><**	**LT** = **<**	**LE** = **<=** or **=<**
' ' = **" "** (must be matched pairs)		

- **Data Types:**

PEDSYS programs require information about data types of items in input files, defined by the data type codes listed in the input Code File. These definitions are used by many programs to determine how data are to be interpreted. However, since PEDSYS Data Files contain nothing but ASCII text, substring test values may be used whatever the Data Type specified in the input Code File. For example

```
if 13 eq [k19661223,1973*] then 17 = k19880116
```

Here, if Item 13 (Birth Date) is equal to the date **19661223** (= December 12, 1966), or to any date beginning with substring **1973** (= any day during the year 1973), Item 17 (Death Date) will be set to **19880116** (= January 16, 1988).

Data types of output items in **newitem.out** are determined by data types of the input file (that is, by the data type codes listed in the input Code File), but these defaults may be overridden by the data type of Conversion Table entries. Numerical items (types I, R or D) may be converted to character data, in which case a warning is issued. Entries for conversion of integer (type I) or real data (type R) in fields of the Conversion Table are automatically right-justified. Entries for character data (type C) are left-justified, although leading blanks are honored.

Data types of *items* referred to in the Conversion Table **newitem.tab** are determined by data types of items the input file, but values of test and change *values* must be defined unambiguously. The table below gives examples of command sequences and their interpretations:

Data Type D, I or R

Test	Interpretation
`IF > 11 eq 8`	Is Item 11 equal to Item 8 (whatever the numeric value)?
`IF > 11 eq [9,10]`	Is Item 11 equal to either Item 9 or Item 10?
`IF > 11 ne [9,10]`	Is Item 11 equal to neither Item 9 nor Item 10?
`IF > 11 lt k8`	Is Item 11 less than numeric constant 8?
`IF > 18 eq k[3,4]`	Is Item 18 equal numeric constant 3 or 4?
`IF > 18 eq b`	Is Item 18 entirely blank?
`IF > 18 eq`	Is Item 18 entirely blank?
`IF > 18 eq * *`	Does Item 18 contain a blank character in any field position?

Change	**Interpretation**
`THEN> 5 = 6`	Set Item 5 equal to Item 6 (whatever the numeric value).
`THEN> 5 = k6`	Set Item 5 equal to numeric constant 6.
`THEN> 18 = "Dth"`	Convert numeric item to character data (See notes a and b).
`THEN> 20 = '2'`	Set Item 20 equal to character 2 (See notes a and c).
`THEN> 19 = b`	Set Item 19 entirely blank.
`THEN> 19 =`	Set Item 19 entirely blank.
`THEN> 19 = ' '`	Set Item 19 entirely blank.

Data Type C

Test	**Interpretation**
`IF > 7 eq 21`	Is Item 7 identical to Item 21 (whatever the character string)?
`IF > 7 eq [5,6]`	Is Item 7 identical to Item 5 or Item 6?
`IF > 12 eq ID045`	Is Item 12 identical to the exact string ID045?
`IF > 12 eq 'ID045'`	Is Item 12 identical to the exact string ID045?
`IF > 12 eq ID04*`	Does Item 12 start with the sub-string ID04?
`IF > 12 eq *D045`	Does Item 12 end with the sub-string D045?
`IF > 12 eq *ID04*`	Does Item 12 contain the sub-string ID04 in any field position?
`IF > 18 eq *"*`	Does Item 18 contain a double quote in any field position?
`IF > 18 eq ***`	Does Item 18 contain an asterisk in any field position?
`IF > 21 eq b`	Is Item 21 entirely blank?
`IF > 21 eq`	Is Item 21 entirely blank?
`IF > 21 eq " "`	Is Item 21 entirely blank?
`IF > 21 eq *b*`	Does Item 21 contain the character b in any field position?
`IF > 21 eq * *`	Does Item 21 contain a blank character in any field position?

Change	**Interpretation**
`THEN> 14 = 21`	Set Item 14 equal to Item 21 (whatever the character string).
`THEN> 14 = ID015`	Set Item 14 equal to string ID015.
`THEN> 14 = "ID015"`	Set Item 14 equal to string ID015.
`THEN> 16 = " Fem "`	Set Item 16 equal to string Fem (see notes b and d).
`THEN> 12 = ID1136`	Set Item 12 equal to string ID1136 (see note b).
`THEN> 20 = b`	Set Item 20 entirely blank.
`THEN> 20 =`	Set Item 20 entirely blank.
`THEN> 20 = " "`	Set Item 20 entirely blank.

Notes:
a - Warning: Data type change applies to all entries for this item.
b - Warning: Field width of item will be increased.
c - Warning: All entries for this item will be left-justified.
d - Leading and trailing blanks honored.

- A dummy value **SKIP** is sometimes helpful in constructing compound sequences. For example:

```
if 16 eq 'M' or 16 eq 'U' then skip else 16 = 'Fem'
```

  would change the female Sex Code from **F** to **Fem**, leaving males and unknown sex codes in their original form.

- NEWITEM performs its tests in order and makes no permanent changes until all tests have been completed for each input data record. This means that a command sequence entered at the top of the list in the Conversion Table may be overridden by a sequence entered later in the list. For example, the following pair of command sequences

```
if 16 eq 'M' then 16 = k1
if 16 eq 'F' then 16 = k2 else 16 = k3
```

  fails to change SEX code **M** to numeral **1**, as specified in the first line, because it is superseded by the second sequence. The following command set will change all codes:

```
if 16 eq 'M' then 16 = k1
if 16 eq 'F' then 16 = k2
if 16 eq 'U' then 16 = k3
```

  The sequence of operations also means that new items created in the current NEWITEM run are initially blank, and thus will not contain useful test values until the program has finished its execution.

- A warning is given when item field length is increased as the result of conversion.

- Floating point numbers (Data Type R) are rounded when converted to integers (Data Type I).

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **newitem subset.out**.

Conversion Table Format

The Conversion Table begins with a group of records that define Data Items whose values (a) are to be changed, or (b) are used to determine changes in other items. The first field in these records consists of a sequential number (three characters), followed by four six-character strings that are identical to mnemonic words of the appropriate Descriptor Record of the input Code File. All fields are separated by a single blank.

These records are followed by a blank line, which in turn is followed by records specifying changes to be made. These records have a format identical to that found in the **Last entry:** line shown in the command entry screen described above. Shown below is the Conversion Table resulting from the changes specified in the examples shown above:

```
  1 NKID
  2 FSIBSZ
  3 SEX
  4 KidSib Ratio

IF SEX EQ 'M' THEN SEX = '1'
IF SEX EQ 'F' THEN SEX = '2'
IF SEX EQ 'U' THEN SEX = BLANKS
IF FSIBSZ NE K0 THEN KidSib Ratio = NKID / FSIBSZ ELSE KidSib Ratio = NKID
```

Provision is made for reuse of the Conversion Table. This is possible, however, only when mnemonic words specified in the table are identical to those in the Code File accompanying file to be modified. If a mnemonic occurs in the table that matches none in the Code File, the following message appears, and the program terminates:

```
ERROR. Table mnemonic does not match Code File mnemonic.
```

Files:

Input:
1. Any Master File or Data File and an associated Code File.
2. A Conversion Table File as defined above.

Recursive use of output files is permitted, that is, file **newitem.out** may be used as input to program NEWITEM.

Output: **newitem.out** contains all records of the input file in the order they have been selected, with specified items changed according to the Conversion Table.

newitem.cde defines the record structure of **newitem.out**. Items are ordered according to the sequence in which they are selected.

newitem.tab contains a new or revised Conversion Table. This file is created each time a new Conversion Table is generated, or when a previously created Table is edited.

Program limits:

100 Conversion table entries, 20 test or change values within square brackets.

PEDTRIM

PEDTRIM simplifies a pedigree by removing individuals that contribute no information to genetic analysis. The program iteratively removes from a pedigree all individuals whose phenotypic or genotypic data are missing or known to be of poor quality, if they have (a) no offspring or (b) no parents and a single offspring (even though this offspring may have data). The file containing the pedigree may be either a Master File or a pedigree file consisting minimally of records containing IDs for Ego and Ego's parents. Data upon which trimming is based may be included on records in the pedigree file, or may be taken from one or more Data Files if they can be matched with records in the pedigree file.

Introductory Display:

```
PEDSYS program PEDTRIM - Copyright 1992, 1999 SFBR

PEDTRIM iteratively removes from a pedigree all individuals whose phenotypic
or genotypic data are missing, if they have (a) no offspring or (b) no
parents and single offspring.

  1.  Unlinked File
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>2
```

1. Selecting Data Files

a. A Linked Data file containing genotypic or phenotypic data

```
Linked Files:

    1.  Example Master File
    2.  Markers + Quant. P'types
    3.  Blood Sample Inventory
    4.  Hemoglobin Levels

Data may be selected from one or more linked files. Enter numbers corres-
ponding to files(s) above. Enter RETURN alone to continue to next step.
>2
```

Here, the Markers + Quantitative Phenotypes file has been selected, from which items are selected for inclusion in output records:

```
 * 1 EGO Permanent ID        3 ADA  Phenotype          5 CA1  Phenotype
   2 ABO  Phenotype          4 APOB Pvu2a Genotype     6 G6PD Phenotype
--------------------------------------------------------------------------------
  PEDTRIM     Select output items             File - QUANTGEN.XMP
--------------------------------------------------------------------------------
Note: Permanent and Sequential IDs from the Master File will be automatically
added to output records. Enter numbers corresponding to additional items to be
included on output records. Enter "A" to select all, [C] to continue.
>
```

Since this is a Linked Data File, pedigree information required by PEDTRIM are taken automatically from the Example Master File to which it is linked.

***Note**: In order to include items from the Master File in PEDTRIM, the Master File itself must be chosen explicitly from the File Selection Display above (not done in this example).*

b. An unlinked file containing genotypic or phenotypic data

It is often convenient to organize multiple genotypes and/or phenotypes on the same record. An example of this organization is given by file **PEDTRTEST.XMP** in which genotypes for five loci have been assembled on records for each individual (display from program BROWSE):

```
PEDTRTEST.XMP                              Items: 6       Top record: 1/32
 (1)   (2)   (3)   (4)   (5)   (6)
 ID    ABO   ADA   APOB  CA1   G6PD
       PHENO PHENO Pvu2a PHENO PHENO

ID015 A                   B
ID019 B     C     A2A2    A     B
ID034 A     AB    A1A2    A     AB
ID038 B     A     A2A2    B     A
ID042 A     AB    A1A2    A     B
ID045             A2A2    A
ID047 O     B             B     B
ID049 O     B     A2A2
ID054 A     AB    A1A1    A     B
ID055 A     A     A2A2    A     B
ID056 B           A2A2    B
ID061 AB    A     A1A2    AB    A
ID062       B     A2A2    AB    B
ID063 B     A     A1A2    AB    B
ID065 AB    AB    A1A1    AB    AB
ID066 A     B     A2A2    B     B
ID067 A                   AB
ID068 B                   AB
ID069 O     B                   B
ID071 AB    A     A1A2    A     A
```

When an unlinked file has been selected, PEDTRIM compares its Code File with entries in **CODE.STD** for standard mnemonics identifying Ego and Ego's parents. As in the example shown above, if the Code File does not contain mnemonics for Permanent IDs **FA** (or **SIRE**) and **MO** (or **DAM**), and/or Sequential IDs **SEQ**, **FSEQ** (or **SSEQ**) and **MSEQ** (or **DSEQ**), a pedigree file containing this information (along with Ego PIDs that match those in the input file) must be selected:

```
Pedigree information is not present in file PEDTRTEST.XMP

Enter - I to select IDs from this file
        P to select a PEDIGREE file
        R to RESELECT data file
        Q to QUIT program
>
```

2. Selecting data for analysis

a. Specifying Critical Data Items

Some data selected for inclusion in output may play little or no role in determining whether an individual is trimmed from the pedigree -- for example, an occasional missing covariate such as age or sex code may not be a serious loss for some analyses, making removal of such individuals unwarranted. Other data, however, may be more critical to an analysis, such that its absence may make Ego a candidate for removal -- this is typically the case for genetic marker or phenotype data that is the primary object of analysis. PEDTRIM requires that a list of critical data items and their unacceptable values be kept as a table. Instructions appearing in the next display control the source of this table:

```
Critical data items and their specifications are kept as a table stored in
file pedtrim.tab.

Enter - S to START a new Pedtrim Table from the keyboard
        U to USE a previously created Pedtrim Table
        Q to QUIT program without executing

>s
```

PEDTRIM creates a default table file named **pedtrim.tab** when a new table is started from the keyboard (choice **S**), as is shown above. Specifying critical data items using the table is a two-step process:

Step 1. Marking Critical Data Items

Critical data items are divided into two classes:

1. Single independent items

   Missing or invalid data for any one of these items make Ego a candidate for removal from the pedigree. These are designated by preceding item numbers with an equal sign when they are entered in the command line of the Item List Display. These are marked with **=** after selection.

2. Grouped alternate items

   Missing or invalid data for some proportion of these items taken as a group make Ego a candidate for removal from the pedigree. These items are designated by preceding item numbers with a plus sign at entry. These are marked with **+** after selection.

Critical data items are selected from the next Item List Display containing the items chosen previously for inclusion in output records:

```
   1 EGO Permanent ID        4 ABO  Phenotype        + 7 CA1  Phenotype
   2 Sire's Permanent ID   = 5 ADA  Phenotype        + 8 G6PD Phenotype
   3 Dam's Permanent ID    + 6 APOB Pvu2a Genotype
--------------------------------------------------------------------------------
  PEDTRIM     Mark critical data
--------------------------------------------------------------------------------
Items above will be included on output records. Precede numbers with "=" for
items that MUST contain valid data. Precede numbers with "+" to select a set
of which a MINIMUM NUMBER must contain valid data.  Enter [C] to continue to
next step.
>
```

In the example above, ABO, ADA, APOB Pvu2a Genotype, CA1 andG6PD Phenotypes, were selected previously for inclusion on output records. The equal sign marking Item 5 indicates that Ego is a candidate for removal from the pedigree if ADA phenotype is invalid or missing from his or her record in the input file **MARKERS.XMP**. The plus signs marking Items 6, 7 and 8 designate APOB Pvu2a Genotype, and CA1 and G6PD Phenotypes as data that will be considered as a group. The disposition of the data group is determined in the display that follows:

```
APOB Pvu2a Genotype
CA1  Phenotype
G6PD Phenotype

Enter the minimum number of the 3 items above that must contain valid data
in order for Ego to be included in the trimmed pedigree.
>2
```

The number 2 entered here specifies that Ego will be trimmed from the pedigree unless two of the three markers listed are present on his or her input data record. Entry of **1** would eliminate Ego only if all three markers were absent simultaneously, etc.

The absence of marks on Item 2 indicates that ABO phenotypic data (including blanks or invalid entries) will be not be used to determine Ego's eligibility for removal. PIDs of Ego, Sire and Dam are included automatically on the output record.

***Note**: Although individuals with missing or invalid data are candidates for removal from the pedigree, they may not in fact be removed if their position in the pedigree is required to link together other family members for which data is present.*

Step 2. Specifying the Validity of Data Entries

The PEDTRIM default is to trim individuals from a pedigree when data are missing from their records. However, there are two situations in which this convention may require modification:

1. The PEDSYS convention of representing missing data as a field of blank characters can potentially lead PEDTRIM to remove the wrong individuals from a pedigree. For example, an attempt to restrict membership of a pedigree to living individuals requires that records containing blank death date entries be *retained* since a blank death date implies that the individual is still living.

2. Some non-blank data items may consist of values that are known to be unacceptable. For example, serum samples used to determine marker phenotypes are often designated **Unt** if they are untypable for some technical reason. Clearly, an individual labeled this way is a candidate for removal, even though the field is not filled with blanks.

The next operation makes it possible to override the simple missing-data default criterion for removal, and to specify alternate data values that are unacceptable, making the individual bearing them a candidate for trimming:

```
   1 ADA  Phenotype        * 3 CA1  Phenotype
   2 APOB Pvu2a Genotype     4 G6PD Phenotype
--------------------------------------------------------------------------------
  PEDTRIM     Redefine exclusions
--------------------------------------------------------------------------------
Enter numbers corresponding to items for which the default missing-data
criterion for exclusion from the pedigree is to be overriden. Enter [C]
to continue.
>
```

In this example, Item 3, the CA1 Phenotype has been selected. Entry of **C** produces the following display, in which unacceptable values for this item may be added:

```
Unacceptable values for - CA1  Phenotype


To ADD new values, enter operator (EQ, NE, GE, GT, LE or LT) and value
separated by a single blank in space below. Enter [C] to continue.
>eq Unt
```

Here, **eq Unt** appearing in the entry space indicates that **Unt** will be designated an unacceptable value for the CA1 phenotype. As each value added, it is numbered and displayed at the top of the screen where it can be modified with the editing function provided:

```
Unacceptable values for - CA1  Phenotype

    1. EQ Unt

To ADD new values, enter operator (EQ, NE, GE, GT, LE or LT) and value
separated by a single blank in space below. To DELETE value, precede
number from list above with "D". Enter [C] to continue.
>eq ?   <
```

As can be seen in this display, value **Unt** has become a numbered value which can be edited according to instructions at the bottom of the screen, while the entry space contains the string **eq ?**, indicating that **?** will be added to the list of invalid data entries.

Note: *Having chosen to override the default exclusion criterion, missing data must be defined explicitly. This is done by specifying blank fields as unacceptable values, as shown in the following screen display:*

```
Unacceptable values for - CA1  Phenotype

    1. EQ Unt
    2. EQ ?

To ADD new values, enter operator (EQ, NE, GE, GT, LE or LT) and value
separated by a single blank in space below. To DELETE value, precede
number from list above with "D". Enter [C] to continue.
>eq     <
```

As a consequence of these entries, CA1 phenotypic values of **Unt**, **?** or blanks are defined as unacceptable when encountered on any record. If these values are found, missing data for either the APOB genotype or G6PD phenotype on same record will make Ego a candidate for removal from the pedigree.

```
Unacceptable values for - CA1  Phenotype

    1. EQ Unt
    2. EQ ?
    3. EQ blanks

To ADD new values, enter operator (EQ, NE, GE, GT, LE or LT) and value
separated by a single blank in space below. To DELETE value, precede
number from list above with "D". Enter [C] to continue.
>       <
```

Unacceptable values are defined separately for each data item selected, after which the next query appears:

```
Remove references to parents trimmed from pedigree?  [Y]/N
>
```

This instruction makes it possible to remove from EGO's record, IDs of parents that have been trimmed from the pedigree. When Standard Mnemonics are used in the input Code File, PEDTRIM automatically replaces parental PIDs with blanks, and sets their Sequential IDs to zero. This is a requirement for many genetic analysis programs.

Notes:

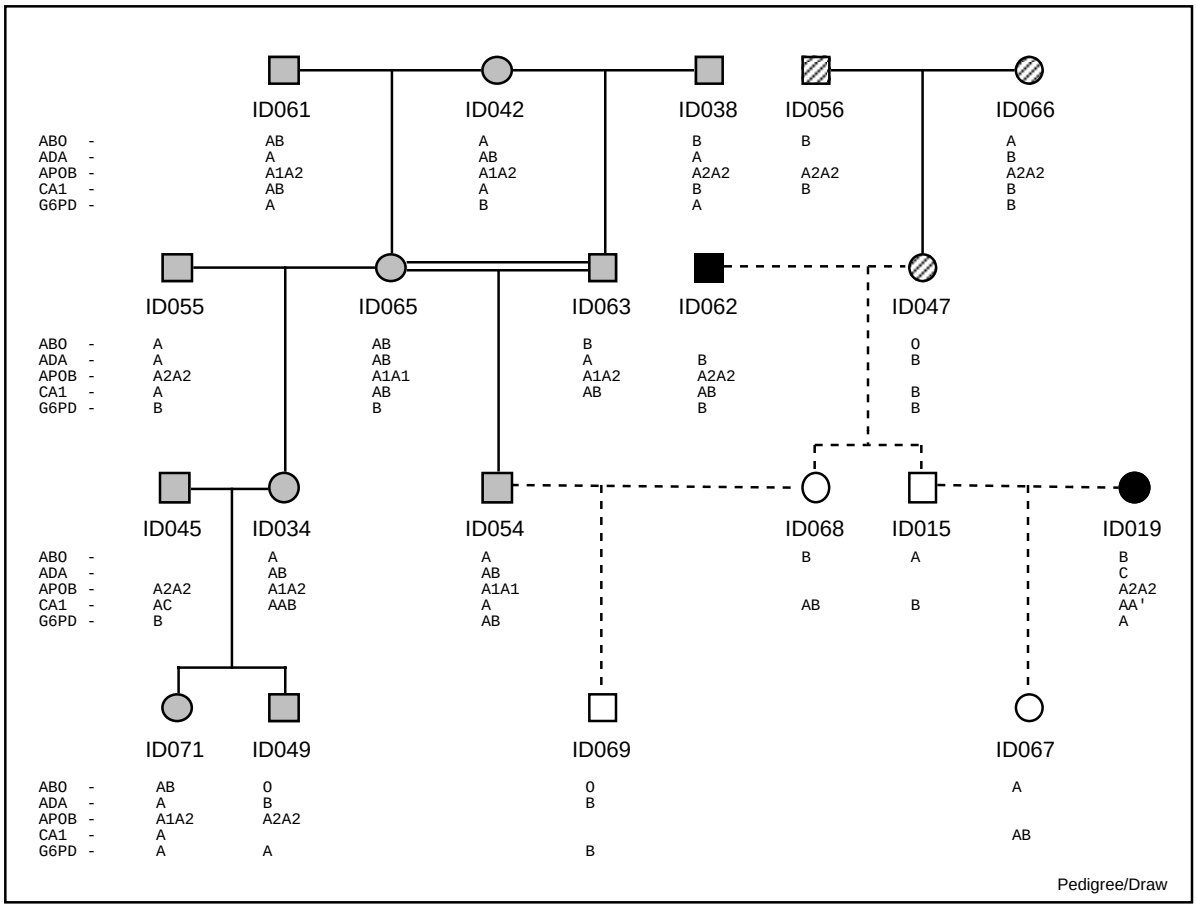
- Operators used to designate unacceptable values for character data (Data Type **C**) are limited to **EQ** and **NE**. Operators for numerical data (Data Types **I**, **R** and **D**) are **EQ**, **NE**, **LT**, **LE**, **GT** and **GE**.

- PEDTRIM minimally requires a pedigree file containing IDs for Ego and Ego's parents as input. This need not be a Master File unless data are required from a linked Data File. If an unlinked file is used, its Code File must contain at least standard mnemonic **EGO** (or **ID**). In addition, an unlinked file must have access to pedigree information. This may be included either (a) in the file itself, in

which case its Code File must also contain standard mnemonics **FA** (**SIRE**) and **MO** (**DAM**) or **SEQ**, **FSEQ** (**SSEQ**) and **MSEQ** (**DSEQ**), or (b) in an independent pedigree file whose Code File includes standard mnemonics for PIDs **EGO** (or **ID**), **FA** (**SIRE**) and **MO** (**DAM**).

Output:

- The pedigree diagram below shows the results of trimming the larger of the two pedigrees in the Example Master File on the basis of the five markers specified above as critical data. Below the IDs listed for each individual is a set of five alphanumeric symbols, each representing a phenotype (or genotype) of one of the five genetic loci listed on the left-hand side of the figure (ABO, ADA, APOB, CA1 and G6PD). Empty pedigree symbols connected by broken lines represent those that have been removed. Filled symbols represent individuals that remain after trimming, but note that what was originally a single pedigree has been broken into two (represented by hatched and stippled fill patterns, respectively), plus two individuals remaining in the sample, but now showing no family links at all (black fill).



The reasons for removal of the six individuals shown are as follows:

ID049 lacks data for both CA1 and G6PD phenotypes, and has no offspring.

ID069 lacks data for both APOB genotype and CA1 phenotype, and has no offspring.

ID067 has data only for CA1 phenotype, and has no offspring.

ID068 and ID015 are missing APOB genotypes and ADA and CA1 phenotypes, and they are left without offspring in the pedigree since ID067 and ID069 have been trimmed.

ID062 is left without offspring in the pedigree since ID015 and ID068 have been trimmed.

Note that as the result of removing the six individuals, the original pedigree has been broken into four separate subunits, each represented in the diagram by distinctive shading. Individuals ID019 and ID062 (represented by a black symbol) are no longer connected to any pedigree, but remain in the output file because they has valid data which may be useful in population measures such as gene and genotype frequencies, etc.

- File **pedtrim.tab** contains the table of critical data items that determine the inclusion of individuals in the trimmed pedigree. The contents of this table are illustrated below (the column of numbers on the left-hand side of the panel have been added as a key to comments following, and do not appear as part of the table itself):

```
 1-        1/ 1 of the following items must be valid
 2-             ADA    PHENO  TYPE          PEDTRTEST.XMP
 3-               1 defined as invalid
 4-                  EQ
 5-        2/ 3 of the following items must be valid
 6-             APOB   Pvu2a  GENO   TYPE   PEDTRTEST.XMP
 7-               1 defined as invalid
 8-                  EQ
 9-             CA1    PHENO  TYPE          PEDTRTEST.XMP
10-               3 defined as invalid
11-                  EQ Unt
12-                  EQ ?
13-                  EQ
14-             G6PD   PHENO  TYPE          PEDTRTEST.XMP
15-               1 defined as invalid
16-                  EQ
```

Record 1 indicates that the single item to follow must be valid if Ego is to remain in a pedigree.

Record 2 starts with the mnemonics of the critical item (here the mnemonics for the ADA phenotype), followed by the name of the file from which data items are taken (here **PEDTRTEST.XMP**). This pattern is repeated in records 6, 9 and 14.

Record 3 indicates the number of values defined explicitly as unacceptable for this item.

Record 4 the **EQ** operator by itself defines the default missing-data criterion of exclusion (a blank character designating missing data in this case).

Record 5 indicates that two of the three following items must be valid if Ego is to remain in the pedigree.

Records 6 - 8 repeat the pattern of Records 2 through 4. This sequence is repeated again in records 14 through 16.

Record 10 specifies that three unacceptable values for this item are designated explicitly, overriding the default missing-data criterion of exclusion.

Records 11 - 13 specify the three values designated as unacceptable.

Although **pedtrim.tab** may be used as input for subsequent runs of PEDTRIM, it is best to rename this file after it is generated to avoid overwriting it.

Note:

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **pedtrim subset.out**.

Files:

Input: Any Master File or an unlinked pedigree file with an associated Code File.
The Standard Mnemonics File (CODE.STD).

Recursive use of output files is permitted, that is, file **pedtrim.out** may be used as input to program PEDTRIM.

Output: **pedtrim.out** contains records of individuals remaining in the trimmed pedigree.

pedtrim.cde defines the record structure of **pedtrim.out**.

pedtrim.tab contains the table of critical data items that determine the inclusion of individuals in the trimmed pedigree.

PEELSEQ

Program PEELSEQ determines a peeling sequence (Cannings, et al. 1978) from an indexed pedigree file.

Introductory Display:

```
PEDSYS program PEELSEQ - Copyright 1995, 1999 SFBR

PEELSEQ determines a peeling sequence from an indexed pedigree file.

 1.  Unlinked File
 2.  Example Master File

Enter number, file name, or "Q" to quit.
>
```

Input file requirements are Sequential IDs of EGO, FA and MO, and Pedigree ID number. The program runs automatically if mnemonic PEDNO, or Standard Mnemonics for IDs (SEQ, FSEQ, MSEQ, EGO, FA and MO for humans, SEQ, SSEQ, DSEQ, ID, SIRE and DAM for nonhumans), are found in the input Code File. Otherwise, an opportunity is provided to designate required items labeled with nonstandard mnemonics. When a pedigree file has been chosen, a request for an alternate output file name appears if the default file **peelseq.tab** is not wanted:

```
Enter output file name or RETURN for default name peelseq.tab.
>
```

The program then begins execution. Format and content of output records from file **peelseq.tab** are illustrated in the following table:

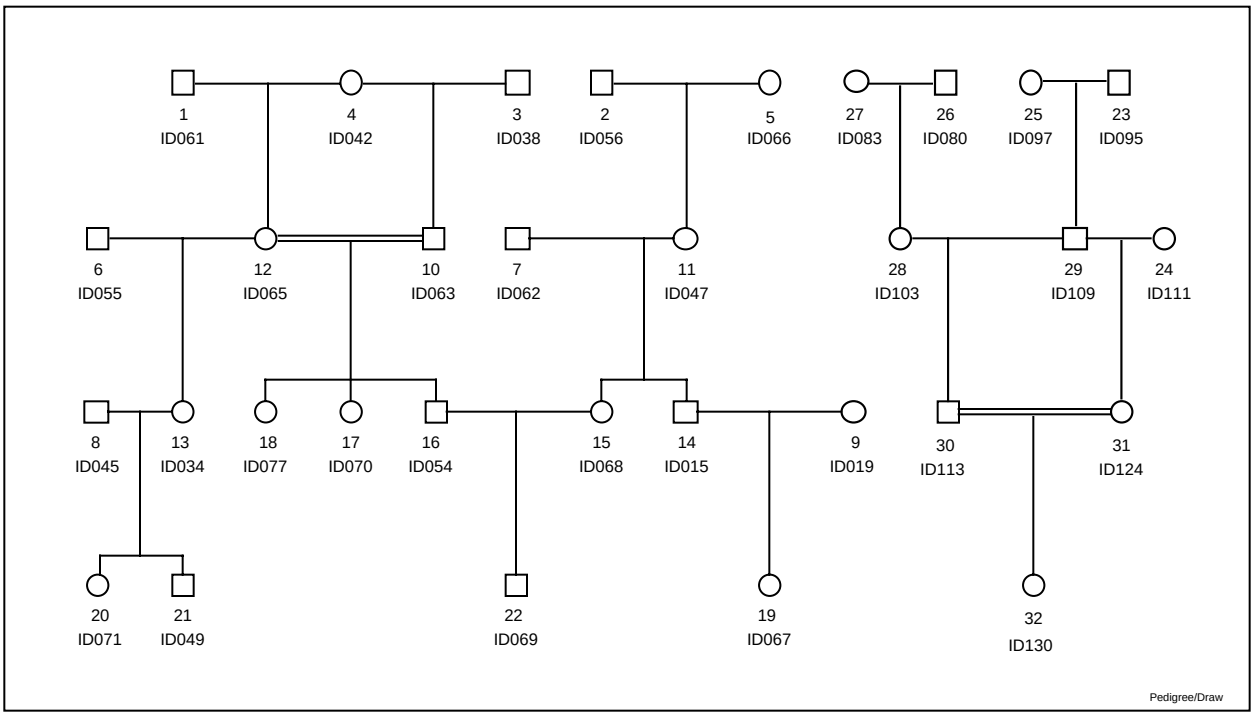
ID FA	ID MO	Founder Status FA	Founder Status MO	No. of Kids	ID Kid 1	ID Kid 2	ID Kid 3	Cut-set Size	Link ID	Link ID
10	12	0	0	3	16	17	18	2	12	10

The first five variables are invariant, but **No. of Kids** variable *i* is followed by *i* offspring **ID**s, and **Cut-set Size** *j* is followed by *j* **Link ID** s. This structure means that the number of variables in a record will vary depending upon the number of offspring and the cut-set size of the nuclear family it represents. Linking individuals are those that connect a nuclear family to other parts of the pedigree that have not yet been included in the peeling sequence. At least one of these linking individuals is a member of the nuclear family in question who is also a member of another nuclear family not yet included in the sequence. If the cut-set size for the nuclear family is greater than 1, additional linking individuals may be relatives of a family member, who are also members of families that have not yet been processed. Note that each record in the output file corresponds to a nuclear family in the input file. **Founder Status** 0 indicates that both parents of an individual are known; status 1 indicates that neither parent is known.

The table below shows the contents of output file **peelseq.tab** giving the peel sequence generated from the Example Master File formatted for input to PAP:

```
 2    5    1    1    1   11    1   11
 8   13    1    0    2   20   21    1   13
14    9    0    1    1   19    1   14
 6   12    1    0    1   13    1   12
 7   11    1    0    2   14   15    1   15
16   15    0    0    1   22    1   16
10   12    0    0    3   16   17   18    2   12   10
 3    4    1    1    1   10    2    4   12
 1    4    1    1    1   12    0
23   25    1    1    1   29    1   29
26   27    1    1    1   28    1   28
29   28    0    0    1   30    2   29   30
31   30    0    0    1   32    2   29   31
29   24    0    1    1   31    0
```

A pedigree diagram of the Example Master File from which the peel sequence was computed is shown below:



The table below shows the contents of file **peelseq2.tab** in which (for ease of interpretation) PIDs are substituted for the Sequential IDs required by PAP. Except for this substitution, entries in this table are identical to those in output file **peelseq.tab**:

```
ID056   ID066    1    1    1   ID047    1   ID047
ID045   ID034    1    0    2   ID071   ID049    1   ID034
ID015   ID019    0    1    1   ID067    1   ID015
ID055   ID065    1    0    1   ID034    1   ID065
ID062   ID047    1    0    2   ID015   ID068    1   ID068
ID054   ID068    0    0    1   ID069    1   ID054
ID063   ID065    0    0    3   ID054   ID070   ID077    2   ID065   ID063
ID038   ID042    1    1    1   ID063    2   ID042   ID065
ID061   ID042    1    1    1   ID065    0
ID095   ID097    1    1    1   ID109    1   ID109
ID080   ID083    1    1    1   ID103    1   ID103
ID109   ID103    0    0    1   ID113    2   ID109   ID113
ID124   ID113    0    0    1   ID130    2   ID109   ID124
ID109   ID111    0    1    1   ID124    0
```

If the program executes successfully, it terminates with the following message

```
*****************************************************************************
*                                                                           *
*     PEELSEQ is finished.                                                  *
*     Output is in directory /export/home/population/bdyke/example32/       *
*        peelseq.tab                                                        *
*        peelseq2.tab                                                       *
*     The maximum cutset size is 2                                          *
*                                                                           *
*     Be sure to rename output files before rerunning this program.         *
*                                                                           *
*****************************************************************************
```

Notes:

- IDs appearing in output file **peelseq.tab** refer only to the indexed pedigree file used as input to PEELSEQ. Thus, these IDs may not correspond to Sequential IDs in the population Master File.

- Output file **peelseq2.tab** is not produced if the input Code File does not contain standard mnemonics for PIDs.

Files:

Input: Any Master File or indexed pedigree file with an accompanying Code File.

Output: **peelseq.tab** contains the peeling sequence (with integer IDs) formatted for input to PAP.

peelseq2.tab contains a more easily interpretable version of the peeling sequence (with Permanent IDs). This file is not used as input to PAP.

Algorithm: J. W. MacCluer, D. T. Bishop

Reference:

Cannings C, Thompson EA and Skolnick MH 1978 Probability functions on complex pedigrees. *Advances in Applied Probability* 10:26-61.

PREPDRAW

Program PREPDRAW changes PEDSYS records into *Pedigree/Draw* 5.0 format. The program automatically selects required items (PIDs and SEX) and allows choice of text fields from PEDSYS records. Codes designating *Pedigree/Draw* symbol types and attributes may selected from four symbol palettes. Application of symbol codes is conditional upon values found in specified items on the PEDSYS input record. The file containing data to be converted to *Pedigree/Draw* format may be a Master File alone, a Master File with one or more Linked Data Files, or any pedigree file consisting of records each of which contains minimally a unique identifier for EGO, Father and Mother.

Introductory Display:

```
PEDSYS program PREPDRAW - Copyright 1996, 1999 SFBR

PREPDRAW converts PEDSYS pedigree records to Pedigree/Draw 5.0 format.

  1.  Unlinked File
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>2
```

When a PEDSYS Data Directory such as the EXAMPLE Directory shown above, the following display appears:

```
  * 1.  Example Master File
    2.  Markers + Quant. P'types
    3.  Blood Sample Inventory
    4.  Hemoglobin Levels

Data may be selected from one or more linked files. Enter numbers corres-
ponding to files(s) above, or RETURN to continue.
>2
```

Note that the Example Master File has been pre-selected, and that information from one or more Linked Data Files may be chosen for inclusion in output. The option of assembling data from multiple files is not given if an unlinked pedigree file has been selected initially.

Once input data files have been selected, construction of a *Pedigree/Draw* file involves a series of steps in which (1) a Symbol Selection Table is specified, (2) items to appear as text fields in the the *Pedigree/Draw* diagram are specified, (3) symbol types and attributes are selected from standard *Pedigree/Draw* palettes, (4) the Symbol Selection Table may be modified (optionally), and (5) individuals to be included in the output are selected.

Step 1. Specifying a Symbol Selection Table

The Symbol Selection Table collects the sequence of text and symbol selections as they are entered during the operation of PREPDRAW. Once created, this table (which is distinct from the Symbol Palettes described below) may be edited during any run of the program as described in **Step 4** below. The table also may be saved and reused so that new *Pedigree/Draw* files can be created from the same

PEDSYS files without having to repeat an entire symbol selection sequence. When files have been chosen, the following display appears, from which the source of a Symbol Selection Table may be chosen:

```
A previously created table of pedigree text and symbol selections may be
 used, or a new table may be entered from the keyboard.

Enter - S  to START a new Symbol Selection Table
        T  to USE an existing Symbol Selection Table
        Q  to QUIT program.
>s
```

In the example above, creation of a new table has been specified.

Step 2. Specifying Text Fields

Next, items that will appear as text fields below the symbols representing individuals in the *Pedigree/Draw* diagram are selected:

```
   1 EGO SEQUENTIAL ID      10 MATERNAL SIBSHIP SIZ   19 Mate's Permanent ID
   2 SIRE'S SEQ             11 FULL SIBSHIP SIZE      20 PEDIGREE NUMBER
   3 DAM'S SEQ             *12 EGO Permanent ID       21 GENERATION NUMBER
   4 FIRST OFFSPRING SEQ   *13 Birth Date (YYYYMMDD   22 Ego ID
   5 NEXT PATERNAL SIB SE  *14 Sire's Permanent ID   *23 ADA  Phenotype
   6 NEXT MATERNAL SIB SE  *15 Dam's Permanent ID    *24 APOB Pvu2a Genotype
   7 NEXT FULL SIB SEQ     *16 Sex (M, F or U)        25 Tot. Ser. Cholestero
   8 NUMBER OF OFFSPRING   *17 Exit Date (YYYYMMDD)   26 APOAI Level
   9 PATERNAL SIBSHIP SIZ   18 Exit Code
--------------------------------------------------------------------------------
  PREPDRAW    Select text fields
--------------------------------------------------------------------------------
Enter numbers corresponding to items to be included as optional text fields
(5 items maximum). Required items are pre-selected. All text items longer
than 24 characters are truncated. Enter [C] to continue.
>
```

PIDs of EGO, FA and MO, as well as SEX, birth date and death date are pre-selected here, because they are automatically maintained as special-purpose symbols and text fields in *Pedigree/Draw* records. Item 23 (ADA Phenotype) has been entered in the command line as the first of the five *optional* text fields in the example above.

Note that items from the Master File, plus those from the Data File chosen appear together in this display. Here, Items 1 - 21 are from the Example Master File, and are followed immediately by Items 22 - 26 from the Markers + Quant. P'types File.

Step 3. Selecting Symbol Types and Attributes

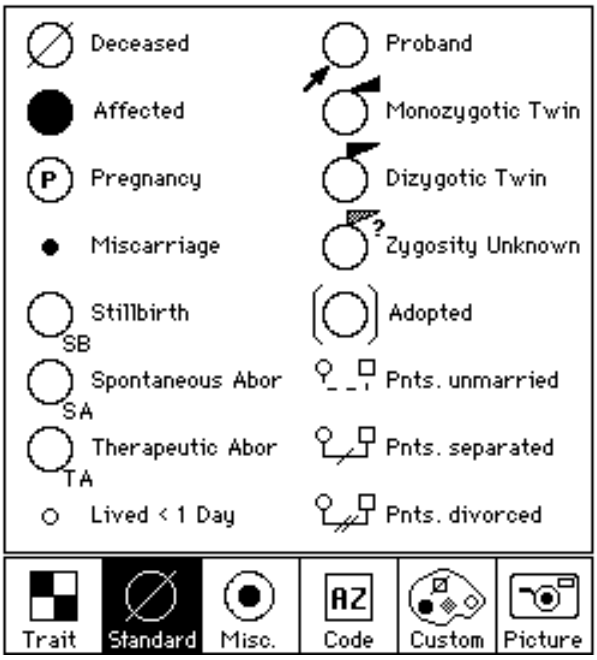
After text fields have been specified, the following screen appears, from which palettes containing codes specifying graphic symbols may be chosen. In addition to PIDs of EGO, father and mother, *Pedigree/Draw* needs only SEX coding to determine basic symbol representation (a square or circle designating males or females, respectively), so that this next step may be skipped if no additional attributes are to be assigned to individual symbols in the pedigree.

```
Pedigree/Draw Palettes:

  1 - Standard pre-defined symbol types
  2 - Filled quadrants representing combinations of up to 4 traits
  3 - Two-character alphabetic codes
  4 - Miscellaneous pre-defined attributes shading, etc.

Enter number corresponding to palette from which symbol types and attributes
will be selected for inclusion on output records.  Enter [C] to continue to
next step.
>1
```

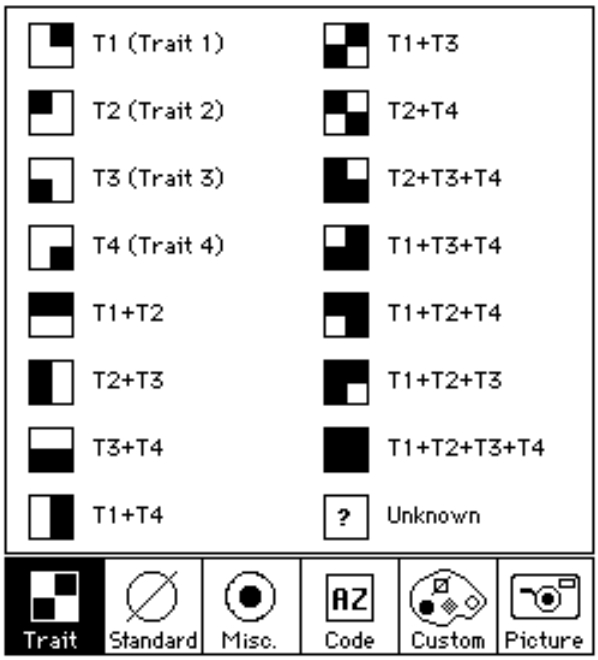
Each palette contains 16 symbols or patterns that can be combined in various ways to signify characteristics of EGO as represented graphically in a pedigree diagram. Entry of **1** here results in display of data items, plus a list of standard symbol types found in *Pedigree/Draw* Palette 1. Entries from this list are what appear in the legend box of *Pedigree/Draw* output.

```
   1 EGO SEQUENTIAL ID      10 MATERNAL SIBSHIP SIZ   19 Mate's Permanent ID
   2 SIRE'S SEQ             11 FULL SIBSHIP SIZE      20 PEDIGREE NUMBER
   3 DAM'S SEQ              12 EGO Permanent ID       21 GENERATION NUMBER
   4 FIRST OFFSPRING SEQ    13 Birth Date (YYYYMMDD   22 Ego ID
   5 NEXT PATERNAL SIB SE   14 Sire's Permanent ID    23 ADA  Phenotype
   6 NEXT MATERNAL SIB SE   15 Dam's Permanent ID     24 APOB Pvu2a Genotype
   7 NEXT FULL SIB SEQ      16 Sex (M, F or U)        25 Tot. Ser. Cholestero
   8 NUMBER OF OFFSPRING    17 Exit Date (YYYYMMDD)   26 APOAI Level
   9 PATERNAL SIBSHIP SIZ   18 Exit Code
--------------------------------------------------------------------------------
  PREPDRAW    Palette 1  Standard types
--------------------------------------------------------------------------------
After IF> enter <ItemNo> <Operator> <Test> THEN SYMBOL = <symbol>.  EXAMPLES
   IF> 1 EQ ID045 THEN S = I,   IF> 2 EQ ID055 AND 3 EQ ID065 THEN SYMBOL = P
Enter "H" for explanation and shortcuts, [C] to continue.

A Deceased        E Stillbirth       I Proband          M Adopted
B Affected        F Abort. (spont.)  J Monozygot. twin  N Parents unmarr.
C Pregnancy       G Abort. (thera.)  K Dizygotic twin   O Parents separat.
D Miscarriage     H Lived < 1 day    L Zygosity unk.    P Parents divorc.

Last entry: IF 18 EQ "1" THEN SYMBOL = A
IF>12 eq 'ID015' then i
```

Here, the standard "deceased" pattern (a diagonal line drawn through the basic symbol) has been selected, and will be assigned to each individual in the pedigree that has an Exit Code (Item 18) equal to **1**, and the command line contains an entry that will select individual ID015 as the proband (a diagonal arrow pointing to the symbol designating EGO). Rules of pattern assignment are the same as those governing conditional item modification in program NEWITEM. Graphic representation of Palette 1 is shown in the left-hand panel below.



Palette 1 - Standard Types



Palette 2 - Traits (quadrants)

- **Note:** *No more than a total of four symbol types may be assigned to the same individual from Palette 1 and Palette 4 combined.* Additional attributes may be assigned to these symbols from other palettes, for a total of six altogether.

Choice of Palette **2** (shown in the right-hand panel of the figure above) results in the following display:

```
   1 EGO SEQUENTIAL ID      10 MATERNAL SIBSHIP SIZ   19 Mate's Permanent ID
   2 SIRE'S SEQ             11 FULL SIBSHIP SIZE      20 PEDIGREE NUMBER
   3 DAM'S SEQ              12 EGO Permanent ID       21 GENERATION NUMBER
   4 FIRST OFFSPRING SEQ    13 Birth Date (YYYYMMDD   22 Ego ID
   5 NEXT PATERNAL SIB SE   14 Sire's Permanent ID    23 ADA  Phenotype
   6 NEXT MATERNAL SIB SE   15 Dam's Permanent ID     24 APOB Pvu2a Genotype
   7 NEXT FULL SIB SEQ      16 Sex (M, F or U)        25 Tot. Ser. Cholestero
   8 NUMBER OF OFFSPRING    17 Exit Date (YYYYMMDD)   26 APOAI Level
   9 PATERNAL SIBSHIP SIZ   18 Exit Code
--------------------------------------------------------------------------------
  PREPDRAW    Palette 2  Quadrants
--------------------------------------------------------------------------------
After IF> enter <ItemNo> <Operator> <Test> THEN SYMBOL = <symbol>.  EXAMPLES
   IF> 11 EQ A THEN SYMBOL = A,   IF> 12 EQ '-' THEN S = P
Enter "H" for explanation and shortcuts, [C] to continue.

A Trait 1         E Trait 1+2        I Trait 1+3        M Trait 1+2+4
B Trait 2         F Trait 2+3        J Trait 2+4        N Trait 1+2+3
C Trait 3         G Trait 3+4        K Trait 2+3+4      O Trait 1+2+3+4
D Trait 4         H Trait 1+4        L Trait 1+3+4      P ? (unknown)

Last entry: IF 25 LE K99 THEN SYMBOL = F
IF> 25>k99 then h
```

Here, the line displaying the last entry specifies that the two right-hand quadrants of symbols representing individuals with a total serum cholesterol greater than 99 will be filled. Other quadrants and combinations of quadrants may be used to represent other phenotypes, as well.

- **Note:** *Only one pattern from Palette 2 may be assigned to a single individual.*

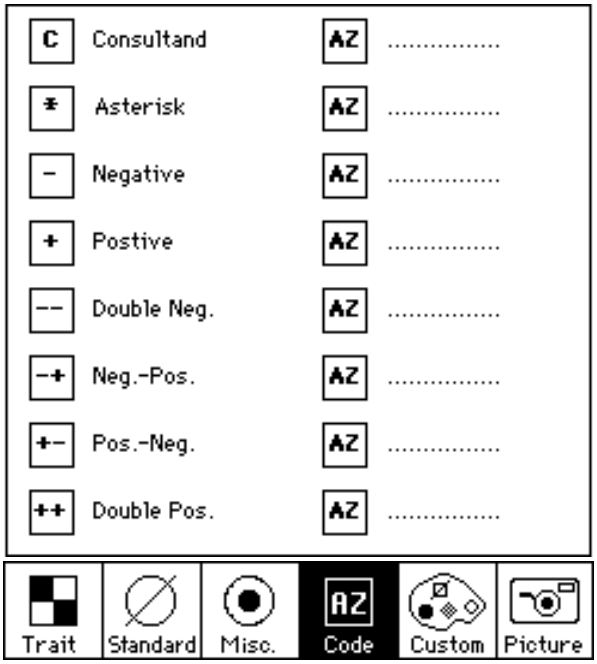
Selection of Palette **3** results in the following display (page 2 shown here also):

```
   1 EGO SEQUENTIAL ID      10 MATERNAL SIBSHIP SIZ   19 Mate's Permanent ID
   2 SIRE'S SEQ             11 FULL SIBSHIP SIZE      20 PEDIGREE NUMBER
   3 DAM'S SEQ              12 EGO Permanent ID       21 GENERATION NUMBER
   4 FIRST OFFSPRING SEQ    13 Birth Date (YYYYMMDD   22 Ego ID
   5 NEXT PATERNAL SIB SE   14 Sire's Permanent ID    23 ADA  Phenotype
   6 NEXT MATERNAL SIB SE   15 Dam's Permanent ID     24 APOB Pvu2a Genotype
   7 NEXT FULL SIB SEQ      16 Sex (M, F or U)        25 Tot. Ser. Cholestero
   8 NUMBER OF OFFSPRING    17 Exit Date (YYYYMMDD)   26 APOAI Level
   9 PATERNAL SIBSHIP SIZ   18 Exit Code
--------------------------------------------------------------------------------
  PREPDRAW    Palette 3  Alpha. codes
--------------------------------------------------------------------------------
After IF> enter <ItemNo> <Operator> <Test> THEN SYMBOL = <symbol>.  EXAMPLES
   IF> 14 EQ 'UNT' THEN S = B,   IF> 9 EQ K800101 THEN SYMBOL = C
Enter "H" for explanation and shortcuts, [C] to continue.

A C Consultand    E -- Double neg.   I Unassigned 1     M Unassigned 5
B * Asterisk      F -+ Neg.-Pos.     J Unassigned 2     N Unassigned 6
C - Negative      G +- Pos.-Neg.     K Unassigned 3     O Unassigned 7
D + Positive      H ++ Double pos.   L Unassigned 4     P Unassigned 8


IF> 26=b then b
```

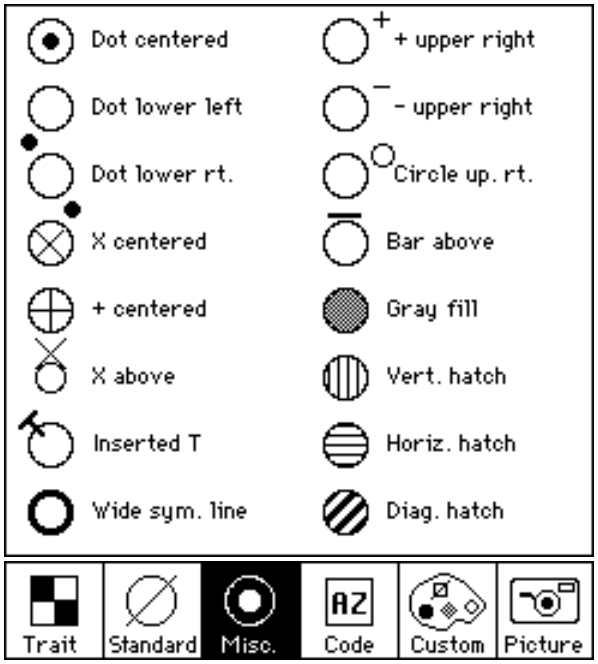
In this example, an asterisk (Code B) will placed in the center of symbols representing all individuals with an unlisted (blank) APOAI Level (Item 26).

Entries in Palette 3 differ from those of other palettes in that the first two characters are interpreted as alphabetic codes that serve as symbol attributes displayed in *Pedigree/Draw* graphic output. The Palette 3 default is shown graphically in the left-hand panel of the figure below. Code **AZ** signifies an unassigned attribute, which can be edited in *Pedigree/Draw*.

- **Note:** *Only one alphabetic code from Palette 3 may be assigned to a single individual.*



Palette 3 - Alphabetic Codes



Palette 4 - Misc. Attributes

Selection of Palette **4** (shown graphically in the right-hand panel in the figure above) results in the following display:

```
   1 EGO SEQUENTIAL ID      10 MATERNAL SIBSHIP SIZ   19 Mate's Permanent ID
   2 SIRE'S SEQ             11 FULL SIBSHIP SIZE      20 PEDIGREE NUMBER
   3 DAM'S SEQ              12 EGO Permanent ID       21 GENERATION NUMBER
   4 FIRST OFFSPRING SEQ    13 Birth Date (YYYYMMDD   22 Ego ID
   5 NEXT PATERNAL SIB SE   14 Sire's Permanent ID    23 ADA  Phenotype
   6 NEXT MATERNAL SIB SE   15 Dam's Permanent ID     24 APOB Pvu2a Genotype
   7 NEXT FULL SIB SEQ      16 Sex (M, F or U)        25 Tot. Ser. Cholestero
   8 NUMBER OF OFFSPRING    17 Exit Date (YYYYMMDD)   26 APOAI Level
   9 PATERNAL SIBSHIP SIZ   18 Exit Code
--------------------------------------------------------------------------------
  PREPDRAW    Palette 4  Miscellaneous
--------------------------------------------------------------------------------
After IF> enter <ItemNo> <Operator> <Test> THEN SYMBOL = <symbol>.  EXAMPLES
   IF> 12 EQ AB THEN S = D,   IF> 10 NE b AND 10 LT k942131 THEN SYMBOL = E
Enter "H" for explanation and shortcuts, [C] to continue.

A Dot centered    E + centered       I + upper right    M Gray fill
B Dot lower left  F X above          J - upper right    N Vert. hatch
C Dot lower rght  G Inserted T       K Circle up. rt.   O Horiz. hatch
D X Centered      H Heavy symbol     L Bar above        P Diag. hatch

Last entry: IF 23 EQ BLANKS AND 24 EQ BLANKS THEN SYMBOL = M
IF>
```

Here, the symbol representing any individual with a missing ADA Phenotype (Item 23) will be filled with gray shading..

- **Note:** *A total of no more than four symbol types may be assigned to the same individual from Palette 1 and Palette 4 combined.* Additional attributes may be assigned to a single individual from other palettes, for a total of six altogether.

Step 4. Modifying the Symbol Selection Table

When selection of symbol types and attributes is done, the palette selection display appears with an additional choice providing an opportunity to modify the Symbol Selection Table:

```
  V  - to VIEW and EDIT text and symbol selections
 [I] - to IDENTIFY pedigree records for inclusion in output file
  S  - to SAVE file "prepdraw.tab" and EXIT program
  Q  - to QUIT program without executing or saving table

Enter letter above corresponding to action to be taken.
>
```

Command **V** displays the Symbol Selection Table Edit Display:

```
Palette 1:
IF EXCODE EQ '1' THEN SYMBOL = Deceased
IF EGO EQ 'ID015' THEN SYMBOL = Proband
Palette 2:
IF TSC GT K0 AND TSC LE K99 THEN SYMBOL = Trait 2+3
IF TSC GT K99 THEN SYMBOL = Trait 1+4
Palette 3:
IF APOAI Level EQ K0 THEN SYMBOL = * Asterisk
Palette 4:
IF ADA PHENO TYPE EQ BLANKS THEN SYMBOL = Gray fill
--------------------------------------------------------------------------------
      PREPDRAW    Add/delete entries       File - prepdraw.tab
--------------------------------------------------------------------------------

Enter - D to DELETE an entry
        A to ADD a symbol from palettes
       [C] to CONTINUE
>
```

Option **D** produces a request for one or more numbers corresponding to text and / or symbol selection commands in the display above. When numbers are entered, corresponding lines are deleted.

A returns the palette selection display, from which new symbols or attributes may be selected.

P produces of subsequent pages of the table.

C is entered when the Symbol Selection Table requires no further modification.

Step 5. Selecting Individuals to be Included in Pedigree

If one or more linked files has been used to define symbols and/or text fields, the edit display menu includes an additional command (I):

```
  V  - to VIEW and EDIT text and symbol selections
 [I] - to IDENTIFY pedigree records for inclusion in output file
  S  - to SAVE file "prepdraw.tab" and EXIT program
  Q  - to QUIT program without executing or saving table

Enter letter above corresponding to action to be taken.
>i
```

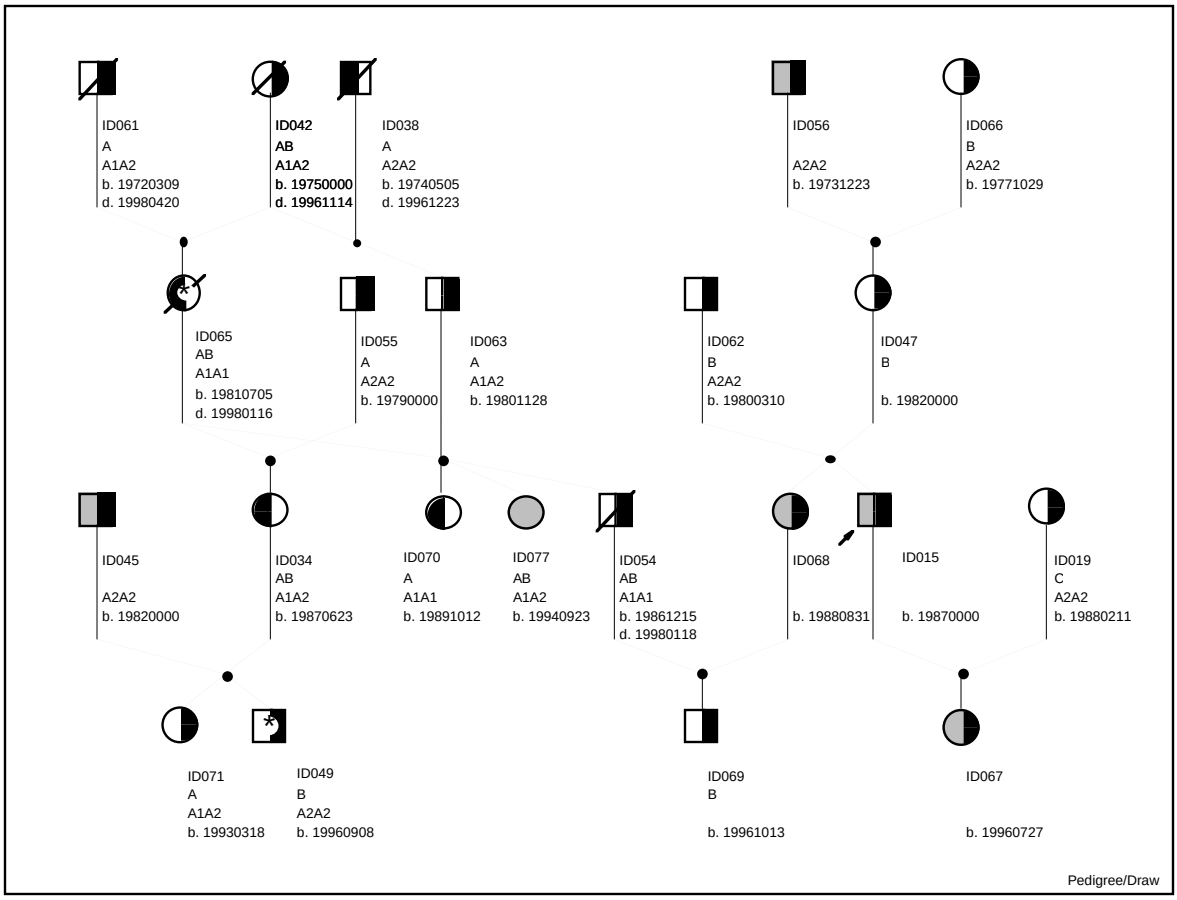
Command **I** results in a request for identification of records of individuals to be included in the output file **prepdraw.tab**:

```
Output file prepdraw.ped may contain all, or a subset of individuals found
in the input pedigree file.  (Note: Pedigree links may be lost if a subset
is chosen.)

  1.  Unlinked File
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>2
```

When an appropriate pedigree file has been chosen, the following display appears:

```
   1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ   16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE   12 EGO Permanent ID       19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE   13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  PREPDRAW    Identify Ego position           File - MASTER.XMP
--------------------------------------------------------------------------------
Enter a number above corresponding to Ego ID; [C] to continue to
next step.
>12
```

Here, the Example Master File was selected as the location of the pedigree records, and the PID has been selected to identify EGO on these records. Alternatively, EGO Sequential ID could have been chosen. When this selection has been made, the program begins execution. An example of the pedigree diagram produced from the output based on selections made above is shown below:



If an unlinked file has been used to define symbols and/or text fields, Step 5 is skipped, and the following display appears:

```
 V  - to VIEW and EDIT text and symbol selections
[C] - to CONTINUE program execution
 S  - to SAVE file "prepdraw.tab" and EXIT program
 Q  - to QUIT program without executing or saving table

Enter letter above corresponding to action to be taken.
>i
```

Command **C** starts program execution.

S saves the Symbol Selection Table in file **prepdrw2.tab**, and exits the program without further execution.

Q exits the program without saving the Symbol Selection Table.

Notes:

- **Using PREPDRAW with an existing Symbol Selection Table**

If option **T** has been chosen from the Introductory Display, a request for the name of the file containing a Symbol Selection Table appears. When this file name is entered, PREPDRAW compares

the mnemonic listings in the file with mnemonics found in the input Code File(s). A Symbol Selection Table Edit Display then appears. If the input file(s) currently being read are not the same as those used during the original construction of the Symbol Selection Table, entries in this display may differ from those of the original table:

a. The current display contains only those text and symbol selection commands whose mnemonics match Code File entries, which means that some of the original entries may be missing.

b. PREPDRAW accepts the first occurrence of a matching mnemonic sequence, which may result in selection of an incorrect item if input consists of multiple files containing duplicate item names.

- **Shortcuts**

A shortcut may be used to speed entry of PREPDRAW command sequences by reducing the number of separate command sequences required for testing a single item against multiple test values. A single sequence may be written for this case, as follows:

```
if 12 EQ [ID071,ID049,ID045,ID034] then symbol = a
```

that is, multiple test values separated by commas may follow a single logical operator. When the initial operator is **EQ**, commas separating test values are interpreted as **OR** operators, so that the statement above is true when Item 12 (EGO PID) is the same as any one of the PIDs in the list. For any other initial operator, commas separating test values are interpreted as **AND** operators. Thus, the following statement

```
IF 8 LT [9,10,11]...
```

is true when Item 8 (NUMBER OF OFFSPRING) is less than all of the values of Items 9, 10 and 11 on that record (PATERNAL, MATERNAL and FULL SIBSHIP SIZE, respectively).

This shortcut applies only to test values, that is, multiple item numbers may not be tested in this way. Data types of test items are determined by the data types listed in the input Code File.

The constant designator can also be used with multiple test values. For example:

```
IF 11 eq k[1,2,3]...      IF 11 eq [9,k2]...
```

are legitimate expressions.

- **Substring searches**

Substrings are defined by preceding or appending an asterisk (*) to the characters sought. For example:

```
if 12 eq *5 then g
if 12 eq ID02* then h
if 12 eq *0* then i
```

The first command assigns a symbol type to any PID ending in the numeric character **5**; the second command makes an assignment to any PID ending with numeric characters **20** to **29**; while the third command assigns a symbol to any PID containing the numeric character **0**.

Substring values may be tested in multiples, as well:

```
if 12 EQ [*071,*049,*045,*034] then s = a
```

Here, the numerical portion of EGO PID is tested for four specific values. These examples illustrate the point that substring searches may be done on any field, no matter how the item Data Type is defined in the Code File (C, D, I or R).
Substring searches are slow, and should be avoided when rapid output is needed.

- **Equivalences:**

As a convenience, several command elements have been given equivalent alternate forms. These are:

EQ = equal sign (=)	**GT** = **>**	**GE** = **>=** or **=>**
NE = **<>** or **><**	**LT** = **<**	**LE** = **<=** or **=<**

'' = **" "** (must be matched pairs)

In addition, the assignment statement has two short forms so that the following commands are equivalent:

```
if 12 EQ ID015 then symbol = a
if 12 EQ ID015 then s = a
if 12 EQ ID015 then a
```

- **Symbol Selection Table Format**

The Symbol Selection Table consists of records containing command sequences formatted as written at time of input. Shown below is the table resulting from the changes specified in the examples shown above:

```
 1 ADA    PHENO  TYPE
 2 APOAI  Level
 3 EGO
 4 EXCODE
 5 TSC

Palette 1 - Standard pre-defined symbol types:
IF EXCODE EQ '1' THEN SYMBOL = Deceased
IF EGO EQ 'ID015' THEN SYMBOL = Proband
Palette 2 - Filled quadrants representing traits:
IF TSC GT K0 AND TSC LE K99 THEN SYMBOL = Trait 2+3
IF TSC GT K99 THEN SYMBOL = Trait 1+4
Palette 3 - Two-character alphabetic codes:
IF APOAI Level EQ K0 THEN SYMBOL = * Asterisk
Palette 4 - Miscellaneous pre-defined attributes:
IF ADA PHENO TYPE EQ BLANKS THEN SYMBOL = Gray fill

 Text lines:
 1 ADA    PHENO  TYPE
 2 APOB   Pvu2a  GENO   TYPE
```

Files:

Input: 1. A Master File, with or without one or more linked Data Files, and associated Code File(s).

2. A pedigree file (having records containing minimally a unique identifier for EGO and EGO's parents), and associated Code File.

Output: **prepdraw.ped** contains a *Pedigree/Draw* 5.0 header record, followed by all records of the input file in the order they have been selected, with specified items changed according to the Symbol Selection Table and palettes.

prepdraw.tab contains a new or revised Symbol Selection Table. This file is created each time a new table is generated, or when a previously created Table is edited.

Program limits:

100 conditional assignments ("IF" statements).

REFORMAT

Program REFORMAT is used to reformat, join, delete, rearrange the order, etc., of items in records of a single Master File or Data File. The program also makes it possible to add items consisting of blanks, a character string, and a series of sequential or randomly ordered numbers.

Introductory Display:

```
PEDSYS program REFORMAT - Copyright 1992, 1999 SFBR

REFORMAT is used to reformat, join, delete, rearrange the order, etc., of
items in records of a single Master File or Data File.

  1.  Unlinked File
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>
```

Next, a Standard Item List is shown:

```
   1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ   16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE   12 EGO Permanent ID       19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE   13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  REFORMAT    Items to be included            File - MASTER.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to items in the order they are to be included in
reformatted record, "A" to include all items. Precede number with > or < to
justify, "U" or "L" to set case, "F" to reformat item. Enter "B", "K", "R"
or "S" to insert blank, constant, random fraction, or integer sequence; "J"
to join items, "H" to help, [C] to continue.
>
```

Commands:

Item number Reordering of items within records is accomplished by entering <n>, the number corresponding to the item in the list above the instruction line, in the order that items are to appear in the output record.

><n> or **<<n>** where <n> is an item number as described above. Entry of this sequence **right** or **left** justifies the character string in the field. If the justification feature is used, the usual asterisk marking items chosen for inclusion will be replaced by directional markers: **>** marks items to be right justified, **<** marks items to be left justified.

U/L Preceding a number with a **u** or **l** changes all alphabetic characters in the corresponding field to upper or lower case, respectively.

F<n> where <n> is an item number as described above. This sequence produces the following display, which controls reformatting of the item selected:

```
 PEDNO currently width 5, Data Type I

   1. Increase field width of item
   2. Decrease field width of item
   3. Subdivide item

Enter number corresponding to format change required, or RETURN alone to
continue without changes
>
```

In this example, Pedigree Number has been selected for reformatting.

- **Increase field width of an item**

  Entry of **1** produces the following display:

```
Enter "<" or ">" plus number of LEADING or TRAILING blanks to add to field.
>
```

Field width is increased by adding blanks to the right or left margin of the item selected. Example: entry of **<2** would add two blank characters to the beginning of the Pedigree Number field, resulting in a new field width of seven.

- **Decrease field width of item**

  Entry of **2** produces the following display:

```
Enter ">" or "<" plus number of characters to remove from LEFT or RIGHT.
Enter ">MAX" or "<MAX" to remove from all records as many blanks as needed
to remove LEADING or TRAILING blanks from the longest string in the field.
>
```

Field width may be decreased in two ways. Entry of **>**<n> (or **<**<n>) simply removes characters (including non-blank characters) from the left (or right) margin of the item selected. Example: entry of **>2** would shorten the Pedigree Number field to three characters by removing the two left-most characters. *Warning: Data may be lost in this mode*. On the other hand, entry of **>max** (or **<max**) removes only blank characters from the left (or right) margin of the field. This is done by finding that entry in the field having greatest number of non-blank characters, determining the number of leading (or trailing) blanks in this field, and then truncating the field by this number of characters on all records. This process reduces field width as much as possible, but prevents loss of non-blank data.

- **Subdivide an item**

  Entry of **3** produces the following display:

```
Subdivide Birth Date (YYYYMMDD)


::::|:::


Segment 1:
>4 < Enter number of characters (1-8) to be assigned to this segment.
```

In this example, the components Birth Date will be subdivided into three separate items (corresponding to **YYYY**, **MM**, and **DD**). A Status Line appears at the top, showing the label of the item to be subdivided. Below this is an Item Ruler in which the colon (:) symbol designates a character that has not yet been assigned to a subdivision (or **segment**). As a visual aid, every fifth unassigned character is marked with a stroke (|). The number of the segment to be created appears next, followed by a command line into which format information is entered. In this example, we create the first segment by assigning four integer characters and the mnemonic YEAR, as follows:

```
Subdivide Birth Date (YYYYMMDD)


::::|:::


Segment 1:
>4 < Enter number of characters (1-8) to be assigned to this segment.
>i< Enter the Data Type of the segment (C, I, D, or R).
>YEAR  < Enter a single mnemonic for this segment.
```

When entry for the first segment is completed, the following display appears:

```
Subdivide Birth Date (YYYYMMDD)

1
YEAR
____ ::::


Segment 2:
>2 < Enter number of characters (1-4) to be assigned to this segment.
```

The process is the same as that used in program ITEMIZE: the number and mnemonic of the newly created segment is displayed above underline characters (_) that form a segment ruler. The Item Ruler is reduced in size correspondingly, and the next segment number is displayed above a new command line. This pattern is followed until subdivisions of the original item are completely defined. The number of subdivisions must be no greater than the original field width of the item selected.

J Two or more items may be joined together with this command which produces the following display:

```
   1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ   16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE   12 EGO Permanent ID       19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE   13 Birth Date (YYYYMMDD  *20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  REFORMAT    Items to be joined              File - MASTER.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to items in the order they are to be joined into
a single item. Enter [C] to continue.
>21
```

Items selected are marked with asterisks in the usual way, and the new composite item is added at the end of the record. A label and a mnemonic set are created from a combination of the first mnemonics of the combined items. A maximum of 10 items can be joined into one; a maximum of 10 joined items may be produced.

B A field of blanks may be entered with this command, which produces a request for entry of field width, item label and mnemonics. Data type is automatically set at **C**.

K This command inserts a field containing a fixed string of characters. The string must be surrounded by single or double quotes, and its entry followed with entry of an item label and mnemonics. Data Type is automatically set at **C**.

R Entering this command produces a request for an integer seed (1 - 8 digits) for a pseudo-random number generator. An item containing a randomly generated 10-digit fraction is added at the end of the record. The seed is saved in the appropriate description field of the Code File **reformat.cde**.

S Entering an **s** adds to each record an item containing an integer (1-99999) corresponding to the record's position in the file

C Continues to confirmation step.

H Entry of an H produces the following help display pages:

```
REORDER items by entering the number corresponding to the item in the list,
in the order that items are to appear in the output record.

Right or left JUSTIFY items by preceding item number with ">" or "<".

Preceding item number with "U" (or "L") sets all alphabetic characters in
the selected field to UPPER (or LOWER) case.

Enter [C] to continue item selection. Enter "P" to page.
>
```

and

```
REFORMAT item by preceding item number with "F". This feature permits in-
crease or decrease of field width and subdivision of item.

A field of BLANKS may be entered with command "B". Field width, item label
and mnemonics are requested.

A CONSTANT (fixed field) of characters may be entered with the "K" command.
The string must be surrounded by single or double quotes. Item label and
mnemonics are requested.

A RANDOM fraction is added at the end of the record with the "R" command.
A 1-8 digit integer 'seed' is requested.

An ordered SEQUENCE number is appended to the record with the "S" command.

JOIN two or more items by entering this command and selecting items to be
joined from the redisplayed Item List.

Enter RETURN to continue selecting items.
>
```

Notes:

- Program REFORMAT permits items to be removed (simply by not selecting them) or duplicated (by repeated selection), as well as rearranged in the output record.

- The justification feature works for all data types, but is normally used only with character data (type C). Left justification of integer (type I) or real data (type R) should be avoided, as many analysis programs treat trailing blanks after numbers as zeros.

- Blanks and constants may be inserted in any position in the record.

- Data Type of items whose field width is changed remains unchanged if leading blanks are added , but are changed to **C** if trailing blanks are added. Changes to dates result in a Data Type change from **D** to **C**. Data Type of subdivided and joined items are changed to **C**.

- The **R** (random fraction) and **S** (integer sequence) features are identical to those in program CALC. These functions may be used only once during the current run of the program.

- The random number generator used is a FORTRAN implementation of a hardware-independent linear congruential algorithm described by R. Sedgewick (p. 37 in Algorithms, Addison-Wesley Publishing Company, Reading, Massachusetts, 1983).

- Insertion of blank or constant items may be repeated during the current run of REFORMAT.

- New items created by the **>**, **<**, **U**, **L**, **F**, **J**, **R** and **S** commands cannot be re-selected during the current run of REFORMAT.

- **Warning**: Data loss may occur when reducing field width.

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **reformat subset.out**.

Files:

Input: Any Master File or Data File, and an associated Code File. Recursive use of output files is permitted, that is, file **reformat.out** may be used as input to program REFORMAT.

Output: **reformat.out** contains all records in the input file in the order selected.

reformat.cde defines the record structure of **reformat.out**.

Program Limits:

A maximum of 500 items may be defined for any one PEDSYS record.

REPORT

Program REPORT produces listings that are formatted so that information from one file is represented as a title or **header** that separates lists of **data** taken from one or more other files. This format is particularly useful in displaying information about individuals taken from a Master File, followed by a list of records pertaining to EGO taken from linked Data Files.

Introductory Display (Selection of Header file):

```
PEDSYS program REPORT - Copyright 1992 SFBR

REPORT produces listings that are formatted so that information from one
file is presented as a title or header that separates lists of data taken
from one or more other files.

  1.  Unlinked File
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter number or name of file to use as HEADER information, "Q" to quit.
>2
```

A Standard File List is used to select the file from which **header** information is to be taken:

```
    1.  Example Master File
    2.  Markers + Quant. P'types
    3.  Blood Sample Inventory
    4.  Hemoglobin Levels

Enter number or name of file to use as HEADER information.
>1
```

Here, records from the Example Master File will be used as headers in the listing.

Selection of Data Files

Next, one or more files may be selected from which **data** pertaining to individuals listed in the headers is kept:

```
Select DATA Files

  1.  Unlinked File
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter number or name of file to use as DATA, "Q" to quit.
>2
```

and

```
    1.  Example Master File
*   2.  Markers + Quant. P'types
    3.  Blood Sample Inventory
    4.  Hemoglobin Levels

Enter the number of the files(s) from which DATA will be taken in the order
they are to be listed after the header.
>4
```

In the example shown here, data will be taken from the Markers + Quant. P'types file (single-entry) and the Hemoglobin Levels file (multiple-entry).

Because REPORT does much of its formatting to conform to a particular output device, the next action required is to select one of a series of output devices (specified in the Printer Control File DEVICES). After a file has been selected whose contents are to be listed, the following display appears:

```
Output from REPORT may be sent to the following output devices:

    [1]  Main Lab LaserWriterII
     2.  Main Lab Line Printer
     3.  Printer Room Sun LaserWriter
     4.  Console Screen
     5.  Output file "list.tab"

Enter number of device, or "Q" to quit without printing.
>
```

Records from the files are matched by searching for the identity of Key Items that are of the same width and Data Type. Program REPORT pre-defines EGO PID as the Key Item in the header records, but leaves open the choice of PID in unlinked data records (that is, the PID of EGO, Sire, Dam or Mate, etc. may be designated as the Key).

Selection of Header Items:

A Standard Item List is displayed for the file from which headers are taken:

```
   1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING   *15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ  *16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ  *17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE  *12 EGO Permanent ID       19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE  *13 Birth Date (YYYYMMDD  *20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ     *14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  REPORT      Items in the header             File - MASTER.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to HEADER items shown above in the order they are
to appear across the page. Enter "A" to list all items, [C] to continue to
next step.
>
```

Items will be printed in the order selected *with one exception*: EGO Permanent ID is pre-selected (reflecting its designation as header Key Item), and will always be printed in the left-most position of the header line. Standard Confirmation and Editing procedures follow.

If an unlinked Data File is chosen to supply header records, REPORT searches its Code File for the Standard Mnemonic **EGO** or **ID**. If not found, the program terminates with an error.

Selection of Data Items:

A Standard Item List is displayed separately for each file from which data records are taken, in the order the files are selected. In this example, the Item List from the Marker Phenotype file is displayed first:

```
   1 Ego ID                * 3 APOB Pvu2a Genotype   * 5 APOAI Level
 * 2 ADA  Phenotype        * 4 Tot. Ser. Cholestero
--------------------------------------------------------------------------------
  REPORT      Data items to be listed         File - QUANTGEN.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to DATA items shown above in the order they are
to appear across the page. Enter "A" to list all items, [C] to continue to
next step.
>
```

An Item List Display from the next Data File (if one has been selected) appears when items from the previous Data File have been chosen:

```
   1 Ego                   * 3 Hemoglobin Level
 * 2 Sample Date             4 INTERNAL POINTER
--------------------------------------------------------------------------------
  REPORT      Data items to be listed         File - HEMOGLOB.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to DATA items shown above in the order they are
to appear across the page. Enter "A" to list all items, [C] to continue to
next step.
>
```

Data items will be printed in the order selected in the next display:.

```
 [1]  Same order as found in Header File.
  2.  PIDs in ascending order
  3.  PIDs in descending order

Enter number corresponding to order that IDs should appear in list.
>
```

Data records are grouped according to the header Key Item (that is, EGO PID). If a linked Multiple-entry Data File has been chosen, the order that data records will appear within header groups in the output list may be selected from the following display:

```
 [1]  Most recently entered records first.
  2.  Most recently entered records last.

Enter number corresponding to order that DATA records should appear in list.
>
```

The choice of order here assumes that records in the file remain in their original sequence, that is, determined by time of entry. This order may be changed (a) by initially selecting the file from the **Unlinked** category in the File List Display when the program is first started and then sorting as described below.

Unlinked Data Files

If an unlinked Data File is chosen to supply data records, it will be necessary to designate the Permanent ID and to ensure that the file has been sorted with the major sort on this as the Key Item. The example below shows the Item List for an unlinked file, from which the Key Item is selected:

```
 + 1 ID                      3 Weight
 + 2 Sample Date             4 INTERNAL POINTER
--------------------------------------------------------------------------------
  REPORT      Select ID and sort items        File - WEIGHTS.XMP
--------------------------------------------------------------------------------
Select items from the list above to be sorted in order of decreasing sig-
nificance. First item selected MUST be a PID (Key Item). Precede item number
with a "D" to sort in descending order. Enter [C] to continue.
>
```

The first item selected here (the major sort) must be a PID. Usually (but not necessarily) this will be EGO's PID whose width and Data Type match that of the EGO Permanent ID pre-selected from the header input file. Selection of additional items from the list permits compound sorting of data records following each header record.

Requests for additional information are displayed before the program is run:

```
 [1] LINELIST - single-spaced data, headers separated by blank line.
  2. LINEFORM - three spaces per record, headers separated by blank line.
  3. PAGELIST - single-spaced data, headers starting on new page.
  4. PAGEFORM - three spaces per record, headers start on new page.

Select number corresponding to format of listing.
>
```

Notes:

- Sometimes the pre-defined order implicit in PEDSYS file linkage is not desirable. One way to overcome this order is to treat a linked Data File as an unlinked file, which forces PEDSYS programs to ignore relationships between records defined in the Master Pointer File, the Data Pointer File, and the internal pointers in any Multiple-entry Data File. This can be accomplished

by selecting the **Unlinked** category in the initial File List Display, and typing in the file name, rather than selecting the file from the numbered list.

- Creating a report that is based on several files can be tedious, since so many choices must be made in order to control the format of each group of records. If a particular report is to be run repeatedly on a production basis, it is a good idea to consider the use of a shell script (see Appendix E) so that the commands necessary to produce the report are stored for repeated use.

- Only one unlinked Data File may be used as a source for data records.

- The command line shortcut with *two* input files (program name, followed by a blank, header filename, another blank and the data filename) may be used to start this program. Example: **report sort.out subset.out**.

Files:

Input: Header: A Master File or Data File (with Code File) in which each records carries a PID.

Data: One or more Data Files (usually multiple-entry) and their associated Code Files.

The Device Control File, **DEVICES**.

Output: Printer, console screen, or disk file **report.tab**.

An example of REPORT output is given below:

```
HEADER FILE:   MASTER.XMP             15 Mar 1999       Page   1
DATA FILE(S):  QUANTGEN.XMP
               HEMOGLOB.XMP

================================================================
EGO    BIRTH       FA     MO     SEX  EXIT        PEDNO

ID061  1972/03/09               M    1998/04/20  1
================================================================

       ADA    APOB   TSC    APOAI
       PHEN   Pvu2          Level
       TYPE   GENO
              TYPE

       A       A1A2    103    113.0

      ------------------------------------------------------------

          DATE       hgb

       1982/03/24    14.7
       1983/01/12    17.0
       1984/10/04    17.5
       1985/10/15    19.1
```

(REPORT output example continued)

```
        1986/10/06    18.6
        1987/06/10    18.1
        1988/04/15    17.2
        1989/04/14    17.2
        1990/01/15    13.7
        1992/01/10    16.6
        1993/01/13    16.4
        1994/01/20    16.9
        1995/01/24    17.1

=================================================================
EGO    BIRTH       FA     MO     SEX  EXIT        PEDNO

ID056  1973/12/23               M               1
=================================================================

       ADA    APOB   TSC    APOAI
       PHEN   Pvu2          Level
       TYPE   GENO
              TYPE

               A2A2    104    115.3

      ----------------------------------------------------------

          DATE       hgb

       1982/05/10    14.2
       1983/09/21    14.6
       1984/10/02    15.9
       1985/11/04    16.0
       1986/11/06    16.4
       1987/10/27    15.4
       1988/10/12    14.9
       1989/09/12    16.1
       1990/01/12    13.8
       1992/05/08    14.9
       1993/10/12    15.0

=================================================================
EGO    BIRTH       FA     MO     SEX  EXIT        PEDNO

ID038  1974/05/05               M    1996/12/23  1
=================================================================

       ADA    APOB   TSC    APOAI
       PHEN   Pvu2          Level
       TYPE   GENO
              TYPE

       A       A2A2     94     83.1

      ----------------------------------------------------------
```

SEGCHECK

Program SEGCHECK tabulates offspring phenotypes (or genotypes) by parental mating type. Single-locus segregation tables may be constructed for offspring of (a) individual parents of one sex and their mates, or (b) all parents of one sex and their mates. Tabulation may be limited to parents of specific phenotypes.

SEGCHECK makes the initial assumption that the data are autosomal codominant *phenotypes* unless one of the last three mnemonic words associated with the Code File entry for a locus is "**GENO**" or "**G'TYPE**", in which case the data are interpreted as *genotypes*. See note below for procedures to follow if the phenotype selected is not a codominant autosomal.

Introductory Display:

```
PEDSYS program SEGCHECK - Copyright 1992 SFBR

SEGCHECK tabulates offspring phenotypes (or genotypes) by mating type.

NOTE: All loci are assumed to be either autosomal codominant phenotypes or
genotypes. Other modes of inheritance will yield invalid segregation tables.

  1.  Unlinked File
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>
```

Because SEGCHECK must extract data from multiple family records, the program requires the use of indexed IDs for Ego and Ego's parents and sibs. These conditions are fulfilled automatically if marker phenotypes are stored as a Linked Data File such as **QUANTGEN.XMP**.

If SEGCHECK is used when phenotypes are in an unlinked file (choice **1** or **2** above), the following instruction appears:

```
 1.  Unlinked File
 2.  Example Master File

Enter number or name of file containing pedigree information.
>
```

After the necessary files have been selected, the following display appears from which a single locus may be selected for tabulation:

```
   1 EGO Permanent ID        2 D1S1656 Genotypes       3  Sample No.
--------------------------------------------------------------------------------
  SEGCHECK    Select locus                    File - D1S1656
--------------------------------------------------------------------------------
Enter a number corresponding to the locus for which segregation tables are
to be constructed, or [C] to continue.
>2
```

Next, the configuration of tables to be constructed is selected:

```
Select type of segregation table to be constructed:

 1. Separate tables for fathers (identified by father's PID)
 2. Separate tables for mothers (identified by mother's PID)
 3. Separate tables for parental pairs (identified by both PIDs)
 4. Combined table for fathers with same genotype
 5. Combined table for mothers with same genotype

Enter number above corresponding to choice.
>4
```

Here, a table for sires by genotype has been selected. Tables will contain a row for each mating type, with columns representing offspring genotypes. If **3** (separate tables for parental pairs) is chosen, the following option is given:

```
Display grandparental genotypes?  Y/[N]
>
```

The following sequence then appears:

```
Tabulate separately by sex of offspring?  Y/[N]
>

Enter minimum number of offspring per mating
>1
```

If tables for individual parents are selected (choices **1** - **2** above), the following sequence appears, in which genotypic data for the selected parent (here, the sire) is defined:

```
Select type of segregation table to be constructed:

 1. Separate tables for fathers (identified by father's PID)
 2. Separate tables for mothers (identified by mother's PID)
 3. Separate tables for parental pairs (identified by both PIDs)
 4. Combined table for fathers with same genotype
 5. Combined table for mothers with same genotype

Enter number above corresponding to choice.
>1

Tabulate separately by sex of offspring?  Y/[N]
>n

Enter minimum number of offspring per sire
>2
```

```
Select father's genotype

 1. Heterozygous fathers only
 2. Homozygous fathers only
 3. Both types

Enter number above corresponding to choice.
>3

Exclude matings that do not reveal segregation?  [Y]/N
>
```

The program then reads through the Data File to determine which alleles are present:

```
The following alleles were found for D1S 1656 G'TYPE

   A       B       C       D       E

Enter RETURN to include genotypes based on all of the above, "E" to select
a subset, "Q" to quit without executing
>
```

If an **e** is entered, an opportunity is given to select a subset of the alleles shown. If the Data File contains inferred phenotypes or genotypes, the following option is given:

```
 1.  Tabulate without distinguishing between known and inferred genotypes.
 2.  Tabulate known and inferred genotypes separately in same table.
 3.  Exclude inferred genotypes from tables.

Enter number of choice above.
>
```

SEGCHECK then proceeds to construct segregation tables.

Notes:

- SEGCHECK requires unambiguous representation of zygosity, which means that tabulation is limited to (a) codominant autosomal phenotypes or (b) genotypes. Segregation tables for dominant/recessive and X-linked phenotypes can be generated, however, by transforming these representations to their genotypic forms with program INFER. File **infer2.out** can then be used as unlinked input to SEGCHECK.

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **segcheck subset.out**.

Error Checking:

Error detection in SEGCHECK is limited to identifying discrepancies between parental and offspring phenotypes (or genotypes). When such a discrepancy is found, a record is written in file **segcheck.tab** containing PIDs and phenotypes.

Algorithm: R. H. King, L. Freeman-Shade

Files:

Input: A Data File with records containing genotypic or phenotypic data, with associated Code File, and a source of pedigree information (typically, a Master File). The Data File need not be linked.

Output: **segcheck.tab** contains segregation tables as specified when program is run. Some examples of the content of this file are shown below:

Example 1. Table configuration choice **1** (separate tables for fathers identified by PID, tabulation without regard to offspring sex, minimum 2 offspring per father, all father's genotypes, non-informative matings excluded).

```
Segregation of TYPED FATHERS with at least 2 offspring

MATINGS THAT SHOW SEGREGATION, INFERRED PHENOTYPES INCLUDED

LOCUS:   ADA    PHENO  TYPE
ALLELES: A       B       C


Father: ID063

                    | Offspring genotypes:
Father   Mother     |   A/A     A/B     A/C     B/B     B/C     C/C   UNKNWN  TOTAL
-------------------------------------------------------------------------------------
A/A      A/B        |    1       2       0       0       0       0       0       3
-------------------------------------------------------------------------------------
TOTAL               |    1       2       0       0       0       0       0       3


Father: ID109

                    | Offspring genotypes:
Father   Mother     |   A/A     A/B     A/C     B/B     B/C     C/C   UNKNWN  TOTAL
-------------------------------------------------------------------------------------
A/B      A/B        |    0       1       0       0       0       0       0       1
A/B      B/B        |    0       0       0       1       0       0       0       1
-------------------------------------------------------------------------------------
TOTAL               |    0       1       0       1       0       0       0       2
```

Example 2. Table configuration choice **2** (separate tables for mothers identified by PID, tabulation by sex of offspring, minimum 1 offspring per mother, heterozygous mothers only, informative matings only).

```
Segregation of ALL HETEROZYGOUS MOTHERS with at least 1 offspring

MATINGS THAT SHOW SEGREGATION, INFERRED PHENOTYPES INCLUDED

LOCUS:   ADA    PHENO  TYPE
ALLELES: A       B       C

Mother: ID034

                    | Offspring genotypes:
Mother   Father     |   A/A     A/B     A/C     B/B     B/C     C/C   UNKNWN  TOTAL
-------------------------------------------------------------------------------------
A/B      UNKNWN   M|    0       0       0       1       0       0       0       1
                  F|    1       0       0       0       0       0       0       1
                  U|    0       0       0       0       0       0       0       0
-------------------------------------------------------------------------------------
TOTAL              M|    0       0       0       1       0       0       0       1
TOTAL              F|    1       0       0       0       0       0       0       1
TOTAL              U|    0       0       0       0       0       0       0       0

Mother: ID042

                    | Offspring genotypes:
Mother   Father     |   A/A     A/B     A/C     B/B     B/C     C/C   UNKNWN  TOTAL
-------------------------------------------------------------------------------------
A/B      A/A      M|    1       0       0       0       0       0       0       1
                  F|    0       1       0       0       0       0       0       1
                  U|    0       0       0       0       0       0       0       0
-------------------------------------------------------------------------------------
TOTAL              M|    1       0       0       0       0       0       0       1
TOTAL              F|    0       1       0       0       0       0       0       1
TOTAL              U|    0       0       0       0       0       0       0       0

Mother: ID065

                    | Offspring genotypes:
Mother   Father     |   A/A     A/B     A/C     B/B     B/C     C/C   UNKNWN  TOTAL
-------------------------------------------------------------------------------------
A/B      A/A      M|    0       1       0       0       0       0       0       1
                  F|    1       2       0       0       0       0       0       3
                  U|    0       0       0       0       0       0       0       0
-------------------------------------------------------------------------------------
TOTAL              M|    0       1       0       0       0       0       0       1
TOTAL              F|    1       2       0       0       0       0       0       3
TOTAL              U|    0       0       0       0       0       0       0       0
```

Example 2. (continued)

```
Mother: ID103

                    | Offspring genotypes:
Mother   Father     |   A/A     A/B     A/C     B/B     B/C     C/C   UNKNWN  TOTAL
-------------------------------------------------------------------------------------
A/B      A/B      M|    0       0       0       0       0       0       0       0
                  F|    0       1       0       0       0       0       0       1
                  U|    0       0       0       0       0       0       0       0
-------------------------------------------------------------------------------------
TOTAL             M|    0       0       0       0       0       0       0       0
TOTAL             F|    0       1       0       0       0       0       0       1
TOTAL             U|    0       0       0       0       0       0       0       0


Mother: ID113

                    | Offspring genotypes:
Mother   Father     |   A/A     A/B     A/C     B/B     B/C     C/C   UNKNWN  TOTAL
-------------------------------------------------------------------------------------
A/B      B/B      M|    0       0       0       0       0       0       0       0
                  F|    0       0       0       1       0       0       0       1
                  U|    0       0       0       0       0       0       0       0
-------------------------------------------------------------------------------------
TOTAL             M|    0       0       0       0       0       0       0       0
TOTAL             F|    0       0       0       1       0       0       0       1
TOTAL             U|    0       0       0       0       0       0       0       0
```

Example 3. Table configuration choice **3** (separate tables for parental pairs, grandparental phenotypes displayed, tabulation without regard to offspring sex, minimum 2 offspring of both sexes, informative matings only, inferred types included).

```
Segregation in FULL SIBSHIPS of at least 2 individuals

MATINGS THAT SHOW SEGREGATION, INFERRED PHENOTYPES INCLUDED

LOCUS:   ADA    PHENO  TYPE
ALLELES: A       B       C




Mother    ID034     Phenotype: AB        |  Father    ID045     Phenotype:
                                          |
Mo's FA:  ID055     Phenotype: A         |  Fa's FA:  unk
Mo's MO:  ID065     Phenotype: AB        |  Fa's MO:  unk

                    | Offspring genotypes:
                    |   A/A     A/B     A/C     B/B     B/C     C/C   UNKNWN  TOTAL
-------------------------------------------------------------------------------------
TOTAL               |    1       0       0       1       0       0       0       2
-------------------------------------------------------------------------------------
```

Example 3. (continued)

```
Mother    ID065     Phenotype: AB        |  Father    ID063     Phenotype: A
                                         |
Mo's FA:  ID061     Phenotype: A         |  Fa's FA:  ID038     Phenotype: A
Mo's MO:  ID042     Phenotype: AB        |  Fa's MO:  ID042     Phenotype: AB

                  | Offspring genotypes:
                  |   A/A     A/B     A/C     B/B     B/C     C/C   UNKNWN  TOTAL
-------------------------------------------------------------------------------------
TOTAL             |    1       2       0       0       0       0       0       3
-------------------------------------------------------------------------------------
```

Example 4. Table configuration choice **4** (combined table of fathers with the same genotype, tabulation without regard to offspring sex).

```
Segregation in Offspring of ALL FATHERS by FATHER'S GENOTYPE

ALL MATINGS, INFERRED PHENOTYPES INCLUDED

LOCUS:   ADA    PHENO  TYPE
ALLELES: A       B       C

                  | Offspring genotypes:
Father   Mother   |   A/A     A/B     A/C     B/B     B/C     C/C   UNKNWN  TOTAL
-------------------------------------------------------------------------------------
A/A      A/B      |    2       4       0       0       0       0       0       6
A/A      B/B      |    0       2       0       0       0       0       0       2
-------------------------------------------------------------------------------------
TOTAL             |    2       6       0       0       0       0       0       8

                  | Offspring genotypes:
Father   Mother   |   A/A     A/B     A/C     B/B     B/C     C/C   UNKNWN  TOTAL
-------------------------------------------------------------------------------------
A/B      A/B      |    0       1       0       0       0       0       0       1
A/B      B/B      |    0       0       0       1       0       0       0       1
A/B      UNKNWN   |    0       0       0       1       0       0       0       1
-------------------------------------------------------------------------------------
TOTAL             |    0       1       0       2       0       0       0       3

                  | Offspring genotypes:
Father   Mother   |   A/A     A/B     A/C     B/B     B/C     C/C   UNKNWN  TOTAL
-------------------------------------------------------------------------------------
B/B      A/B      |    0       0       0       1       0       0       0       1
B/B      B/B      |    0       0       0       0       0       0       2       2
-------------------------------------------------------------------------------------
TOTAL             |    0       0       0       1       0       0       2       3
```

Example 4. (continued)

		Offspring genotypes:							
Father	Mother	A/A	A/B	A/C	B/B	B/C	C/C	UNKNWN	TOTAL
UNKNWN	A/B	1	0	0	1	0	0	0	2
UNKNWN	B/B	0	0	0	1	0	0	0	1
UNKNWN	C/C	0	0	0	0	0	0	1	1
UNKNWN	UNKNWN	5	1	0	5	0	1	2	14
TOTAL		6	1	0	7	0	1	3	18

Example 5. Table configuration choice **5** (combined table of fathers with the same genotype, tabulation by sex of offspring).

Segregation in Offspring of ALL MOTHERS by MOTHER'S GENOTYPE

ALL MATINGS, INFERRED PHENOTYPES INCLUDED

LOCUS: ADA PHENO TYPE
ALLELES: A B C

			Offspring genotypes:							
Mother	Father		A/A	A/B	A/C	B/B	B/C	C/C	UNKNWN	TOTAL
A/B	A/A	M	1	1	0	0	0	0	0	2
		F	1	3	0	0	0	0	0	4
		U	0	0	0	0	0	0	0	0
A/B	A/B	M	0	0	0	0	0	0	0	0
		F	0	1	0	0	0	0	0	1
		U	0	0	0	0	0	0	0	0
A/B	B/B	M	0	0	0	0	0	0	0	0
		F	0	0	0	1	0	0	0	1
		U	0	0	0	0	0	0	0	0
A/B	UNKNWN	M	0	0	0	1	0	0	0	1
		F	1	0	0	0	0	0	0	1
		U	0	0	0	0	0	0	0	0
TOTAL		M	1	1	0	1	0	0	0	3
TOTAL		F	2	4	0	1	0	0	0	7
TOTAL		U	0	0	0	0	0	0	0	0

Example 5. (continued)

Mother	Father		Offspring genotypes: A/A	A/B	A/C	B/B	B/C	C/C	UNKNWN	TOTAL
B/B	A/A	M	0	1	0	0	0	0	0	1
		F	0	1	0	0	0	0	0	1
		U	0	0	0	0	0	0	0	0
B/B	A/B	M	0	0	0	1	0	0	0	1
		F	0	0	0	0	0	0	0	0
		U	0	0	0	0	0	0	0	0
B/B	B/B	M	0	0	0	0	0	0	1	1
		F	0	0	0	0	0	0	1	1
		U	0	0	0	0	0	0	0	0
B/B	UNKNWN	M	0	0	0	0	0	0	0	0
		F	0	0	0	1	0	0	0	1
		U	0	0	0	0	0	0	0	0
TOTAL		M	0	1	0	1	0	0	1	3
TOTAL		F	0	1	0	1	0	0	1	3
TOTAL		U	0	0	0	0	0	0	0	0

Mother	Father		Offspring genotypes: A/A	A/B	A/C	B/B	B/C	C/C	UNKNWN	TOTAL
C/C	UNKNWN	M	0	0	0	0	0	0	0	0
		F	0	0	0	0	0	0	1	1
		U	0	0	0	0	0	0	0	0
TOTAL		M	0	0	0	0	0	0	0	0
TOTAL		F	0	0	0	0	0	0	1	1
TOTAL		U	0	0	0	0	0	0	0	0

Mother	Father		Offspring genotypes: A/A	A/B	A/C	B/B	B/C	C/C	UNKNWN	TOTAL
UNKNWN	A/B	M	0	0	0	1	0	0	0	1
		F	0	0	0	0	0	0	0	0
		U	0	0	0	0	0	0	0	0
UNKNWN	UNKNWN	M	5	0	0	1	0	0	2	8
		F	0	1	0	4	0	1	0	6
		U	0	0	0	0	0	0	0	0
TOTAL		M	5	0	0	2	0	0	2	9
TOTAL		F	0	1	0	4	0	1	0	6
TOTAL		U	0	0	0	0	0	0	0	0

SETDATA

SETDATA is the second program of the two-step process required to make batch changes to PEDSYS Data Files. The program uses information from a special-purpose output file named **<filename>.UPD** which has been produced by program ALTERDAT.

Introductory Display:

```
PEDSYS program SETDATA - Copyright 1992, 1999 SFBR

SETDATA is the second program of the two-step process required to make batch
changes to PEDSYS Data Files.  The program requires output from ALTERDAT in
the form <filename>.UPD or <filename>.DEL where <filename> is the file to be
modified.

  1.  Unlisted File
  2.  EXAMPLE

Enter number corresponding to the file or file group to be altered.
>
```

Confirmation Displays:

If changes are to made to a Data File that already exists, the following display will appear when the file has been selected (using the Example Marker Phenotype file as an example):

```
Data File MARKERS.XMP will be updated, but the file as it was before
changes were made will be saved under the name MARKERS.OLD.  In addition,
the update file, MARKERS.UPD will be renamed MARKERS.OUP after the program
has finished.

Enter "C" to CONTINUE, or "X" to EXIT program without updating Data File.
>
```

If a new Data File is to be created from information stored in **<filename>.UPD** the following display will appear:

```
Data File NEWFILE.XMP will be created, and update file NEWFILE.UPD will be
renamed NEWFILE.OUP.

Enter "C" to continue, or "X" to EXIT program without creating Data File.
>
```

In this example, Data File **NEWFILE.XMP** has been specified in the File Selection sequence.

Error Messages and Displays:

Program SETDATA is usually run as the last step in permanently updating or creating PEDSYS Data Files. Considerable effort has gone into preventing errors from being incorporated into the data. Protection from permanent damage is provided at two levels:

Level 1. Errors that affect single records are recorded in the **setdata.err** file, and the record in question is not updated.

Level 2. Backup copies of original input files are created so that it is always possible to discard incorrect changes and return to an earlier version of the data.

Here are the error messages that may appear in **setdata.err**, and the measures that should be taken to correct them. For convenience, errors are grouped into three classes:

Group 0. (Unprintable Characters)

In order to prevent unprintable characters from being added to the Data File records, each new entry in **<filename>.UPD** is scanned for characters with ASCII decimal code values less than 32 or greater than 126. The following errors may be reported if any of these values is found:

```
<ego>: Record not added; unprintable character (ASCII <nasc>) found.          (01)
<ego>: <mne> not changed; contains unprintable character (ASCII <nasc>).      (02)
```

where <ego> is the PID of the record to be updated in **<filename>.UPD** , <nasc> is the ASCII code corresponding to the unprintable character in the **<filename>.UPD** record, and <mne> is a mnemonic identifying the item in file **<filename>.UPD**.

Error 01 may occur when adding a new individual, Error 02 when making changes to a record already in the Data File. Probable causes of these errors are (a) an incorrect key combination hit during data entry, (b) disk error, or (c) a noisy communication line during data transfer.

To correct Error 01, re-enter entire record using ALTERDAT in ADD mode. To correct Error 02, re-enter item using ALTERDAT in EDIT mode. Re-run SETDATA using the most recent Update and Data Files (see note on renaming and deleting these files).

Group 1. (Mismatched Mnemonics)

A check is done to assure that mnemonics identifying items in **<filename>.UPD** correspond to those in the Code File associated with the Data File. The following error is reported if the update mnemonics do not match the mnemonic fields in the appropriate Descriptor Record of **<filename>.CDE**:

```
<ego>: Entry not changed; <mne1> <mne2> <mne3> <mne4> not valid mnemonic.      (11)
```

where <ego> is the PID of the record to be updated in **<filename>.UPD**, and <mne1>, etc. are mnemonic words identifying the item in file **<filename>.UPD**.

This error may occur either when adding a new individual, or when making changes to a record previously entered in the Data File. Probable cause is a change made to **<filename>.cde** since the time it was used with ALTERDAT.

The most convenient way to correct this error if it has occurred infrequently, is to use program ALTERDAT to re-enter the changes that failed, and then re-run SETDATA, using the same Code File with both programs. If the error has occurred many times, it may be easier to start over, re-running SETDATA using the correct Code File and the previous Update and Data Files. Since these files are now named **<filename>.OUP** and **<filename>.OLD**, respectively, they will have to be given their original names **(<filename>.UPD** and **<filename>.<ext>**, respectively) before they can be used with SETDATA.

Group 2. (PID Conflicts)

Changes and additions to Data Files are prohibited if errors or conflicts are found in Permanent IDs (PIDs) of records in the Data File containing records to be changed. These errors are reported as follows:

```
<ego>: Not added; record already exists in Data File.                    (21)
<ego>: Not found in Data File; entry not changed.                        (22)
```

where <ego> is the PID of the record to be updated in **<filename>.UPD** .

Error 21 may occur when adding new data to a Single-entry Data File; Error 22 occurs when making changes to data in either Single-entry or Multiple-entry Data Files. Probable cause of Error 21 is double entry of data in a foreign file that is converted to PEDSYS format. Probable cause of Error 22 is prior deletion of a data record which might occur if program DELRECS has been run before program SETDATA.

Changes and additions to Data Files are prohibited also if errors or conflicts are found in Permanent IDs (PIDs) of records in the Master File associated with the Data File containing records to be changed. These errors are reported as follows:

```
<ego>: Not found in Master File; record not added to Data File.          (23)
<ego>: Not found in Master File; entry not changed.                      (24)
```

where <ego> is the PID of the record to be updated in <FILENAME>.UPD.

Error 23 may occur when adding data, Error 24 when modifying data in either Single-entry or Multiple-entry Data Files. Probable cause of either error is prior deletion of a Master File record between running ALTERDAT and SETDATA.

Changes and additions to Multiple-entry Data Files are prohibited if errors or conflicts are found in the internal pointers of records in the Data File containing records to be changed. These errors are reported as follows:

```
<ego>: Entry not changed; no internal pointer in update record.          (25)
<ego>: Entry not changed; internal pointer not consistent with PID.      (26)
```

where <ego> is the PID of the record to be updated in <FILENAME>.UPD.

Probable cause of Error 25 is incorrect specification of Data File Type (Single- or Double-entry) in the Directory File. Probable cause of Error 26 is prior deletion of a data record which might occur if program DELRECS has been run before program SETDATA.

Notes:

- The number of each error appears in parentheses at the end of the error record.
- All errors are best corrected with program ALTERDAT.

Files:

Input: Any PEDSYS Data File and its associated Code File.

<filename>.UPD produced by program ALTERDAT containing changes or additions to Data File records.

MPOINT.<EXT> and **DPOINT.<EXT>** are required if new records are added to an existing Data File which is linked to a Master File.

Output: **<filename>.<ext>** is the new or updated Data File produced by the program.

<filename>.OLD is a backup copy of the input Data File produced if **<filename>.<ext>** is generated as the result of changes and/or additions to the input Data File. **<filename>.OLD** is not produced if **<filename>.<ext>** is newly created.

<filename>.OUP is a backup copy of **<filename>.UPD**

setdata.log contains a record of each change that was successfully completed by the program.

setdata.err contains a record and diagnosis of changes that were attempted, but were unsuccessfully completed because of an error or inconsistency detected by the program.

DPOINT.<EXT> (the Data Pointer File) is written when the Data File is updated. This will be a modified version of the previously created file.

DPOINT.OLD is a backup copy of the previous version of **DPOINT.<EXT>.**

SETMASTR

SETMASTR is a second program of the two-step process required to make changes in Master Files. The program uses information from a special-purpose output file named **UPDATE.<EXT>** which has been produced by program ALTMASTR.

Introductory Display:

```
PEDSYS program SETMASTR - Copyright 1992, 1999 SFBR

SETMASTR updates Master Files from changes stored in the special-purpose
output file UPDATE.<ext> produced by program ALTMASTR.  The following
files are used as input to the program:

  CODE.<ext>   always required.
  UPDATE.<ext> "      "

  MASTER.<ext> required if changes are made to an existing Master File.
  MPOINT.<ext> "           "                      "        "      "

  DPOINT.<ext> is required if new records are added to a Master File
  to which Data Files have already been linked.

Enter - [C] to make CHANGES to an existing Master File
         S  to START new Master File
         Q  to QUIT program
>
```

Commands:

C Changing an existing Master File

Changing a preexisting Master File always requires an up-to-date Master Pointer File; if new records are to be added to a Master File that is already linked to one or more Data Files, an up-to-date Data Pointer File is also required. When this command is given, the following display appears:

```
 1.  Example Master File

Enter the number corresponding to the file to change.
>1


MASTER.XMP will be updated, but the file as it was before changes were made
will be saved under the name MASTER.OLD.  File UPDATE.XMP will be renamed
UPDATE.OLD. Enter [C] to continue, "Q" to quit without updating file.
>
```

Upon entry of **c**, the program begins execution.

S Starting a new Master File

Starting a new Master File from information stored in UPDATE.<EXT> with SETMASTR uses some of the same procedures as program DBFILER. That is, it will create and configure a Directory File DBFILES if one does not already exist. This commands results in the following display:

```
File UPDATE.<ext> will be used to create a new MASTER.<ext>, where <ext> is
a three-character file extension used as a population identifier.

>DOG<- Enter the identifier for this population.

MASTER.DOG will be created and file UPDATE.DOG will be renamed UPDATE.OLD
Enter [C] to continue, "Q" to quit without creating file.
>

DBFILES has been created

Assign individuals that have neither parents nor offspring

 1. All to the same pedigree (Pedigree 0)
[2] Each to a separate pedigree.

Enter number corresponding to choice
>1

>Beagles                  <- Enter population label for MASTER.DOG

>This is a Dog Master File<- Enter file label for MASTER.DOG
```

Here, a Master File Descriptor Record with extension **.DOG**, population label **Beagles**, and file label **This is a Dog Master File** will be added to (a newly created) **DBFILES**. Note that passwords have not been added -- this will have to be done with the edit feature of DBFILER.

If SETMASTER has been run previously, the following display which controls the disposition of a previously created log file may appear:

```
setmastr.log already exists.

Enter - D to DELETE Log File
        A to APPEND to current Log File
        X to EXIT program
>
```

Error Messages and Displays:

Program SETMASTR is usually run as the last step in permanently updating or creating Master Files. Because its output plays such a critical part in the organization of pedigree databases, considerable effort has gone into preventing errors from being incorporated into the data. Protection from permanent damage is provided at three levels:

Level 1. Errors that affect single records are recorded in the **setmastr.err** file, and the record in question is not updated.

Level 2. Whenever possible, the program is stopped before the final creation of new files when errors are found that might affect file structure.

Level 3. Backup copies of original input files are created so that it is always possible to discard incorrect changes and return to an earlier version of the data.

Here are the error messages that may appear in file **setmastr.err**, and the measures that should be taken to correct them. For convenience, errors are grouped into seven classes:

Group 0. (Unprintable characters)

In order to prevent unprintable characters from being added to the Master File records, each new entry in **UPDATE.<EXT>** is scanned for characters with ASCII decimal code values less than 32 or greater than 126. The following errors may reported if any of these values is found:

```
<ego>: Record not added; unprintable character (ASCII <nasc>) found.                (01)
<ego>: <mne1> <mne2> <mne3> <mne4> not changed; contains unprintable character ASCII
       <nasc>.                                                                      (02)
```

where <nasc> is the ASCII code corresponding to the unprintable character, and <mne1>, etc. are mnemonics identifying the item, in the **UPDATE.<EXT>** record

Error 01 may occur when adding a new individual, Error 02 when making changes to a record already in the Master File. Probable causes of error are (a) incorrect key combination hit during data entry, (b) disk error, or (c) noisy communication line during data transfer.

To correct Error 01, re-enter entire record using ALTMASTR in ADD mode. To correct Error 02, re-enter item using ALTMASTR in ALTER mode. Re-run SETMASTR using the most recent Update and Master files (see note on renaming and deleting these files).

Group 1. (Mismatched Mnemonics)

A check is done to assure that mnemonics identifying items in **UPDATE.<EXT>** correspond to those in the Code File associated with the Master File. The following error is reported if update mnemonics do not match mnemonic fields in the appropriate Descriptor Record of **CODE.<EXT>**:

```
<ego>: Entry not changed; <mne1> <mne2> <mne3> <mne4> not valid mnemonics.          (11)
```

where <mne1>, etc. are mnemonics identifying the item in file **UPDATE.<EXT>.** This error may occur either when adding a new individual, or when making changes to a record previously entered in the Master File. Probable cause is a change made to **CODE.<EXT>** since the time it was used with ALTMASTR.

To correct the error, re-run SETMASTR using the previous Update and Master Files with the appropriate Code File.

Group 2. (Duplicate Entries)

To prevent addition of duplicate entries for the same individual, the Master File is searched for EGO PID. If not found, records in **UPDATE.<EXT>** are searched from the beginning to the most recent

update to determine whether a new record for EGO has already been entered during this session. The error is reported if EGO is found in either list.

```
<ego>: Record not added; EGO already present.                    (21)
```

This error occurs only when adding a new individual to the Master File. Probable causes are (a) incorrect choice of ADD mode in ALTMASTR, (b) incorrect keyboard entry of EGO PID, or (c) inadvertent double entry of the same individual.

To correct the error: No action is required if (c), providing that all items in both entries are identical. Otherwise, re-enter information for new individual using correct EGO PID (use ALTMASTR in ALTER mode). Re-run SETMASTR using most recent Update and Master Files (see note on renaming and deleting these files).

Group 3. (Missing entry in Master File)

There must be a record in the Master File for every PID entered in the Update File (that is, EGO, SIRE, or DAM) for which items are to be changed. To assure this, the Master File (and records in **UPDATE.<EXT>** that have already been processed) are searched for the relevant PID. The error is reported if the PID is not found in either list. Errors reported may be:

```
<ego>: <mne1> <mne2> <mne3> <mne4>  not changed; EGO is not in MF.   (31)
<ego>: SIRE PID not changed; <newsire> is not in MF.                 (32)
<ego>: DAM PID not changed; <newdam> is not in MF.                   (33)
```

where <mne1>, etc. are mnemonics identifying the item to be changed, <newsire> is EGO's SIRE PID, and <newdam> is EGO's DAM PID, all as given in the **UPDATE.<EXT>** record.

These errors occur only when making changes to a record previously entered in the Master File. Probable causes are (a) the PID appears neither in the Master File, nor in **UPDATE.<EXT>** prior to this attempted change, or (b) an incorrect PID in **UPDATE.<EXT>**, presumably at time of keyboard entry.

To correct the error: If (a), first enter record for missing individual using ALTMASTR in ADD mode, and then re-enter the change in EGO's record using ALTMASTR in ALTER mode. If (b), re-enter change with correct PID using ALTMASTR in ALTER mode. Re-run SETMASTR using most recent Update and Master Files (see note on renaming and deleting these files).

Group 4. (Inconsistent Gender Assignments)

Before making changes to parental PIDs, SETMASTR verifies that the sex of individuals listed as parents in the Update File is consistent with their sex as it appears in the Master File. This check is done by searching the Master File (and records in UPDATE.<EXT> that have already been processed) for the appropriate parental PID. Errors are reported as follows if sex assignments are found to be inconsistent:

```
<ego>: SIRE PID not changed; <newsire> listed as female in MF.       (41)
<ego>: DAM PID not changed; <newdam> listed as male in MF.           (42)
<ego>: <parntyp> not assigned; <parntid> listed as <sex> in MF.      (43)
```

where <newsire> is the SIRE PID, <newdam> is the DAM PID, and <parntyp> is the type of parent (i.e., SIRE or DAM), as given in the **UPDATE.<EXT>** file, and <parntid> is the PID and <sex> is the sex (i.e., M or F) of the parent as it appears in the Master File.

These errors occur only when making changes to a record previously entered in the Master File. Probable causes are (a) incorrect keyboard entry of parental PID, (b) incorrect assignment of SEX to the

individual (who is EGO's parent) in the Master File, or (c) incorrect assignment of SEX to the individual (who is EGO's parent) in **UPDATE.<EXT>**.

To correct the errors: If (a), re-enter correct parental PID. If (b), first change SEX of the individual in the Master File, then re-enter change of parental PID. If (c), change SEX of the individual in the update file. In any case, use ALTMASTR in ALTER mode and re-run SETMASTR using the most recent Update and Master files (see note on renaming and deleting these files).

Important Warning: The following messages (Groups 5 - 7) result from errors in the Master File detected as the result of checking for the consistency of family structure before making the changes specified in **UPDATE.<EXT>**. These errors tend to be complex, and may indicate serious problems with the data. It is very important to examine closely the families in which these errors occur before assuming that changes have been made correctly.

As an example of what can go wrong, consider the case in which assignment of both parental PIDs is found to be incorrect, and ALTMASTR is used to delete these PIDs from EGO's record. If SETMASTR finds no inconsistencies in the family as it is determined from the DAM's perspective, EGO's DAM PID will be erased and the number of the DAM's offspring (NKIDS) will be reduced by one. If, however, there is an error in the family from the SIRE's perspective (EGO is not listed among SIRE's offspring, for instance), SETMASTR will not remove EGO's SIRE PID, nor will it reduce the SIRE's NKIDS by one. Unless proper steps are taken, the resulting inconsistent tabulation of NKIDS for the two parents can persist in the Master File even after subsequent runs of ALTMASTR and SETMASTR.

What must be done when errors 51 - 74 are reported is to run program INDEX at some point after corrections are made with ALTMASTR and SETMASTR. INDEX will re-compute NKIDS and re-establish all family links.

Group 5. (Conflict in Gender Assignment)

SETMASTR will not allow change of SEX assignment for EGO already present in the Master File if (a) EGO is listed as a parent in other Master File records, or (b) the record of the individual listed as EGO's first offspring does not in turn list EGO as a parent. If these conditions are found (by cross-checking PIDs of family members in the Master File), the following errors may be reported:

```
<ego>: SEX not changed; EGO listed as <parntyp> of KID1 <kid1> in MF.          (51)
<ego>: SEX not changed; EGO not listed as parent of KID1 <kid1> in MF.         (52)
```

where <parntyp> is the type (i.e., SIRE or DAM) of EGO's parent, and <kid1> is the PID of this parent's first offspring, all as given in the Master File.

These errors occur only when making changes to a record previously entered in the Master File. Probable causes are (a) incorrect keyboard entry of EGO PID or SEX, or (b) incorrect assignment of parental PID to EGO or <kid1> in the Master File.

To correct these errors: If (a), re-enter correct EGO PID or SEX. If (b), first re-assign parental PID of EGO or <kid1> in Master File, then re-enter change of EGO's SEX. In either case, use ALTMASTR in ALTER mode and re-run SETMASTR using the most recent Update and Master files (see note on renaming and deleting these files).

Group 6. (Incorrect parental assignment)

Parental PIDs cannot be deleted or changed if (a) the parent does not list EGO as an offspring in the Master File, or (b) the parent type (i.e, SIRE or DAM) assumed for the new entry is not consistent with the parent type as found in the Master File. If these conditions are found (by cross-checking PIDs of

family members in the Master File), the following errors may be reported:

```
<ego>: SIRE PID not deleted; EGO not found as offspring of <sire> in MF.          (61)
<ego>: DAM PID not deleted; EGO not found as offspring of <dam> in MF.            (62)
<ego>: SIRE PID not deleted; <sire> listed as DAM of KID1 <kid1> in MF.           (63)
<ego>: SIRE PID not changed; <dam> listed as SIRE of KID1 <kid1> in MF.           (64)
<ego>: SIRE PID not changed; <ego> cannot be own parent.                          (65)
<ego>: DAM PID not deleted; <ego> cannot be own parent.                           (66)
```

where <sire> and <dam> are PIDs of EGO's parents, and <kid1> is the PID of the first offspring of the parent, all as listed in the Master File.

These errors occur only when making changes to a record previously entered in the Master File. Probable causes are (a) incorrect assignment of parental PID to EGO in the Master File, (b) incorrect assignment of parental PID to <kid1> in the Master File, or (c) incorrect assignment of SEX to parent of EGO or <kid1> as given in the Master File.

To correct these errors: If (a), re-assign correct parental PID to EGO in Master File. If (b), first reassign correct parental PID to <kid1> in Master File, then re-enter change of parental PID for EGO. If (c), first re-assign SEX of appropriate parent, then re-enter change of parental PID for EGO. In all cases, use ALTMASTR in ALTER mode and re-run SETMASTR using the most recent Update and Master files (see note on renaming and deleting these files). Run INDEX after all changes have been made successfully.

Group 7. (Birth Date changes)

A change in birth date may change the order by which individuals are arranged in a sibship. Reordering of sibships requires that parents of sibs be identified correctly, and that parental sex is assigned properly. A change in BIRTH date is not permitted if (a) EGO's parent (as listed in the Master File) does not list EGO as an offspring, or (b) the type of this parent (i.e., SIRE or DAM) listed for EGO is not the same type as listed for its other offspring in the Master File. If these conditions are found (by cross-checking PIDs of family members in the Master File), the following errors may be reported:

```
<ego>: BIRTH not changed; EGO not found as offspring of <sire> in MF.            (71)
<ego>: BIRTH not changed; EGO not found as offspring of <dam> in MF.             (72)
<ego>: BIRTH not changed; DAM <dam> listed as SIRE of KID1 <kid1> in MF.         (73)
<ego>: BIRTH not changed; SIRE <sire> listed as DAM of KID1 <kid1> in MF.        (74)
```

where <sire> and <dam> are PIDs of EGO's parents, and <kid1> is the PID of the first offspring of the parent, all as given in the Master File.

These errors occur only when making changes to a record previously entered in the Master File. Probable causes are (a) incorrect assignment of parental PID to EGO or <kid1> in the Master File, or (b) incorrect assignment of parental PID to <kid1> in the Master File.

To correct these errors: If (a), first re-enter correct parental PID for EGO or <kid1>, then re-enter change of EGO's BIRTH. If (b), first change assignment of <kid1>'s parent in the Master File, then re-enter change of EGO's BIRTH. In all cases, use ALTMASTR in ALTER mode and re-run SETMASTR using the most recent Update and Master files (see note on renaming and deleting these files).

Notes:

- <ego> refers above to the PID of EGO as it was entered in the **UPDATE.<EXT>** file.

- In rare circumstances it may be easier to correct SETMASTR errors by repeating an update session using the previous input files. This may require deleting the newly created **MASTER.<EXT>**, **MPOINT.<EXT>** and **DPOINT.<EXT>** files, renaming **MASTER.OLD** to **MASTER.<EXT>**, and reconstituting the previous **MPOINT.<EXT>** and **DPOINT.<EXT>**.

- The number of each error appears in parentheses at the end of the error record.

Files:

Input: Any Master File and its associated Code File and Master Pointer File.

UPDATE.<EXT> produced by program ALTMASTR, containing changes or additions to Master File records.
DPOINT.<EXT> is required if new records are added to an existing Master File to which Data Files have already been linked.

Output: **MASTER.<EXT>** is the new or updated Master File produced by the program.

MASTER.OLD is a backup copy of the input Master File produced if **MASTER.<EXT>** is generated as the result of changes and/or additions to the input Master File. **MASTER.OLD** is not produced if **MASTER.<EXT>** is newly created.

UPDATE.OLD is a backup copy of **UPDATE.<EXT>**

DBFILES (the Directory File) is written when a new Master File is created. This file will be a modified version of a previously created file.

DPOINT.<EXT> is created when a Master File is created.

setmastr.log contains a record of each change that was successfully completed by the program.

setmastr.err contains a record and diagnosis of changes that were attempted, but were unsuccessfully completed because of an error or inconsistency detected by the program.

MPOINT.<EXT> (the Master Pointer File) and **DPOINT.<EXT>** (the Data Pointer File) are written when the Master File is updated. These will be modified versions of the previously created files.

SHOW

Program SHOW displays data items on separate lines on the screen. Items from the records of EGO and both parents are displayed if the file being accessed is a Master File or a Linked Data File, while a single record is displayed if the file is an Unlinked Data File.

Introductory Display:

```
PEDSYS program SHOW - Copyright 1992 SFBR

SHOW displays data items on separate lines on the screen.

  1.  Unlinked File
  2.  sort.out
  3.  EXAMPLE
  4.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>
```

1. Displaying Linked Files:

Entry of choice **3** above results in the following Secondary File List, from which up to five linked files may be selected:

```
Linked files:

  * 1.  Example Master File
  * 2.  Markers + Quant. P'types
    3.  Blood Sample Inventory
    4.  Hemoglobin Levels

More than one LINKED file may be selected for repeated sequential display.
Enter numbers corresponding to file(s) above, in the order they are to be
shown. Enter RETURN alone to continue to next step.
>1
```

Here, the Example Master File and the Markers + Quant. P'types file have been marked with an asterisk, indicating that they have been selected for display. The order of selection and display is usually not critical, since switching between files is cyclical. However, the Master File is frequently chosen first. Entry of **RETURN** alone results in a display of the first individual in the file. The display below shows the record for individual ID061:

```
ID: ID061  File - MASTER.XMP                    Record 1/32        Page  1/ 1

                                      EGO              SIRE              DAM
EGO SEQUENTIAL ID    -                  1
SIRE'S SEQ           -                  0
DAM'S SEQ            -                  0
FIRST OFFSPRING SEQ  -                 12
NEXT PATERNAL SIB SEQ-                  0
NEXT MATERNAL SIB SEQ-                  0
NEXT FULL SIB SEQ    -                  0
NUMBER OF OFFSPRING  -                  1
PATERNAL SIBSHIP SIZE-                  0
MATERNAL SIBSHIP SIZE-                  0
FULL SIBSHIP SIZE    -                  0
EGO Permanent ID     -              ID061
Birth Date (YYYYMMDD)-         1972/03/09
Sire's Permanent ID  -
Dam's Permanent ID   -
Sex (M, F or U)      -                  M
Exit Date (YYYYMMDD) -         1998/04/20
Exit Code            -                  1
Mate's Permanent ID  -              ID042
PEDIGREE NUMBER      -                  1
GENERATION NUMBER    -                  0

Enter PID, S<n> (1-32),  =<n> (1-32) or [+]/-<n> to select record; L to lock
items; * to switch files; F to find; W to write record; N for new file; H
for help; Q to quit.
>
```

Note here that the file contains no records for sire or dam.

Commands:

Records for another individual and his or her parents may be selected by entering:

PID Entry of a valid PID brings up information for EGO, FA and MO in parallel columns on the screen. *This option can be used only when a Master File or linked Data File has been selected.*

S <n> where <n> is a Sequential ID Number. For Master Files and unlinked Data Files <n> must be in the range 1 through <nrecs>, where <nrecs> is the number of records in the file. For Linked Data Files <nrecs> is the number of records in the Master File and is identical to the last Sequential ID Number in the file. In the example above, it can be seen in the status line at the top of the display that <nrecs> = 32 .

Linked Data Files need not contain records for all individuals in the Master File, which means that *the number of records will not be the same as the last Sequential ID Number.* This is because the Sequential ID Number refers to entries in the Master File, while the record counter applies to the Data File. Entry of a Sequential ID (**s<n>**) for an individual not in the Data File produces the following message:

```
No record for sequential number s<n> in <filename>
```

momentarily, after which the display reverts unchanged to the previously selected record.

= <n> where <n> is a pointer, that is, the position of a particular record in the file. <n> must be in the range 1 through <nrecs>. In the example above, the counter on the status line of the display reads **Record: 10/ 20**, so that <nrecs> = 20, and 1 ≤ <n> ≤ 20. The **=** command makes it possible to specify the display of records in a Data File independently of the Sequential ID, and thus is also appropriate for use with Unlinked Data Files. Note that used with a Master File, it produces the same result as the **S** command.

+<N>-<n> Entry of + or - followed by an integer and RETURN advances through the file containing EGO's records by steps, displaying a new set of individuals ±<n> records from the current record. For example, if record **1** is displayed on the screen, entry of **+4** would result in the display of record **5**. The default value of <n> is 1, so that entry of RETURN alone advances the display by one record . This command, like the **=** command, permits specification of records independently of the Sequential ID, making it appropriate for use with Unlinked Data Files.

L This command locks *rows* in a manner analogous to the *column* locking feature of program BROWSE, permitting reordering of items displayed, and making it possible to assure that selected items of interest always appear on the first screen page. (See explanation of row locking below.)

P If the complete record will not fit on a single page, entry of **P<n>**, where <n> is a number from 1 through the total number of pages, displays the corresponding page. Entry of **P** alone advances the screen page by one.

***** This command makes it possible to switch the SHOW display rapidly between linked files. Initial entry of an asterisk has one of two consequences, depending on the prior state of the program. If multiple files have been selected previously from the Secondary File List, the record display switches to the next file selected. If multiple files have *not* yet been chosen, a File Selection display appears, from which one or more linked Data Files may be selected. Subsequently, (after returning to the record display) entry of an asterisk switches the display to one of the selected files. The top record of each of these displays will belong to the same individual, providing that there is in fact an entry for that individual in the newly selected file. If there is no entry in the file, the message

```
No record for <PID> in <filename>. Enter RETURN for the first record of this file, or
* for next file.
```

This option applies only when a Master File or linked Data File has been selected.

F initiates the find function (see below).

W Writes EGO's entire current record (in standard PEDSYS record format) to output file **show<n>.out** (Code File **show<n>.cde**), where **<n>** is the order in which alternate files are written. For example, if the first record to be written is the Marker Phenotype entry for individual ID061 shown in the screen above, the entire record will be saved in file **show1.out**. If the next write command is issued while displaying the Master File entry for this individual (the first screen display shown above), that entire Master File record will be saved in file **show2.out**. Parental records are not written. When SHOW is used with an unlinked file, the single record displayed on the screen is written to file **show1.out**.

N This command results in a return to the File List, from which a new file (or set of files) may be selected for display in the usual manner.

H Entry of an H produces the following help display:

```
To display a record, enter a Permanent ID, "S" followed by EGO Sequential
ID, or "=" followed by a number representing a record's position in the
file, and then hit RETURN.

To move FORWARD through a file, enter RETURN for steps of one, precede
RETURN with "+"<n> for steps of <n>. To move BACKWARD through file,
precede RETURN with "-" for steps of one, "-"<n> for steps of <n>.

To change the order of items, or limit display to a subset of items, enter "L".

To switch cyclically from one LINKED file to the next, enter "*". If files
have not been preselected, initial entry of "*" presents a File List, from
which they may be chosen.

Enter "W" to write a record to output file SHOW<n>.OUT


Press RETURN to continue.
>
```

Q Exits the program.

Row locking:

By default, items in the record are displayed from top to bottom of the screen in their original order. This default may be overridden so that some items may retain a fixed position on the screen. Entry of **L** produces the following Item List Display:

```
   1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING   *15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ  *16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ  *17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE     *18 Exit Code
   5 NEXT PATERNAL SIB SE  *12 EGO Permanent ID      *19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE  *13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ     *14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  SHOW                                 File - MASTER.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to items shown above in the order they are to
appear from top to bottom of screen. Enter [C] to continue.
>
```

In the example above, Items 12 through 19 have been allocated to locked rows (using common spreadsheet terminology) starting at the top of the display. The remaining items are not displayed:

```
ID: ID063  File - MASTER.XMP                    Record 10/32        Page  1/ 1

                                    EGO              SIRE              DAM
EGO Permanent ID     -           ID063            ID038             ID042
Birth Date (YYYYMMDD)-      1980/11/28       1974/05/05        1975/00/00
Sire's Permanent ID  -           ID038
Dam's Permanent ID   -           ID042
Sex (M, F or U)      -               M                M                 F
Exit Date (YYYYMMDD) -                       1996/12/23        1996/11/14
Exit Code            -               0                1                 1
Mate's Permanent ID  -           ID065            ID042             ID038


Enter PID, S<n> (1-32),  =<n> (1-32) or [+]/-<n> to select record; L to lock
items; * to switch files; F to find; W to write record; N for new file; H
for help; Q to quit.
>
```

- Items in locked rows form a subset of other items displayed, and are positioned in order of selection from the top of the screen. They remain visible in their locked positions from one record to the next.

- Unlike program BROWSE, unlocked items are not displayed. These can be redisplayed by reissuing the lock command, and selecting items as required.

- Rows may also be locked by entering **L**, followed by an Item Number. For example, rows can be ordered in the same sequence as shown in the display above by entering **L12-19**.

- Rows may be unlocked (that is returned to their original order) by entering **L0**.

2. Displaying Unlinked Files:

The display of information from an unlinked file is different from that of a linked file in that:

- Information for only one record will be displayed. That is, parental records are not shown even if IDs for FA and MO are present on the record.

- The instruction line will not contain commands associated with linked files (**PID**, **S** or *****).

- The status line at the top of the display will contain no reference to EGO PID.

- Records from unlinked files are written only to file **show1.out**.

For this example, we wish to display file **sort.out**, which appears on the second line of the Primary File List shown at the beginning of this chapter (Reusable Output Files are unlinked by definition). As explained in Chapter III, this file may be selected in two ways: Entry of choice **1** from the Primary File List produces a Secondary File List of unlinked files containing **sort.out** which may be selected by entering its corresponding number. Alternatively, since **sort.out** already appears in the Primary File List (as the most recently produced output file), entry of choice **2** produces an immediate display of the first record in the file:

```
   File - sort.out                              Record 1/32        Page  1/ 1

EGO Permanent ID     - ID067
Birth Date (YYYYMMDD)- 1996/07/27
Sire's Permanent ID  - ID015
Dam's Permanent ID   - ID019
Sex (M, F or U)      - F
Exit Date (YYYYMMDD) -
Exit Code            - 0
Mate's Permanent ID  -
PEDIGREE NUMBER      -     1
GENERATION NUMBER    -     3


Enter =<n> (1-32) or [+]/-<n> to select record; L to lock items; F to find;
W to write record; N for new file; H for help; Q to quit.
>
```

Records may be accessed by entering any number between **1** and **<nrecs>**, where the number refers to the position of the record in the file. In the example above, the counter on the status line of the display reads **Record 1 of 32** so that <nrecs> = 32, and $1 \leq$ <n> ≤ 32. It is also possible to precede the number with **=**, as explained above, but this is not necessary.

The remaining command line functions **+/-**, **L**, **F**, **N**, **W**, **X** and **P** (when more than one page is required to see the full record) operate as described above.

The Find function:

The Find function can be activated in one of two ways. Entering **F** alone at the command line results in a Standard Item List with a request for the value sought for the particular item in EGO's record. When this has been entered, instructions for starting the search are given:

```
   1 Ego ID                 3 APOB Pvu2a Genotype     5 APOAI Level
   2 ADA  Phenotype         4 Tot. Ser. Cholestero
Enter item number, relational (EQ or NE) and value sought.  (Enclose sub-
strings in quotes)
>4 eq 140


Enter record number at which to start search (1-32), or RETURN to start at
current record (1)
>
```

In this example, the Markers + Quant. P'types file will be searched for the first record in which Total Serum Cholesterol (Item 4) has the value **140**. When a record containing the item sought is found, it is displayed in full:

```
ID: ID067  File - QUANTGEN.XMP                Record 4/32        Page  1/ 1

                                    EGO              SIRE              DAM
Ego ID                -            ID067            ID015             ID019
ADA  Phenotype        -                                                 C
APOB Pvu2a Genotype   -                                                A2A2
Tot. Ser. Cholesterol-              140              122               175
APOAI Level           -           144.0            179.0             161.3

Enter "F" to find next occurrence of Tot. Ser. Cholesterol EQ 140
Press RETURN to end search. Enter "W" to write record.
>
```

Alternatively, the Item List display can be bypassed by entering **F**, an item number, a relational, and the value sought (all separated by blanks), which leads directly to the request for a starting record number. For example, entry of **f 4 eq 140** at the command line is equivalent to the sequence described above.

Subsequent records found where the item sought has the same value by entering **F** again, etc. The message **End of search** appears when the value sought is not found in subsequent records in the file. The message will also appear if the value is not found at all in the file. A substring feature permits a search for a sequence of characters that occurs at any position in the field occupied by the item. For example the substring '1966' would search for all occurrences of the year, regardless of the month or day included in the date field.

Notes:

- Permanent IDs can be used to select individual records only from Master Files and their Linked Data Files. This is because a Master Pointer File containing indexed PIDs is required for the binary search algorithm used to locate individuals by these IDs. Records in Unlinked Data Files and files chosen with the unlinked file option in the File List Display must be selected by entering a pointer, that is, an integer designating the position of the record in the file.

- Sometimes the predefined order implicit in PEDSYS file linkage is not desirable. One way to overcome this order is to treat a linked Data File as an unlinked file, which forces PEDSYS programs to ignore relationships between records defined in the Master Pointer File, the Data Pointer File, and the internal pointers in any Multiple-entry Data File. This can be accomplished by selecting the **Unlinked** category in the initial File List Display, and typing in the file name, rather than selecting the file from the numbered list.

- The Find function relies on a simple sequential search through the records in a file, and is consequently quite slow, particularly when searching for substrings. Searches for values known to occur in multiple records (such as is the case with Multiple-entry Data Files) are often better done with program SUBSET.

- Alternates are provided for the leading command characters specifying Sequential ID Numbers. This is because there is the possibility of conflict between the default and PIDs that may begin with the same characters. For example, suppose a Master File contains records of individuals identified by PIDs, all or some of which begin with the letter **S**, followed directly by a number (S10 S11, S12, etc.). In this case, there is no way to distinguish between entry of the PID **S10** and the Sequential ID Number **10**, which must be specified in the command line with a leading **S**, that is, also as **S10**. The same problem may occur if PIDs begin with command characters **P** (or **p**), the page command).

To circumvent this possibility, the following alternates may be used:

#<n>	may be substituted for	**S**<n>	or	**s**<n>
.<n>	" " "	"	**P**<n>	or	**p**<n>

For example, specification of Sequential ID **10** would be **#10** to distinguish it from PID **S10**. Likewise, page 2 may be selected by entering **.2** to distinguish it from ID **P2**.

Characters entered at the console are first assumed to constitute a PID, so that IDs beginning with **S** or **P**, for example will always be found and displayed. The entry is interpreted as a control sequence only if no corresponding PID is found in the file.

The alternates are available in all relevant PEDSYS programs.

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **show subset.out**.

Files:

Input: Any Master File or Data File, and an associated Code File. Recursive use of output files is permitted, that is, file **show<n>.out** may be used as input to program SHOW.

Output: Console screen, and with the **W** "write to file" command **show<n>.out**, where **<n>** is the order in which alternate files are written.

Program limits:

Only five linked files may be selected for sequential display.

The Find function relationals are limited to **EQ** (equal to) and **NE** (not equal). Compound searches (for occurrences of combined item values) are not possible.

SHOWDATA

Program SHOWDATA gives a direct screen display of all records for a single individual from a PEDSYS Multiple-entry Data File. Single records for this individual in the Master File and linked Single-entry Data Files may be displayed as well. SHOWDATA is in some ways similar to BROWSE, but unlike BROWSE, it is *individual-oriented*, rather than *record-oriented*. That is, SHOWDATA displays multiple records of a single individual, whereas BROWSE displays multiple records for more than one individual in a file.

Introductory Display:

```
PEDSYS program SHOWDATA - Copyright 1992, 1999 SFBR

SHOWDATA gives a direct screen display of all records for a single
individual.

  1.  Unlinked File
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>
```

Access to records in Linked Data Files may follow either (1) the order of corresponding records in the Master File, or (2) the order of records in the Data File itself. Records in unlinked files on the other hand are sorted and temporarily indexed by SHOWDATA, making order of access a function only of their file position. These considerations lead to slightly different effects of some of the program's commands, dependent upon the file type, which will be noted in the command descriptions described below. SHOWDATA treats single-entry files in much the same way as does program BROWSE, except that unlike BROWSE, only one record is displayed at a time.

1. Displaying Linked Data Files

In the example below, the Blood Sample Inventory file and the Hemoglobin Levels file have been marked with an asterisk, indicating that they have been selected for display. The order of selection and display is usually not critical, since switching between files is cyclical.

```
Linked files:

    1.  Example Master File
    2.  Markers + Quant. P'types
*   3.  Blood Sample Inventory
*   4.  Hemoglobin Levels

More than one LINKED file may be selected for repeated sequential display.
Enter numbers corresponding to file(s) above, in the order they are to be
shown. Enter RETURN alone to continue to next step.
>
```

Entry of **RETURN** results in a display of records belonging to the first individual in the Sample Inventory File. At the top of the screen, a status line appears showing the file name (**SAMPLES.XMP**), the PID of the first individual in the file (**ID097**), the number of items per record (**9**), the number of

records belonging to the individual selected (**3**), the order of access (**LIFO**), the number of pages required to display these records (**1/1**), and the position of the record currently selected in the display (**1/96**). The display window extends to the right for as many complete items as will fit on the screen. Columns are unlocked.

```
SAMPLES.XMP  ID:ID097  Items:9  Recs:3 (LIFO)    Page:1/1    FilePos:1/96

 ( 1)  ( 2)     ( 3)    ( 4)  ( 5)   ( 6)  ( 7)   ( 8)   ( 9)
 ID   SAMPLE    DATE    VIAL SAMPLE ANIMAL PROJ  COMMNT  PNTR
                TAKEN        STATUS STATUS

ID097  W219  1988/12/22 TYG               HLM             2
ID097  S219  1980/07/01 9ML   LOW         HL   HOSP       1
ID097  R219  1980/07/01 2ML   CRIT        HL   HOSP       0

Enter [+]<n> or -<n> for new page; PID, S<n> (1-32) or =<n> (1-96) to set
top row; ">" or "<"<n> to shift window; I<n> (1-9) to set window; L to
lock columns; O to change order; * to switch files; N for new file; H for
help; Q to quit.
>
```

Commands:

PID Entry of a valid PID displays at least one corresponding record at the top of the display window, providing that there is a record for the individual in the file. If the file is a Single-entry file, only one record is shown; if it is a Multiple-entry file, multiple records of the individual chosen will be seen. The screen displayed below results from entering PID **ID042** in the Sample Inventory File.

S <n> where <n> is a Sequential ID Number of an individual in the Master File. Entry of a valid Sequential ID displays at least one corresponding record at the top of the display window, providing that there is a record for the individual in the file. If the file is a Single-entry file, only one record is shown; if it is a Multiple-entry file, multiple records of the individual chosen will be seen. The screen displayed below also results from entering **s4**, the SEQ of individual ID042 in the Example Master File:

```
SAMPLES.XMP  ID:ID042  Items:9  Recs:3 (LIFO)    Page:1/1    FilePos:56/96

 ( 1)  ( 2)     ( 3)    ( 4)  ( 5)   ( 6)  ( 7)   ( 8)   ( 9)
 ID   SAMPLE    DATE    VIAL SAMPLE ANIMAL PROJ  COMMNT  PNTR
                TAKEN        STATUS STATUS

ID042  W218  1988/12/22 TYG           D    HLM             6
ID042  S218  1980/10/03 9ML   LOW     D    HL              4
ID042  R218  1980/10/03 2ML   LOW     D    HL              0

Enter [+]<n> or -<n> for new page; PID, S<n> (1-32) or =<n> (1-96) to set
top row; ">" or "<"<n> to shift window; I<n> (1-9) to set window; L to
lock columns; O to change order; * to switch files; N for new file; H for
help; Q to quit.
>
```

= <n> where <n> is a number corresponding to the position of a specific record in the file (shown as **FilePos** in the right-most position of the status line). <n> must be in the range 1 through <nrecs>, where <nrecs> is the number of records in the file currently selected for display. This command results in a display of all records belonging to the PID found on the record specified. Note that the same display can result from entry of more than one number. For example, entry of **=4**, **=6** or **=56** produce the screen displayed above, since there are records for **ID042** in positions 4, 6 and 56 in this file.

+<n>, **-**<n> Entry of **+** or **-** followed by an integer displays records belonging to the individual ±<n> records from the last one selected for display (= **FilePos**). The default value of <n> is 1, so that entry of **+** alone advances the display by one record. Horizontal position remains unchanged.

Important note: *Entry of the* **+/-** *command while displaying records in a Linked Multiple-entry file will result in the display of records for an apparently unpredictable individual.* This is because the record counter is set at the record of the individual selected for display (= **FilePos**), and the **+/-** command operates by incrementing the record counter (= **FilePos** +<n>). Since ordering of records in Linked Multiple-entry Data Files may be unpredictable, so too may be the next individual shown.

RETURN Entry of RETURN alone advances the display to the record immediately following the last record currently displayed. Again, owing to the unpredictable order of records in Linked Multiple-entry Data Files, repeated entry of RETURN results in an unpredictable sequence of individuals shown.

***** This command makes it possible to switch the SHOWDATA record display rapidly between linked files selected when the program is started. Entry of an asterisk switches the display cyclically to the next of the selected files (here file **HEMOGLOB.XMP**):

```
HEMOGLOB.XMP  ID:ID042  Items:4  Recs:13 (LIFO)   Page:1/1    FilePos:31/238

 ( 1)    ( 2)     ( 3)  ( 4)
 EGO     DATE     hgb   PNTR

ID042 1982/04/29  15.6    30
ID042 1983/05/06  14.8    29
ID042 1984/01/05  14.4    28
ID042 1985/01/03  16.7    27
ID042 1986/01/21  15.0    26
ID042 1987/01/14  14.3    25
ID042 1988/03/08  13.0    24
ID042 1989/01/25  15.0    23
ID042 1990/01/03  41.4    22
ID042 1991/05/15  14.2    21
ID042 1992/05/01  14.0    20
ID042 1993/04/29  13.2    19
ID042 1994/04/08  13.0     0

Enter [+]<n> or -<n> for new page; PID, S<n> (1-32) or =<n> (1-96) to set top
row; ">" or "<"<n> to shift window; I<n> (1-9) to set window; L to lock
columns; O to change order; * to switch files; N for new file; H for help;
Q to quit.
>
```

Note that the individual (**ID042**) displayed in the previously selected file (**SAMPLES.XMP**) is the one shown when the switch to the new file occurs. If there is no entry for an individual in the new file, however, the following message will appear:

```
No record for <PID> in <filename>.  Enter RETURN for the first record of this
file, or * for next file.
```

At this point, entry of an asterisk skips to the next file selected for display, showing (a) a single record if the next file is a Master File or Single-entry Data File, or (b) multiple records if the new file is a Multiple-entry Data File.

><i> or **<**<i> where <i> is a number of items to be shifted in the range shown on the display. Entry of this sequence followed by **RETURN** shifts the unlocked portion of the window horizontally by ±<i> items, without changing page position. In the example above 1 ≤ <i> ≤4, since there are 4 items per record in the Hemoglobin Levels file.

I<i> where <i> is an integer in the range 1 through <nitems>, the number of items on the record. Entry of this sequence sets the horizontal position of the item selected at the left-most unlocked column. When **RETURN** is hit, the same page is displayed with the window shifted horizontally as specified. In the example above 1 ≤ <i> ≤ 4, since there are 4 items per record in the Hemoglobin Levels file.

L Entry of this command makes it possible to lock columns, that is, fix their position so that they remain visible despite horizontal shifts of the display window (see explanation below).

P The page command is used when more than one display page is required for an individual's multiple records. Entry of **p** displays the next page, **p**<n> the <n>th page.

O The default order in which records of a Multiple-entry Data File are displayed is LIFO (last-in-first-out), with the most recently entered records appearing at the top of the screen. The **O** command switches back and forth between LIFO and FIFO (first-in-first-out), with the most recently entered records displayed at the bottom of the list.

N This command results in a return to the File List, from which a new file (or set of files) may be selected for display in the usual manner.

H Entry of an H produces the following help display pages:

```
To display records belonging to a specific individual, enter EGO's PID or "S"
followed by EGO Sequential ID as it appears in the Master File.

To display records without reference to IDs, enter "=" followed by a number
representing position of a record in the file.

Entry of "+"<n> or "-"<n> moves through the file, displaying records belong-
ing to a series of individuals whose order depends on the sequence of records
in the Data File.  NOTE: Use of this command with Linked Files may result in
an unpredictable sequence of individuals displayed.

To shift the window RIGHT or LEFT, enter ">" or "<" for shifts of one item,
">"<i> or "<"<i>, respectively, for shifts of <i> items.

To set the item that will appear in the left-most unlocked column, enter
"I"<i> where <i> is the item number, then hit return.

Enter [C] to return to data display. Enter "P" to page.
>
```

and

```
Enter "L" to select columns for LOCKED display, that is, fixed in position
on the left-hand side of the screen despite horizontal movement of the dis-
play window.

Enter "O" to switch between LIFO and FIFO order of multiple records.

Enter "P" to see the next, or "P<n>" to see the <n>th page of multiple
records

Enter "*" to switch cyclically from one file to the next. If files have not
been preselected, initial entry of "*" presents a File List, from which
they may be chosen.

Enter "N" to select new files for display.

Enter "Q" to quit program.

Press RETURN to continue.
>
```

Q exits the program.

2. Displaying Unlinked Data Files

When an unlinked file is selected for display, the following display appears, from which an identifier common to multiple records (usually a PID) is selected:

```
   1 Animal ID              4 Vial Type              7 Project Code
   2 Sample Number          5 Sample Status          8 Comments
   3 Date Taken             6 Animal Status          9 INTERNAL POINTER
--------------------------------------------------------------------------------
  SHOWDATA    Select ID and sort items        File - reformat.out
--------------------------------------------------------------------------------
Enter item number corresponding to common identifier of multiple records.
Enter additional item numbers as needed to order records within sets. By
default, records are sorted in ascending order; precede item numbers with a
"D" to sort in descending order. Enter [C] to continue.
>1
```

Once an identifier has been selected, a screen display identical in configuration to those shown for Linked Files appears.

Commands:

Control of unlinked file display is for the most part the same as for Linked Files, with the following differences:

S <n> In the case of unlinked files, <n> is interpreted as the *nth* individual in the Data File, rather than as a Sequential ID Number of an individual in the Master File.

+<n>, **-**<n> The **+**/**-** command behaves exactly the same for both Linked and unlinked files; that is, entry of **+** or **-** followed by an integer displays records belonging to the individual ±<n> records from the last one currently selected for display (= **FilePos**). The default value of <n> is 1, so that entry of **+** alone advances the display by one record. In case of unlinked files, however, this command has a more predictable result, because records are ordered sequentially by ID.

RETURN As in the case of Linked Data Files, entry of RETURN alone advances the display to the record immediately following the last record currently displayed. Since unlinked files are ordered, this command makes it possible to step through the file sequentially, individual by individual.

Column locking:

By default, all items in the record may appear in the display window in their original order, depending on the position of the window with respect to the record. This default may be overridden so that some items may retain a fixed position in the window. Entry of **L** produces the following Item List Display:

```
 * 1 Animal ID              4 Vial Type            * 7 Project Code
 * 2 Sample Number          5 Sample Status          8 Comments
   3 Date Taken             6 Animal Status          9 INTERNAL POINTER
--------------------------------------------------------------------------------
  SHOWDATA                              File - SAMPLES.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to items shown above in the order they are to be
assigned to locked left-hand columns,  or enter [C] to continue.
>
```

In the example above, EGO Permanent ID, Sample ID Number and Project (Items 1, 2 and 7) have been allocated to locked columns (using common spreadsheet terminology) at the left-most side of the display. The remaining items remain unlocked:

```
SAMPLES.XMP  ID:ID097  Items:9  Recs:3 (LIFO)    Page:1/1    FilePos:1/96

 ( 1)  ( 2)  ( 7)    ( 3)    ( 4)  ( 5)   ( 6)    ( 8)   ( 9)
 ID   SAMPLE PROJ    DATE    VIAL SAMPLE ANIMAL  COMMNT  PNTR
                     TAKEN        STATUS STATUS

ID097  W219  HLM  1988/12/22 TYG                            2
ID097  S219  HL   1980/07/01 9ML   LOW          HOSP        1
ID097  R219  HL   1980/07/01 2ML   CRIT         HOSP        0

Enter [+]<n> or -<n> for new page; PID, S<n> (1-32) or =<n> (1-96) to set top
row; ">" or "<"<n> to shift window; I<n> (1-9) to set window; L to lock
columns; O to change order; * to switch files; N for new file; H for help;
Q to quit.
>
```

- Items in **locked** columns form a subset of other items displayed, and are positioned in order of selection from the left-most boundary of the screen. They remain visible in their locked positions despite subsequent horizontal shifts of the remaining items selected for display. Locking is particularly useful for maintaining visibility of a PID, while shifting horizontally across long records.

- Any item not explicitly designated as locked is assigned by default to the **unlocked** display. Items in the unlocked portion are positioned in order from the right-most locked column. Visibility of items in the unlocked portion depends on their number, the width of the unlocked segment of the window, and the position of the window as specified by the relevant command sequence.

- The total formatted width of locked items may not exceed the width of the screen.

Notes:

- Multiple records are shown (a) when a Permanent or Sequential ID Number is entered while a Multiple-entry file is being displayed, or (b) when the ***** command switches the display to a Multiple-entry file.

- Single records are shown when a Permanent ID Number, a Sequential ID Number or the **=** or **+/-** commands are entered while a Single-entry file is being displayed, or if an individual has only one records in a Multiple-entry file.

- Currently, there is no FIND function in SHOWDATA. Output may not be directed to a file.

- Alternates are provided for the leading command characters specifying Sequential ID and Item numbers. This is because of the possibility of conflict between the defaults and PIDs that may begin with the same characters. For example, suppose a Data File contains records of individuals identified by PIDs, all or some of which begin with the letter **S**, followed directly by a number (S10, S11, S12, etc.). In this case, there is no way to distinguish between entry of the PID **S10** and the Sequential ID Number **10**, which must be specified in the command line with a leading **S**, that is, also as **S10**. The same problem may occur if PIDs begin with command characters **I**, or **i**, and **P** or **p**.

  To circumvent this possibility, the following alternates may be used:

#<n>	may be substituted for	**S**<n>	or	**s**<n>
@<n>	"	"	"	"	**I**<n>	or	**i**<n>
.<n>	"	"	"	"	**P**<n>	or	**p**<n>

  In the example just given, specification of Sequential ID **10** would be **#10** to distinguish it from PID **S10**. Likewise, **@5** may be substituted for **i5** to specify that Item 5 be placed in the left-most unlocked column, and **.2** may be used in place of **p2** to specify the second display page, where needed.

  The default characters were chosen to speed up command sequence entry by avoiding the use of the keyboard shift key. The alternates are available in all relevant PEDSYS programs.

- To save screen space for data, a maximum of two mnemonics per item are displayed in the header.

Files:

Input: Up to five Linked Data Files, or a single unlinked file with their associated Code Files.

Output: Console screen.

SORT

Program SORT performs a compound sort of records in any file for which an appropriate Code File has been constructed.

Introductory Display:

```
PEDSYS program SORT - Copyright 1992, 1999 SFBR

SORT performs a compound sort of records in any file for which an
appropriate Code File has been constructed.

  1.  Unlinked File
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>
```

In this program, the usual asterisk that marks items chosen for inclusion is replaced by directional markers: **+** marks items to be sorted in ascending order, **-** marks items sorted in descending order:

```
   1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ  +16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE   12 EGO Permanent ID       19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE   13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ      14 Sire's Permanent ID   -21 GENERATION NUMBER
--------------------------------------------------------------------------------
  SORT        Items to be sorted              File - MASTER.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to items to be sorted in order of decreasing
significance. Precede item number with a "D" to sort in descending order.
Enter [C] to continue.
>
```

Items to be sorted in ascending order may be selected by entering item numbers alone, or by preceding them with **A** (for example, **a10**); items to be sorted in descending order are preceded by a **D** (example, **d13**). Here, selections have been made to sort the Example Master File in ascending order by Sex, which results in Item 16 being marked by a **+**, and descending order by Generation Number, which is indicated by the **-** mark adjacent to Item 21.

Confirmation Display:

```
 + 1 Sex (M, F or U)       - 2 GENERATION NUMBER
--------------------------------------------------------------------------------
  SORT        Items to be sorted              File - MASTER.XMP
--------------------------------------------------------------------------------

Is this correct?  [Y]/N
>
```

The order of the sort is shown here. Standard Editing procedures follow if **N** is entered.

Notes:

- Program SORT uses the Opt-tech Sort routine which is the copyrighted property of Opt-Tech Data Processing, P.O. Box 678, Zephyr Cove, NV 89448, and is distributed under license from the vendor.

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **sort subset.out**.

Files:

Input: Any Master File or Data File, and an associated Code File. Recursive use of output files is permitted, that is, file **sort.out** may be used as input to program SORT.

Output: **sort.out** contains all records in the input file in sorted order. **sort.cde** defines the record structure of **sort.out**.

Program limits:

A maximum of 20 items may be included in a compound sort.

SUBSET

SUBSET generates compound subsets of records in any file for which an appropriate Code File has been constructed.

Introductory Display:

```
PEDSYS program SUBSET - Copyright 1992, 1999 SFBR

SUBSET generates compound subsets of records in any file for which an
appropriate Code File has been constructed.

  1.  Unlinked File
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>
```

When a file has been selected, the following Item List Display appears:

```
   1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ  *16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE   12 EGO Permanent ID       19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE   13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  SUBSET                                File - MASTER.XMP
--------------------------------------------------------------------------------
Enter operator (IN or EX), item number, relational (EQ,NE,GT.GE,LT or LE) and
value, each separated by blanks. Precede operator with "*" to compare items;
precede 2nd line with AND or OR for compound comparison. Enclose substrings
in quotes. Enter "H" for help, [C] to begin subset.
 in 16 eq M
>and 20 gt 0
```

The display above shows commands entered to generate a subset of the Example Master File that would include all males (**in 16 eq M**) that are in generation 1 or higher (**and 20 gt 0**).

Notes on query logic:

- To compare a listed item with a numerical or alphabetic value, enter commands separated by a single blank in the following order:

  Operator: **IN** or **EX** (include or exclude)
  Item number: Left-justified

Relational: Use FORTRAN conventions: EQ, LE, LT, GE, GT, or NE
Value or string : Left-justified; enclose substrings in matched single or double quotes

Examples: **in 13 lt 19780630** includes all individuals born before 30 June 1978 (but will also include individuals whose birth dates are blank).

ex 1 gt 10 includes only the first 10 individuals in the input file

IN 18 NE 'X' excludes individuals with Exit Code **X**.

- To compare one item with another, begin the command line with an asterisk.

  Example: ***EX 13 EQ 17** excludes stillborn individuals (also excludes individuals for which both birth date and death date are blank).

- To make a compound comparison, begin second command line with **AND** or **OR**

  Example: see command line in the Item List Display above.

- Substring comparison is limited to character data or dates (that is, items of Data Types **C** and **D** as defined in the Code File); comparison is case-sensitive. Only relationals **EQ** and **NE** are permitted in substring comparison.

Confirmation Display:

```
 1.  INCLUDE Sex (M, F or U)       EQ  <M>
       AND  PEDIGREE NUMBER        GT  <    0>

--------------------------------------------------------------------------------
             SUBSET will be constructed as shown above.
--------------------------------------------------------------------------------

Is this correct?  [Y]/N
>
```

Standard Editing procedures are used, with the addition of the following queries:

- If the subset is being defined by any item whose data type is **I**, **R** or **D**, the following request will appear:

```
Interpret blank fields as zero?  Y/[N]
>
```

Responding with **Y** directs SUBSET to treat numerical fields filled with blanks as a single 0 (or 0.), following FORTRAN integer and floating point format conventions. This interpretation applies to all-blank entries made at the keyboard for comparison with numerical items, as well as blank numerical items read from records in the input file. Once chosen, this option applies to all

numerical items -- it is not possible, for example, to interpret blanks as zeros in one numerical field and not in another.

Responding with **N** directs SUBSET to honor blanks as legitimate characters in their own right (ASCII Code 32) that are quite different from the zero character (ASCII Code 48). This setting makes it possible, for example, to distinguish and eliminate unwanted blank entries in numerical items.

- A final request always appears before SUBSET begins its execution:

```
INCLUDED records will be saved in subset.out.  Save EXCLUDED records
in nosubset.out?  Y/[N]
>
```

Entering **Y** directs SUBSET to save separately records that do not meet the criteria for inclusion in the subset. Entering N directs SUBSET to skip excluded records.

Notes:

- The substring feature permits a sequence of characters occurring at any position in the field occupied by the item to be compared with the sequence entered at the keyboard. For example, the command **in 13 eq '1966'** would create a subset that included all individuals born in that year, regardless of the month or day included in the date.

- The substring feature also permits the use of compound coding. To illustrate this technique, at the Southwest Foundation many animals are used as subjects in more than one research project. Rather than add a separate item for each project code to records in the Master File, we have combined codes into a single item. In our scheme, **H** signifies that the animal is used in a heart disease project and **P** indicates that the individual is included in a panel of animals used to characterize various genetic marker systems. Several code combinations are possible: **H_**, **P_**, **HP**, **PH**, and **__**. Constructing a subset by including records having the substring "H" anywhere in this item will assemble a file containing all animals in the heart disease project, regardless of their panel membership; *excluding* records containing the substring "_" would result in a file containing all animals used in both projects, etc.

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **subset sort.out**.

Files:

Input: Any Master File or Data File, and an associated Code File. Recursive use of output files is permitted, that is, file **subset.out** or **nosubset.out** may be used as input to program SUBSET.

Output: **subset.out** contains records that meet criteria for inclusion in the subset, in the same order as in the input file.

nosubset.out contains a complementary subset that do not meet criteria for inclusion in **subset.out**, in the same order as in the input file.

subset.cde defines the record structure of **subset.out**.

nosubset.cde defines the record structure of **nosubset.out**.

Program limits:

IN and **EX** may be repeated up to 25 times each in a single run.

TALLY

Program TALLY generates cross-tabulations, and calculates distributions of items for any file for which an appropriate Code File has been constructed.

Introductory Display:

```
PEDSYS program TALLY - Copyright 1992, 1999 SFBR

TALLY generates cross-tabulations, and calculates distribution parameters for
items in any file for which an appropriate Code File has been constructed.

  1.  Unlinked File
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>
```

When a file has been selected, the following display appears:

```
The following operations can be performed on individual items, either singly
or in a two-way tabulation (for example, NUMBER OF OFFSPRING by SEX):

     1. Frequency distribution
     2. Distribution parameters (mean, variance, skewness, kurtosis)
     3. Both of the above

Enter the number of the task to be performed.
>
```

1. Frequency Distribution

If the **Frequency Distribution** option has been selected (choice **1**), the following Item List Display appears:

```
   1 EGO SEQUENTIAL ID     * 8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ  *16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE   12 EGO Permanent ID       19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE   13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  TALLY       Frequency Distribution   File - MASTER.XMP
--------------------------------------------------------------------------------
Enter number corresponding to each item shown above for which a Frequency
Distribution is to be calculated. For a two-way classification, follow this
number with an asterisk and the number corresponding to the controlling
variable. Example:  8*16 produces an NKID-by-SEX (column-by-row) distribution
for a Master file. Enter [C] to continue to next step.
>
```

Here, a two-way classification of **NUMBER OF OFFSPRING** (the *tabulated* variable) by **SEX** (the *controlling* variable) has been selected. Since the tabulated variable is numeric, the following sequence appears after the Standard Confirmation and Editing step:

```
Set bounds on the value of NUMBER OF OFFSPRING for which a Frequency
Distribution is calculated?  Y/[N]
>y

Enter INCLUSIVE lower bound for value of NUMBER OF OFFSPRING. Values below
this bound will be tallied as a single class. Enter RETURN alone to let
smallest data value determine lower bound.
>1

Enter EXCLUSIVE upper bound for value of NUMBER OF OFFSPRING. Values equal
to or greater than this bound will be tallied as a single class. Enter
RETURN alone to let highest data value determine upper bound.
>5

Enter number of classes (or class width preceded by @) for NUMBER OF
OFFSPRING.
>4
```

If bounds on the tabulated value are not required (response **N** to the first query above), the program will begin processing, tabulating the frequency of each distinct value of the item as a separate class. In this example, however, choice **Y** was selected to set bounds on the tabulated variable (NUMBER OF OFFSPRING), which leads next to the options of setting its lower and upper bounds (**1** and **5** respectively). Note that the lower bound is **inclusive** and the upper bound **exclusive**, which means that in this example, the range of offspring sibships tabulated is 1 through 4. Finally, the choice is given to set the number of classes (here, **4**) into which it is to be grouped. An alternate means of grouping the tabulated variable is to specify class width. In this example, responding to the last query with **@1** would yield the same result.

This sequence of options is repeated for each distribution to be tabulated.

Output Example: Frequency Distribution

Below is an display of file **tally.tab** showing the frequency distribution specified for NUMBER OF OFFSPRING by SEX in the Example Master File:

```
/export/home/population/bdyke/example/MASTER.XMP

DISTRIBUTION OF NUMBER OF OFFSPRING BY Sex (M, F or U)

NUMBER OF OFFSPRING LIMITS: LOWER =  1.000000 (INCLUSIVE), UPPER =  5.000000 (EXCLUSIVE),
CLASS INTERVAL =  1.0000

                | NUMBER OF OFFSPRING
Sex (M, F or U) |       <      1.00000  2.00000  3.00000  4.00000     >=     TOTAL
----------------------------------------------------------------------------------------
F               |       5         8        3        0        1        0       17
M               |       2         9        3        1        0        0       15
----------------------------------------------------------------------------------------
TOTALS          |       7        17        6        1        1        0       32
```

2. Distribution Parameters

If the Distribution Parameters option has been selected (choice **2** in the Introductory Display), the following display appears:

```
   1 EGO SEQUENTIAL ID     * 8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
   2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ  *16 Sex (M, F or U)
   3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE   12 EGO Permanent ID       19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE   13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  TALLY       Distribution Parameters        File - MASTER.XMP
--------------------------------------------------------------------------------
Enter number corresponding to each item shown above for which Distribution
Parameters are to be calculated. For a two-way classification, follow this
number with an asterisk and the number corresponding to the controlling
variable. Example:  8*16 produces NKID-by-SEX (column-by-row) statistics
for a Master file. Enter [C] to continue to next step.
>
```

Here again, a two-way classification of NUMBER OF OFFSPRING by Sex has been selected. Since the tabulated variable is numeric, the following sequence appears after the Standard Confirmation and Editing step:

```
Set bounds on the value of NUMBER OF OFFSPRING for which Distribution
Parameters are calculated?  Y/[N]
>y

Enter INCLUSIVE lower bound for value of NUMBER OF OFFSPRING.  Values below
this bound will not be tallied.  Enter RETURN alone to let smallest data
value determine lower bound.
>1

Enter EXCLUSIVE upper bound for value of NUMBER OF OFFSPRING.  Values equal
to or greater than this bound will not be tallied. Enter RETURN alone to
let highest data value determine upper bound.
>5
```

If **N** is entered in response to the first query, the program will begin computing parameters for all values of the item selected. In this example as well, choice **Y** was made initially to set bounds on the tabulated variable, which leads next to the options of setting its lower and upper bounds (here, **1** and **5** respectively).

Note: the choice of grouping the tabulated variable into a reduced number of classes is not offered for the Distribution Parameters option.

Output Example: Distribution Parameters

Below is an display of file **tally.tab** showing Distribution Parameters specified for NUMBER OF OFFSPRING by SEX in the Example Master File:

```
/export/home/population/bdyke/example/MASTER.XMP

DISTRIBUTION PARAMETERS FOR NUMBER OF OFFSPRING BY Sex (M, F or U)

Sex (M, F or U)|   MEAN    VAR.    STDV.   LOW     HIGH   SKEWNESS  KURTOSIS  SUM       N
---------------------------------------------------------------------------------------------
F              |   1.0588  1.0588  1.0290  0.0000  4.0000  1.1882   1.3796    18.0000   17
M              |   1.2000  0.6000  0.7746  0.0000  3.0000  0.5508  -0.0578    18.0000   15
---------------------------------------------------------------------------------------------
```

Output control:

Tabular output is always sent to file **tally.tab**, but may also be sent directly to one or more printers (depending on the hardware available), or the console screen:

```
Output from TALLY is written to file tally.tab, but also can be sent directly
to the following output devices:

      1. Main Lab LaserWriterII
      2. Main Lab Line Printer
      3. Printer Room Sun LaserWriter
     [4] Console Screen

Enter number of device, or [C] to continue without printing.
>
```

In addition to output sent to file **tally.tab**, results of computations can be output in PEDSYS format and sent to files **tally<n>.out** and **tally<n>.cde**, where **<n>** is the order that variables are selected for computation. This is especially valuable when tabulating values across individuals identified by EGO PID, which makes it possible to use the file as a PEDSYS unlinked Data File. PEDSYS-formatted output is controlled by the following sequence:

```
In addition to output sent to file tally.tab, results can be output in
PEDSYS format and sent to files tally<n>.out and tally<n>.cde, where <n> is
the order that variables are selected for computation.

Write results to output files in PEDSYS format?  Y/[N]
>
```

Output Example: PEDSYS format

Below is an display of file **tally1.out** showing the frequency distribution specified for NUMBER OF OFFSPRING by SEX in the Example Master File (output of program BROWSE):

```
tally1.out                              Items: 8       Top record: 1/2
      (1)        (2)   (3)    (4)    (5)    (6)    (7)   (8)
      SEX        NKID NKID   NKID   NKID   NKID    NKID NKID
                 <    1.0000 2.0000 3.0000 4.0000  >=   TOTAL

 F                  5     8      3      0      1      0    17
 M                  2     9      3      1      0      0    15

Enter [+]<n> or -<n> for new page; <n> (1-2) to set top row; ">" or "<"<n>
to shift window; I<n> (1-8) to set window; L to lock columns; N for new
file; F to find; H for help; Q to quit.
>
```

Notes:

- Integer data are converted to real (floating point) numbers if bounds are set. This change will be preserved in column and row headings in **tally.tab**. Conversion of integer to real is not done if bounds are not required.

- Output is formatted as F12.4, giving a range of 0.0001 through 9999999.9999 in absolute value. Internal computations are done in double precision floating point arithmetic (see Appendix A, page A-3 for specifications).

- Blank fields are treated as missing data, rather than as zeros in calculations of Distribution Parameters.

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **tally subset.out**.

Files:

Input: Any Master File or Data File, and an associated Code File. Recursive use of output files is permitted, that is, file **tally<n>.out** may be used as input to program TALLY.

Output: **tally.tab** contains tabular output of the program. No Code File is generated for this output file.

tally<n>.out contains output of the program in PEDSYS Data File format. **tally<n>.cde** defines the structure of **tally<n>.out**.

Program limits:

A Frequency Distribution may be tabulated for data of any type (C, I, R, or D); Distribution Parameters can be computed only for numerical data (types I, R, or D); bounds on the tabulated variable may be specified only for numerical data (types I, R, or D). Number of classes may be set only for frequency distributions.

The number of items for which computations may be done at one time is limited to twelve for Macintosh OS and Dos/Windows, 50 for Unix systems. Bounded distributions may be subdivided into a

maximum of 1,000 equal interval classes. A maximum of 8,500 unique data values may be tabulated using Macintosh OS and Dos/Windows, 20,500 for Unix systems. Field width of individual items is limited to 20 for Macintosh OS and Dos/Windows, 99 for Unix systems.

TRANSLAT

Program TRANSLAT is used to translate between PEDSYS and other record formats, including Pedigree/Draw and character-delimited field format.

Introductory Display:

```
PEDSYS program TRANSLAT - Copyright 1992, 1999 SFBR

 TRANSLAT converts between PEDSYS and other record formats.

 CHARACTER-DELIMITED FIELDS
   1. PEDSYS to delimited field
   2. Delimited field to PEDSYS

 PEDIGREE/DRAW 4.4 (Use PREPDRAW for Version 5.0)
   3. PEDSYS to Pedigree/Draw 4.4
   4. Pedigree/Draw 4.4 to PEDSYS

 PEDSYS format to:
   5. FISHER              8. CRI-MAP             11. SOLAR (MIXED)
   6. MENDEL              9. PAP
   7. S.A.G.E. (REGC)    10. LINKAGE

Enter number corresponding to type of conversion required, or "Q" to quit.
>
```

RECIPROCAL CONVERSIONS

1. PEDSYS to delimited-field

When PEDSYS to delimited-field format translation is chosen, Standard File List Displays appear, from which the file to be translated is chosen. Once a file is selected and items have been chosen, the following display appears, from which a translation format is chosen:

```
Convert MASTER.XMP from PEDSYS to character-delimited field format.

[1] TAB character
 2. Comma
 3. Single blank character
 4. Other delimiter (may be 1 to 3 characters)

Enter number corresponding to type of separator to use as item delimiter
>1

Enter single or double quote (' or ") to enclose character data. Enter
RETURN otherwise.

Use CODE.XMP Item Labels to create column headings in the first record in
translat.tab?  Y/[N]
>
```

The last option makes it possible to create a header record for output file **translat.tab**, a common format requirement for software that can read character-delimited data.

2. Delimited-field to PEDSYS

When delimited-field to PEDSYS translation is chosen, the following display appears:

```
Enter name of file to be converted to PEDSYS format
>examped.tab


Enter RETURN to create a Code File based on input data field widths, or
enter the name of an existing Code File to define the format of output
records.
>CODE.XMP
```

Here, file **examped.tab** containing character-delimited records has been selected for conversion to PEDSYS format, and rather than simply allowing input record field widths to establish PEDSYS item formats, a pre-existing Code File **CODE.XMP** has been entered. Item Descriptors in this Code File will be used to define the format of output records in **translat.out**.

Note: Use of a pre-existing Code File in this manner will not work if any input field width is greater than the item field width defined in the Code File, if Data Types conflict, or if the number of input fields does not correspond with the number of items defined in the Code File.

Next, the delimiter of fields in the input file is selected:

```
Convert examped.tab to PEDSYS format.

[1] TAB character
 2. Comma
 3. Single blank character
 4. Other delimiter (may be 1 to 3 characters)

Enter number corresponding to type of separator used as item delimiter
>
```

If choice **4** (other delimiter) has been selected, the following display appears:

```
Enter item delimiter enclosed in single or double quotes.
>
```

After any of the four choices of delimiter type, the next display appears:

```
Enter type of quote (' or ") if character data are enclosed in quotes.
Enter RETURN otherwise.
>
```

If an existing Code File is not specified, the following query appears:

```
Does the first record in examped.tab contain column headings? Y/[N]
```

If the first record in the file contains column headings separated by the same delimiters as the data fields in subsequent records, TRANSLAT will use these headings in constructing the PEDSYS output Code File.

Notes on character-delimited field conversion.

- Normally, quotes surrounding character data are needed only when commas or blanks are used as a delimiters, since most software cannot distinguish between a blank or comma included as *data*, and either character used as a *delimiter*.
- In PEDSYS-to-foreign translation, foreign field width for any item is determined as the length of the longest string found in the corresponding field of the PEDSYS record, **excluding** leading and trailing blanks.
- In foreign-to-PEDSYS translation, TRANSLAT computes PEDSYS output field width for any item as the length of the longest string found in the corresponding field of the foreign input record, **including** leading and trailing blanks, except that input fields of null width (designated by a pair of adjacent delimiter characters, or a delimiter character followed directly by the end of record) are assigned a single blank character item in PEDSYS format.
- In foreign-to-PEDSYS translation, computed width can be *increased* if specified by an appropriate pre-existing Code File accompanying the input file, but it cannot be *decreased* this way.
- Output file **translat.cde** will contain the same Item Labels as the pre-existing Code File used to define PEDSYS output format if no field width, Data Type or item number conflicts are found. If conflicts are found, **translat.cde** defaults to generalized descriptors (**ITEM 1**, **ITEM 2**, etc.), and a message is displayed when execution is finished.
- See Appendix B, Program TRANSLAT Format Translations for more information on character-delimited formats.

3. PEDSYS to Pedigree/Draw 4.4

When PEDSYS to Pedigree/Draw 4.4 (Mamelka, et al. 1993) format translation is chosen, Standard File List Displays appear, from which the file to be translated is chosen. The program compares the Code File of the file selected with entries in **CODE.STD** for standard mnemonics identifying Ego and Ego's parents (which are always present in a Master Code File). If the Code File *does not* contain standard mnemonics that identify Ego, parents and SEX (**SEQ**, **SSEQ** (or **FSEQ**), **DSEQ** (or **MSEQ**) and **SEX**, respectively) a Standard Item List appears with instructions to select the unidentified items. These must conform to format and content specifications described below. Pedigree/Draw does not require that gender be identified, so that it is not absolutely necessary to select this item.

When selection of IDs has been done, or if the Code File *does* contain the appropriate standard mnemonics, the following display appears:

```
 * 1 EGO SEQUENTIAL ID       8 NUMBER OF OFFSPRING    15 Dam's Permanent ID
 * 2 SIRE'S SEQ              9 PATERNAL SIBSHIP SIZ  *16 Sex (M, F or U)
 * 3 DAM'S SEQ              10 MATERNAL SIBSHIP SIZ   17 Exit Date (YYYYMMDD)
   4 FIRST OFFSPRING SEQ    11 FULL SIBSHIP SIZE      18 Exit Code
   5 NEXT PATERNAL SIB SE   12 EGO Permanent ID       19 Mate's Permanent ID
   6 NEXT MATERNAL SIB SE   13 Birth Date (YYYYMMDD   20 PEDIGREE NUMBER
   7 NEXT FULL SIB SEQ      14 Sire's Permanent ID    21 GENERATION NUMBER
--------------------------------------------------------------------------------
  TRANSLAT    Select text fields        File - MASTER.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to items to be included as optional text fields
(5 items maximum). Required text items are pre-selected. All items are
truncated if longer than 24 characters. Enter [C] to continue.
>
```

Pedigree / Draw 4.4 allows up to five 24-character text fields per record. Items selected from the Item List Display are entered into these fields left-justified. Inclusion of these items is optional. Note that Sequential IDs of Ego and parents, and SEX are pre-selected.

TRANSLAT also searches the input Code File for the mnemonics **NOREP** and **SYMBOL**, and will copy these items into the appropriate fields of Pedigree / Draw records. Default values are inserted into these fields if these mnemonics are not found in the Code File.

4. Pedigree/Draw 4.4 to PEDSYS:

When Pedigree / Draw to PEDSYS format translation is selected, a request for the appropriate file name is shown. All Pedigree / Draw fields are translated into PEDSYS format. These include items representing data added by Pedigree / Draw programs:

a. **Number of individuals represented**. This item is labeled 'Number represented' and given the mnemonic **NOREP** in output Code File **translat.cde**.

b. **Symbol codes**. This item is labeled 'Symbol codes' and given the mnemonic **SYMBOL** in output Code File **translat.cde**.

The Pedigree / Draw header record is not included in PEDSYS output file **translat.out**, but the pedigree title is copied from the header to the PEDSYS Code File Label Record.

Notes on PEDSYS - Pedigree/Draw 4.4 Conversion:

- *This conversion applies only to records for Pedigree/Draw Version 4.4.* Conversion between PEDSYS and Pedigree / Draw Version 5.0 files is done in program PREPDRAW.

- See Appendix B, Program TRANSLAT Format Translations, for more information on Pedigree / Draw 4.4 formats.

ONE-WAY CONVERSIONS

PEDSYS files may require some preparation before they can be converted to a form that can be used as input to genetic analysis programs CRI-MAP (Green et al. 1990), FISHER and MENDEL (Lange et al. 1988), LINKAGE (Lathrop and Lalouel 1984), PAP (Hasstedt 1989), S.A.G.E. (Elston et al. 1986), and SOLAR (Almasy and Blangero 1998). These requirements are:

a. Records from only one PEDSYS file may be used as input, which means that vital statistics and genealogical information must be combined with marker and quantitative data on the same record (using program MERGE, etc.).

b. Data required are IDs for Ego and Ego's parents, Ego's sex, pedigree ID, as well as phenotypes and genotypes.

c. Marker data must be in genotypic form. However, TRANSLAT will convert autosomal codominant phenotypes to genotypes if phenotype symbols conform to PEDSYS naming conventions (see Appendix C, Rules for Genotype and Phenotype Nomenclature), or if the loci are listed in a Genotype-Phenotype Map File (see Appendix A, Definitions, Data, and File Types).

After a file has been selected, TRANSLAT searches its Code File for the mnemonic SEX. If not found, the program terminates with an error message. TRANSLAT makes the default assumption that an item with the mnemonic PEDNO (produced by programs INDEX and COUNTPED) designates a family identification number. The program makes it possible to assign an alternate family identification number if the standard PEDSYS mnemonic is not found.

The program next compares the Code File of the input file with entries in **CODE.STD**, for standard mnemonics identifying Ego and Ego's parents. CRI-MAP requires Sequential IDs (see below), but some latitude is provided in designating IDs for FISHER, MENDEL, S.A.G.E.(REGC) and LINKAGE, where the following sequence appears if the Code File contains standard PEDSYS mnemonics for both Permanent and Sequential IDs:

```
Conversion to this format requires the presence of IDs for EGO, FA and MO  on
each input record.  Enter RETURN to use Permanent IDs or "S" for  Sequential IDs.
Enter "O" for selection of other IDs that might not be  labeled with PEDSYS
standard mnemonics, "Q" to quit without executing.
>
```

Notes on pedigree analysis conversions:

- A Phenotype-Genotype Map need not be created for autosomal codominant loci that adhere to the conventions listed in Appendix C, Rules for Genotype and Phenotype Nomenclature. Nonetheless, is it frequently helpful to generate a map for autosomal codominant loci, because TRANSLAT will otherwise be forced to read through the entire Data File twice for each locus (once to tally phenotypes, and again to do its conversions). Presence of a map file eliminates the need for the tallying step and can significantly speed execution.

- File **TRANSTEST.XMP** was created to illustrate conversions to input formats for pedigree analysis programs by combining items from records in **MASTER.XMP** (Items 1 - 5 and 11 - 13), and **QUANTGEN.XMP** (Items 6 - 9), and adding an item called Discrete Trait (Item 10) Its Code File **TRANSTEST.CDE** is shown below:

```
Example for Program TRANSLAT
 5 EGO Permanent ID      ID                          C
 5 Sire's Permanent ID   SIRE                        C
 5 Dam's Permanent ID    DAM                         C
 1 Sex (M, F or U)       SEX                         C
 5 PEDIGREE NUMBER       PEDNO                       I
 4 ADA  Phenotype        ADA    PHENO  TYPE          C
 4 APOB Pvu2a Genotype   APOB   Pvu2a  GENO   TYPE   C
 3 Tot. Ser. Cholesterol TSC                         I
 5 APOAI Level           APOAI  Level                R
 1 Discrete Trait        DISCRT TRAIT               I
 5 EGO SEQUENTIAL ID     SEQ                         I
 5 SIRE'S SEQ            SSEQ                        I
 5 DAM'S SEQ             DSEQ                        I
```

5. and 6. PEDSYS to FISHER and MENDEL formats

If FISHER or MENDEL format has been selected, TRANSLAT searches the input Code File for an item with the mnemonic **TWIN**, a one-character status flag; a single blank character is inserted in output records if no twin status flag is found.

Program FISHER expects input of quantitative phenotypic data, which are chosen from a standard Item Selection Display:

```
 * 1 EGO Permanent ID        6 ADA  Phenotype         11 EGO SEQUENTIAL ID
 * 2 Sire's Permanent ID     7 APOB Pvu2a Genotype    12 SIRE'S SEQ
 * 3 Dam's Permanent ID      8 Tot. Ser. Cholesterol  13 DAM'S SEQ
 * 4 Sex (M, F or U)         9 APOAI Level
 * 5 PEDIGREE NUMBER        10 Discrete Trait
--------------------------------------------------------------------------------
  TRANSLAT    Select quantitative data  TRANSTEST.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to QUANTITATIVE data items to be included in
output (decimal points will be added to integer values as needed). Required
items have been pre-selected. Enter [C] to continue.
>8-9
```

Items 1 through 5 are pre-selected and are protected from further change. Items 8 and 9 represent variables that may be selected for inclusion in the output file.

Program MENDEL requires genetic marker loci, chosen in its initial Item Selection Display:

```
 * 1 EGO Permanent ID        6 ADA  Phenotype         11 EGO SEQUENTIAL ID
 * 2 Sire's Permanent ID     7 APOB Pvu2a Genotype    12 SIRE'S SEQ
 * 3 Dam's Permanent ID      8 Tot. Ser. Cholesterol  13 DAM'S SEQ
 * 4 Sex (M, F or U)         9 APOAI Level
 * 5 PEDIGREE NUMBER        10 Discrete Trait
--------------------------------------------------------------------------------
  TRANSLAT    Select marker locus       TRANSTEST.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to items containing genotypes or phenotypes to
be included in output. Required items have been pre-selected. Enter "B" to
insert blanks, [C] to continue to next step.
>6-7
```

Here again, items 1 - 5 are preselected, while items 6 and 7 represent a loci that may be selected for output. Also provided is an opportunity to insert blank fields into the record, which can be filled in at a later time. When selection of markers is complete, quantitative variables are chosen from a display identical to that shown for FISHER above. When all data have been chosen, the following display appears:

```
Sort input file by pedigree ID and sibship?  [Y]/N
>
```

Subsequently, the program begins execution.

Notes on FISHER and MENDEL output:

- Because its records are not all the same length, **translat.tab** is not strictly speaking a PEDSYS Data File. In the case of standard FISHER and MENDEL output, however, records containing pedigree data are formatted identically after header records have been removed, and Code File **transtab.cde** is provided to describe their structure. See Appendix B, Program TRANSLAT Format Translations, page B-6 for more information on standard FISHER and MENDEL formats.

- The SFBR version of the standard FISHER output includes Code File **transtab.cde** in the output file along with the FISHER output as selected above, as well as several other records needed to control the shell program. See Appendix B, Program TRANSLAT Format Translations, page B-7 for an example of FISHER(SFBR) output.

7. PEDSYS to S.A.G.E.(REGC) format

S.A.G.E.(REGC) requires a leading constant on each input record identifying it as belonging to a particular data set. A query requesting this item appears after records have been ordered by Pedigree ID:

```
Enter a constant (up to 5 characters) to be used as Study ID.
>LPD01
```

Next, the program proceeds to selection of quantitative phenotypes:

```
   1 EGO Permanent ID        6 ADA  Phenotype        *11 EGO SEQUENTIAL ID
   2 Sire's Permanent ID     7 APOB Pvu2a Genotype   *12 SIRE'S SEQ
   3 Dam's Permanent ID      8 Tot. Ser. Cholesterol *13 DAM'S SEQ
 * 4 Sex (M, F or U)         9 APOAI Level
 * 5 PEDIGREE NUMBER        10 Discrete Trait
--------------------------------------------------------------------------------
  TRANSLAT    Select data               TRANSTEST.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to QUANTITATIVE data items to be included
in output. Required items have been pre-selected. Enter [C] to
continue to next step.
>8-9
```

Items 4 and 5, and 11 - 13 (Sequential IDs have been chosen) are preselected and are protected from further change. Items 8 and 9 represent quantitative data variables that may be selected for inclusion in

the output file. Two additional blank variables (**Variance Covariate** and **Conditioning Status**) are automatically added to the record if they are not listed in the Item Selection Display as input options.

Notes on S.A.G.E. output:

- In addition to the conventional PEDSYS output file **translat.out** and **translat.cde**, S.A.G.E.(REGC) translation produces two additional small files named **fsp.par**, and **sageregc.fmt**, respectively. See Appendix B, Program TRANSLAT Format Translations for more information about S.A.G.E.(REGC) formats.

- If variables Variance Covariate and Conditioning Status are included on records in the input file for translation to S.A.G.E.(REGC), they must be given single mnemonic words **VARCON** and **CNDSTT**, respectively. These mnemonics are assigned automatically in the output Code File **translat.cde** if the variables are not present on input records.

8. PEDSYS to CRI-MAP format

CRI-MAP requires a number of header records in its input data files, as well as PID and allele symbols in integer form. Since CRI-MAP is a linkage analysis program, at least two genetic marker loci are required in its input:

```
   1 EGO Permanent ID        6 ADA  Phenotype        *11 EGO SEQUENTIAL ID
   2 Sire's Permanent ID     7 APOB Pvu2a Genotype   *12 SIRE'S SEQ
   3 Dam's Permanent ID      8 Tot. Ser. Cholesterol *13 DAM'S SEQ
 * 4 Sex (M, F or U)         9 APOAI Level
 * 5 PEDIGREE NUMBER        10 Discrete Trait
--------------------------------------------------------------------------------
  TRANSLAT    Select locus              TRANSTEST.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to at least two items containing loci for
inclusion in output. Required items have been pre-selected. Enter [C]
to continue to next step.
>6-7
```

Items 4, 5 and 11-13 are preselected and are protected from further change. Items 6 through 9 represent data variables that potentially may be selected for inclusion in the output file, although in this example only Items 6 and 7 would be appropriate, as they are the only ones designated as genetic marker loci. When selection of loci is complete, the following display appears:

```
Enter RETURN to assume all loci are autosomal co-dominant or enter name of
alternate Genotype-Phenotype Map file (loci not listed in a map file are
assumed to be autosomal co-dominant).
>
```

Execution begins when the source of genotype-phenotype translation has been chosen.

Notes on CRI-MAP conversion:

- CRI-MAP requires that IDs of Ego and Ego's parents be in integer format.

- TRANSLAT converts genotype representation from PEDSYS to CRI-MAP format, where for each locus, allele symbols are replaced by integer values corresponding to their alphabetic order, and geno-

types are converted to numerical form. Missing allele symbols are represented by zeros in CRI-MAP format. See Appendix B, Program TRANSLAT Format Translations for more detail on CRI-MAP record and genotype format.

- When TRANSLAT finds a phenotype in the PEDSYS input file for which alleles have not been assigned, the phenotype symbol on Ego's record is changed to zeros, and an error message is written to file **translat.err**. The format of error messages is:

  Locus: <locus name> Record: <recno> Uninterpretable: <entry>

  where <recno> identifies the position of the record in the input file, and <entry> is the symbol that caused the error.

9. PEDSYS to PAP genotypes

TRANSLAT converts PEDSYS genotype coding to PAP (Hasstedt 1989) format. Records of output file **translat.out** contain EGO PID, original genotype/phenotypes, and converted genotype symbols. Other information (parental IDs, sex, family number, etc.) is not included.

```
 * 1 EGO Permanent ID      * 6 ADA  Phenotype         11 EGO SEQUENTIAL ID
   2 Sire's Permanent ID   * 7 APOB Pvu2a Genotype    12 SIRE'S SEQ
   3 Dam's Permanent ID      8 Tot. Ser. Cholesterol  13 DAM'S SEQ
   4 Sex (M, F or U)         9 APOAI Level
   5 PEDIGREE NUMBER        10 Discrete Trait
--------------------------------------------------------------------------------
  TRANSLAT    Select locus                    TRANSTEST.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to loci for which genotypes/phenotypes symbols
are to be converted to PAP form.  Required items have been pre-selected.
Enter [C] to continue.
>
```

EGO Permanent ID (Item 1) is preselected and protected from further change. Here, **ADA Phenotype** (Item 6) and **APOB Pvu2a Genotype** (Item 7) have been selected.

Note on PAP conversion:

- PAP genotypes are represented as integers whose value is determined by genotypic array position. See Appendix B, Program TRANSLAT Format Translations for more information on PAP genotype formats.

10. PEDSYS to LINKAGE format

Translation to LINKAGE format produces a PEDSYS file formatted correctly for input to the LINKAGE preprocessing routine *MAKEPED* (not to be confused with the PEDSYS program with the same name). Assignment of SEX and IDs of EGO, FA and MO fields is required if standard mnemonics for these items are not found in the input file. Otherwise, the first step in this conversion is to select the item representing EGO's affection status (that is, whether EGO is designated as an individual affected with a disease or other phenotype of interest):

```
 * 1 EGO Permanent ID        6 ADA  Phenotype         11 EGO SEQUENTIAL ID
 * 2 Sire's Permanent ID     7 APOB Pvu2a Genotype    12 SIRE'S SEQ
 * 3 Dam's Permanent ID      8 Tot. Ser. Cholesterol  13 DAM'S SEQ
 * 4 Sex (M, F or U)         9 APOAI Level
 * 5 PEDIGREE NUMBER        10 Discrete Trait
--------------------------------------------------------------------------------
  TRANSLAT    Ego Affection Status      TRANSTEST.XMP
--------------------------------------------------------------------------------
Enter number corresponding to Ego Affection Status. If item does not
exist, enter [C] to create field and continue.
>
```

If an item representing Affection Status has not been designated previously, a field is created (containing the value **0**) and added to the output record. Next, genetic loci are selected:

```
 * 1 EGO Permanent ID      * 6 ADA  Phenotype         11 EGO SEQUENTIAL ID
 * 2 Sire's Permanent ID   * 7 APOB Pvu2a Genotype    12 SIRE'S SEQ
 * 3 Dam's Permanent ID      8 Tot. Ser. Cholesterol  13 DAM'S SEQ
 * 4 Sex (M, F or U)         9 APOAI Level           *14 AFFECTION STATUS
 * 5 PEDIGREE NUMBER        10 Discrete Trait
--------------------------------------------------------------------------------
  TRANSLAT    Select locus              TRANSTEST.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to items containing loci for inclusion in output.
Enter [C] to continue to next step.
>
```

Here the ADA and APOB Pvu2a loci have been chosen.

Notes on LINKAGE format conversion:

- TRANSLAT converts genotype representation from PEDSYS to LINKAGE format, where allele symbols are replaced by integer values.
- PEDSYS sex codes are converted as needed to **1** and **2**, corresponding to males and females, respectively; unknown parental PIDs are converted from blanks to a single numeric **0**.
- TRANSLAT adds a leading '**X**' to Permanent IDs (Data Type **C**) if their first character is numeric.
- Fields in LINKAGE record format are space delimited. Alleles of genotypic fields are also space-delimited.

LINKAGE Error Messages and Displays

TRANSLAT detects a number of data conditions that force program LINKAGE to terminate without execution. When any of these conditions is encountered, TRANSLAT writes error messages to file **translat.err**. These messages are:

```
<ego>: One parent missing.
<ego>: No parents and no offspring.
<ego>: Affection Status not 0, 1, or 2.
<ego>: Sex code not 1 or 2.
<ego>: Parent not included in file as Ego.
```

11. PEDSYS to SOLAR(MIXED) format

SOLAR (MIXED) is an interim variance component analysis that requires preprocessing of PEDSYS data. Pedigree Number (PEDNO) and IBD index (IBDID) are required, and SEX must be expressed as **0**, **1** or **2** for U, M and F, respectively. If mnemonic IBDID is not found in the input Code File, the following display appears from which an item may be selected to represent this variable. Here Item 11, EGO SEQUENTIAL ID has been chosen:

```
   1 EGO Permanent ID        6 ADA  Phenotype         11 EGO SEQUENTIAL ID
   2 Sire's Permanent ID     7 APOB Pvu2a Genotype    12 SIRE'S SEQ
   3 Dam's Permanent ID      8 Tot. Ser. Cholesterol  13 DAM'S SEQ
   4 Sex (M, F or U)         9 APOAI Level
   5 PEDIGREE NUMBER        10 Discrete Trait
--------------------------------------------------------------------------------
  TRANSLAT    Select IBDID              TRANSTEST.XMP
--------------------------------------------------------------------------------
Enter number corresponding to IBDID.
>11
```

Next, one or more continuous traits are selected:

```
   1 EGO Permanent ID        6 ADA  Phenotype        *11 EGO SEQUENTIAL ID
   2 Sire's Permanent ID     7 APOB Pvu2a Genotype   *12 SIRE'S SEQ
   3 Dam's Permanent ID      8 Tot. Ser. Cholesterol *13 DAM'S SEQ
 * 4 Sex (M, F or U)         9 APOAI Level
 * 5 PEDIGREE NUMBER        10 Discrete Trait
--------------------------------------------------------------------------------
  TRANSLAT    Select Continuous traits  TRANSTEST.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to items containing CONTINUOUS TRAITS to be
included in output. Required items have been pre-selected. Enter [C] to
continue to next step.
>8
```

Here, Item 8, Total Serum Cholesterol, has been chosen. Next, discrete traits are selected:

```
   1 EGO Permanent ID        6 ADA  Phenotype        *11 EGO SEQUENTIAL ID
   2 Sire's Permanent ID     7 APOB Pvu2a Genotype   *12 SIRE'S SEQ
   3 Dam's Permanent ID    * 8 Tot. Ser. Cholesterol *13 DAM'S SEQ
 * 4 Sex (M, F or U)         9 APOAI Level
 * 5 PEDIGREE NUMBER        10 Discrete Trait
--------------------------------------------------------------------------------
  TRANSLAT    Select Discrete Traits    TRANSTEST.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to DISCRETE TRAITS to be included in output.
Required items have been pre-selected. Enter [C]to continue to the next
step.
>10
```

Here, Item 10, Discrete Trait, has been chosen. Finally, covariates ate selected, in this case Item 9, APOAI Level:

```
   1 EGO Permanent ID        6 ADA  Phenotype        *11 EGO SEQUENTIAL ID
   2 Sire's Permanent ID     7 APOB Pvu2a Genotype   *12 SIRE'S SEQ
   3 Dam's Permanent ID    * 8 Tot. Ser. Cholesterol *13 DAM'S SEQ
 * 4 Sex (M, F or U)       * 9 APOAI Level
 * 5 PEDIGREE NUMBER       *10 Discrete Trait
--------------------------------------------------------------------------------
  TRANSLAT    Select Covariates         TRANSTEST.XMP
--------------------------------------------------------------------------------
Enter numbers corresponding to items containing COVARIATES to be included
in output. Required items have been pre-selected. Enter [C] to continue
to next step.
>
```

At this point, TRANSLAT begins execution.

Notes on SOLAR (MIXED) format conversion:

- PEDNO and IBDID fields required, PROBND field optional
- The SEX field must only be 1 character wide and be an integer field.
- CONTINUOUS TRAITS, DISCRETE TRAITS and COVARIATES may not be represented as Data Type C.
- CONTINUOUS TRAITS may not be discrete, DISCRETE TRAITS may not be continuous.
- Execution is terminated if DISCRETE TRAIT value is other than blank, 0 or 1.
- COVARIATES may not also be selected as TRAITS.
- The PROBND field is optional; if present, (a) it must contain a non-zero, non-blank value, and (b) a FOCAL PROBAND TRAIT field containing a non-blank (continuous or discrete) value for each proband is required.
- FOCAL PROBAND TRAIT has to have been picked as continuous or discrete trait
- At least one individual per family must have complete data. For any family in which this is not the case, the following message is written to fine **translat.err**:

```
Family not included for lack of quantitative data.
```

Files:

Input: PEDSYS-to-delimited field format: Any PEDSYS Master File or Data File, and an associated Code File.

Delimited fields-to-PEDSYS format: A file containing records whose fields are separated by a unique character or sequence of characters.

PEDSYS-to-Pedigree/Draw 4.4-format: Any PEDSYS Master File or pedigree file, and an associated Code File.

Pedigree/Draw 4.4-to-PEDSYS format: Any Pedigree/Draw basic pedigree file.

PEDSYS-to-FISHER/MENDEL format: Any PEDSYS pedigree file containing quantitative phenotypes (and markers in the case of MENDEL), and an associated Code File.

PEDSYS-to-S.A.G.E.(REGC) format: Any PEDSYS pedigree file containing marker phenotypes, and an associated Code File.

PEDSYS-to-CRI-MAP, PAP, and LINKAGE formats: Any PEDSYS pedigree file containing marker phenotypes and its associated Code File, a Genotype-Phenotype Map File.

Output: PEDSYS-to-delimited-field format: **translat.tab**.
Delimited fields to PEDSYS format: **translat.out** and **translat.cde**.

PEDSYS-to-Pedigree/Draw format: **translat.tab**.
Pedigree/Draw to PEDSYS format: **translat.out** and **translat.cde**.

PEDSYS-to-FISHER/MENDEL format: **translat.tab** and **transtab.cde**.

PEDSYS-to-S.A.G.E.(REGC) format: **translat.out**, **translat.cde**, **fsp.par** and **sageregc.fmt**.

PEDSYS-to-CRI-MAP format: **translat.tab**, **translt2.tab** and **translat.err**.

PEDSYS-to-PAP or -LINKAGE format: **translat.out**, **translat.cde**, **translt2.tab** and **translat.err**.

PEDSYS-to-SOLAR (MIXED) format: **translat.tab**, **transtab.cde** and **translat.err**.

Note:

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **translat subset.out**.

References

Almasy L, Blangero J (1998) Multipoint quantitative trait linkage analysis in general pedigrees. American Journal of Human Genetics 62:1198-1211.

Green P, Falls K, Crooks S 1990 Documentation for CRI-MAP, Version 2.4. Department of Genetics, Washington University School of Medicine, St. Louis, MO 63110.

Elston RC, Bailey-Wilson JE, Bonney GE, Keats BJ, Wilson AF 1986 S.A.G.E. - A package of computer programs to perform statistical analysis for genetic epidemiology. Seventh International Congress of Human Genetics (Berlin) Abstracts, p 389.

Hasstedt SJ 1989 Pedigree Analysis Package. Rev 3.0 Department of Human Genetics, University of Utah Medical Center.

Lange K, Weeks D, Boehnke M 1988 Programs for pedigree analysis: Mendel, Fisher and dGene. Genetic Epidemiology 5:471-472.

Lathrop GM, Lalouel JM 1984 Easy calculation of lod scores and genetic risks on small computers. American Journal of Human Genetics 81:3443 -3446.

Mamelka PM, Dyke B, MacCluer JW 1993 *Pedigree/Draw for the Apple Macintosh*. Population Genetics Laboratory Technical Report No. 1, Southwest Foundation for Biomedical Research, San Antonio, TX 78284. Second edition, 81 pp.

Program Limits:

A maximum of 500 items may be defined for any one PEDSYS record; a maximum of 30 loci may be converted in any one run.

TRIPLETS

Program TRIPLETS extracts selected data from the separate records of Ego, father and mother and places this information on a single record. The file from which triplets are generated may be either a Master File, or a pedigree file consisting minimally of records containing IDs for Ego and Ego's parents. Triplets may be constructed for probands entered from either the keyboard, or from a Proband Input File (a file consisting of a list of either Sequential or Permanent IDs)

Introductory Display:

```
PEDSYS program TRIPLETS - Copyright 1992, 1999 SFBR

TRIPLETS assembles data for Ego, father and mother on the same record.

  1.  Unlinked File
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter number, file name, or "Q" to quit.
>2
```

Case 1. Triplet data extracted from one or more Linked Files

If files are linked, data may be extracted from more than one file. If this option is chosen, the following display appears from which files containing the data may be selected:

```
Linked Files:

    1.  Example Master File
*   2.  Markers + Quant. P'types
    3.  Blood Sample Inventory
*   4.  Hemoglobin Levels

Data from more than one LINKED file may be selected for inclusion in an
output record. Enter numbers corresponding to file(s) above, in the order
that items are to be extracted, or RETURN to continue to next step.
>
```

In the example above, the Markers + Quantitative Phenotypes and the Hemoglobin Levels files have been marked with an asterisk, indicating that information will be extracted from both files.

Case 2. Triplet data extracted from a single Unlinked File

If an unlinked Data File is chosen that includes pedigree information, that is, if its Code File contains standard mnemonics EGO (ID), SIRE (FA) and DAM (MO), the following query appears:

```
Unlinked Files

   1. EXAMPED.XMP
   2. PEDTRTEST.XMP
   3. TRANSTEST.XMP
   4. WEIGHTS.XMP

Enter a file number or name.
>1

Use pedigree information found in EXAMPED.XMP?  [Y]/N
>y
```

Here, file **EXAMPED.XMP** has been selected. This file contains full pedigree information, which will be used in constructing triplets. If, however, a file is chosen for which the Code File does not contain the standard mnemonics, the following instruction appears:

```
Pedigree information is not present in file WEIGHTS.XMP

 1.  Unlinked File
 2.  Example Master File

Enter number of file containing pedigree information.
>2
```

In the example above, IDs of Ego and Ego's parents were not found in file **WEIGHTS.XMP** from which triplet data are to be extracted. The Example Master File (choice **2**) has been selected to supply pedigree information.

When a pedigree file has been chosen, an Item List Display of the first Data File selected appears, from which items may be chosen for inclusion in the output records generated by the program:

```
 * 1 Ego ID                  3 APOB Pvu2a Genotype     5 APOAI Level
   2 ADA  Phenotype        * 4 Tot. Ser. Cholesterol
-------------------------------------------------------------------------------
  TRIPLETS    Items for EGO                  File - QUANTGEN.XMP
-------------------------------------------------------------------------------
Enter item numbers in the order they are to be included in the output record,
"A" to select all, or [C] to continue.
>
```

Here, the first file selected was the Markers + Quantitative Phenotypes file, from which the Tot. Ser. Cholesterol Phenotype has been chosen. EGO PID has been preselected. These will be the first items to appear on the output record. Item List Displays of subsequent files appear in the order they are selected:

```
   1 Ego                   * 3 Hemoglobin Level
   2 Sample Date             4 INTERNAL POINTER
--------------------------------------------------------------------------------
  TRIPLETS    Items for EGO                   File - HEMOGLOB.XMP
--------------------------------------------------------------------------------
Enter item numbers in the order they are to be included in the output record,
"A" to select all, or [C] to continue.
>
```

Here, Hemoglobin Level has been selected from the Hemoglobin Levels file. Note that EGO PID is *not* preselected here. Once items for Ego have been chosen, the following display appears:

```
Enter same items for SIRE as selected for EGO?  [Y]/N
>y

Enter same items for DAM as selected for SIRE?  [Y]/N
y
```

Here, the same items selected for Ego (PID, Tot. Ser. Cholesterol and Hemoglobin Level) have been selected for both parents. Had an **n** been entered at either point in the display above, opportunities for selecting a different set of items for the parents would have been given. The files from which these items may be selected are limited to those selected initially. Next, the following display appears, from which individual probands that define the triplets are selected:

```
Pedigrees may be constructed for probands entered from

  1.  Keyboard
  2.  Unlinked File
  3.  EXAMPLE
  4.    LOCUS Sub-directory

Enter number above corresponding to the appropriate source of proband IDs.
>1
```

Source 1: ***Proband IDs entered from the keyboard.***

Choice 1 (shown above) specifies that proband IDs are to be entered from the keyboard. When IDs have been extracted from a Master File, the form of the IDs (Permanent or Sequential) must be specified before keyboard entry begins:

```
Enter - [P] if probands are identified by Permanent IDs
         S  if probands are identified by Sequential IDs
>s
```

Here, Sequential IDs have been selected.

Source 2, 3 or 4*: *Proband IDs read from a Proband Input File.

A proband file is often a subset of the input pedigree file (typically a Master File), but may be any Data File whose records contain IDs in the appropriate form (that is, Permanent or Sequential). Selection of a single item containing the proband ID is made from a Standard Item List Display:

```
   1 EGO Permanent ID        5 Sex (M, F or U)         9 PEDIGREE NUMBER
   2 Birth Date (YYYYMMDD    6 Exit Date (YYYYMMDD)   10 GENERATION NUMBER
   3 Sire's Permanent ID     7 Exit Code
   4 Dam's Permanent ID    * 8 Mate's Permanent ID
--------------------------------------------------------------------------------
  TRIPLETS    Select Proband ID               File - subset.out
--------------------------------------------------------------------------------
Enter a number above corresponding to Proband ID, or [C] to continue to
next step.
>
```

Here, the pedigrees of the mates of individuals found in file **subset.out** have been specified. The program next proceeds to construct triplets.

Notes:

- Data from only the most recently entered record will be included in output if a Multiple-entry Data File has been selected for input to TRIPLETS.

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **triplets subset.out**.

Files:

Input: Pedigree Information: Master File, pedigree contained in Data File itself, or an unlinked pedigree file containing EGO PIDs that also appear in Data File.

Data File: Linked Data File, or unlinked Data File .

Probands: Master File, Data File, or console keyboard.

Output: **triplets.out** contains one or more pedigrees connecting probands to their relatives according to limits specified on input.

triplets.cde defines the record structure of **triplets.out**.

VERIFIER

Program VERIFIER checks any PEDSYS Master File or Data File for illegal data characters and values that are inconsistent with Code File definitions.

Introductory Display:

```
PEDSYS program VERIFIER - Copyright 1992, 1999 SFBR

VERIFIER checks to verify that all fields contain entries consistent with their
type (C, I, R, or D); date fields contain only integers (3, 4, 7 or 8 charac-
ters); character fields contain only printable characters; numerical fields
contain only numbers, leading blanks, and one leading + or -; real number
fields may also contain a single decimal point.  In addition, a Master File
is checked to verify that field widths of EGO, SIRE and DAM PID's are the
same (3-32 characters); birth date is earlier than exit date; and SEX field
contains only 1,2,3; M,F,U or m,f,u.

  1.  Unlinked File
  2.  EXAMPLE
  3.    LOCUS Sub-directory

Enter the number of the file or directory to use.
>
```

When **RETURN** is pressed, a Standard File List Display appears.

Error Messages:

File **verifier.tab** contains the name of the file that was tested (with its accompanying Code File), identifies each record and field in which an error occurs, and gives a description of the error.

Note:

- The command line startup shortcut (program name, followed by a blank and an input file name) may be used to start this program. Example: **verifier subset.out**.

Files:

Input: Any Master File or Data File with an accompanying Code File. If the file to be verified is a Master File, **CODE.STD** is required.

Output: **verifier.tab**

APPENDIX A. Definitions, Data, and File Types

A. Definitions

We have tried to avoid excessive use of jargon, but there are a few useful terms that require definition:

Record - A unit of data storage in which information is kept (usually for a single individual). PEDSYS records are a maximum of 1024 bytes (characters) long and contain only printable characters (ASCII characters 32 - 126) except for the end-of-record mark (EOR) which consists of a carriage return (ASCII character 13) in the case of the Apple Macintosh and Unix machines, or carriage return and line feed (ASCII characters 13 and 10) in the case of MS-DOS and Windows.

File - A collection of records. In most cases all records in a PEDSYS file have the same length and format, except for the last record which may consist of a single end-of-file mark (EOF). This is a CTRL-Z (ASCII character 26).

It is useful to make a distinction between units of data storage (records), classes of information, and values that data can take. Consequently, we define the following units:

Field - A subunit of a record in which a single item of information is stored. Up to 100 fields are permitted in a single record; maximum field width is 99 characters.

Item - A named class of information (usually pertaining to an individual) that is stored in a single field of a record. Also referred to as a *variable.*

Entry - The actual contents of an item. Also referred to as a *value*. For example, we refer to Birth Date as an item of information found in the thirteenth field of a Master File record. The number '19821103' (corresponding to November 3, 1982) might represent the Birth Date entry for a particular individual.

Ego - Generally, the term Ego refers to an individual from which genealogical relations are determined in a pedigree. In some cases this may be equivalent to the medical genetics term **proband**, or the individual whose phenotype motivated construction of the pedigree. In PEDSYS terminology, the individual whose vital statistics appears on each Master File record is referred to as Ego.

B. Data Types

Information in PEDSYS files is stored as character data. However, these data are interpreted by PEDSYS programs according to codes specified in a Code File. Data items may be of the following types (shown with their codes):

C Any printable character (ASCII characters 32 - 126). Normally these will be left justified, but because all blanks are honored, right justification may be forced.

I Numerical integers. Blanks are treated as zeros except in programs CALC and TALLY, where blanks may signify missing values. Integers are right justified. PEDSYS integer input (from disk or keyboard), output (to disk or printer) and computations (internal to the processor) are limited to numbers in the range -2,147,483,648 to 2,147,483,647.

R Floating point (decimal) numbers. Blanks are treated as zeros in PEDSYS floating point computations except in programs CALC and TALLY, where blanks may signify missing values. Floating point numbers are also right justified. PEDSYS floating point input (from disk or keyboard) is limited to ten digits excluding the decimal point (.0000000001 through 9999999999. in

absolute value). Output is formatted as F12.5, giving a range of 0.00001 through 999999.99999 in absolute value. Internal computations are normally done in double precision floating point arithmetic, giving an absolute range of 2.23E-308 through 1.79E308 with precision of one part in 10E16 on the IBM-PC.

D Dates. Dates are kept in Gregorian numerical form with order year-month-day, as specified by International Standard ISO 8601. The ISO 8601 standard primary notation is YYYY-MM-DD, but the standard also specifies alternate formats, including omission of hyphens (YYYYMMDD), the convention used in PEDSYS. Like the ISO 8601 standard, PEDSYS permits representation of year without month or day.

PEDSYS date formats differ from the ISO 8601 standard as follows:

- Although four-digit representation of year (YYYYMMDD and YYYY) are preferred, PEDSYS will accept and compute with the shortened three-digit year forms YYYMMDD, and YYY. Calculations using these formats assume that dates lie between A.D. 1200 and A.D. 2199. A leading zero in three-digit representation of the year implies a third millenium date (i.e., **001** = **2001**).

- PEDSYS does not allow representation of year and month alone (YYYYMM, etc.)

- PEDSYS does not allow two-digit representation of year (YYMMDD, or YY alone).

PEDSYS programs ignore date fields that are filled with blanks or zeros (event has not yet occurred) or nines (event has occurred, but date is unknown).

C. File Types

1. Program Files

The PEDSYS database system currently consists of more than 40 independently executable programs, each designed to perform a single task. Each program is stored as a single file which can be created only from compiled FORTRAN programs.

- With the Unix operating system, an executable file requires no extension. Unix usage is case-sensitive, and by convention we use lower case for executable file names. For example, **merge** is the name of the Unix executable file containing PEDSYS program MERGE.

- With Windows or MS-DOS, an executable file is named **<progname>.EXE**, where **<progname>** is the name of the program, and **.EXE** is an extension reserved for MS-DOS executable files (note that MS-DOS usage is not case-sensitive, so that file names always appear in upper case). For example, **MERGE.EXE** is the name of the MS-DOS executable file containing PEDSYS program MERGE.

- With Macintosh OS, an executable file requires no extension, although it is associated with the icon denoting that it is executable (-). Macintosh OS usage is case-sensitive, and by convention executable file names are upper case.

2. The Master File

The Master File consists of records containing basic information about individuals in a population. The most fundamental items in each record are Permanent IDs (PIDs) of Ego (the individual described

by the record) and his or her parents. Permanent IDs are designated as character data (that is, they need not be numerical), and usually form an unordered list in the Master File. To eliminate the need to make repeated time-consuming searches through the entire list of unordered Permanent IDs when processing extended pedigrees, PEDSYS creates Sequential ID numbers for Ego and selected members of Ego's nuclear family. Sequential IDs (the first seven items in a Master File) serve as pointers (record numbers) designating the location of records of these family members in the Master File. This scheme permits very rapid construction and analysis of large pedigrees, since each individual record contains pre-determined links to records of ascending (father and mother), descending (first offspring), and lateral relatives (maternal, paternal and full sibs).

Master Files may be created in two ways:

- A new Master File may be started by entering basic information about each individual from the keyboard, using PEDSYS program ALTMASTR. This program produces a specially formatted temporary update file that is used as input to program SETMASTR. SETMASTR adds Sequential ID numbers (and other items) to the basic information, and creates a functional Master File formatted as shown in the table below. The ALTMASTR - SETMASTR sequence is also used to update pre-existing Master Files.

- An ASCII text file of individual records, each containing at minimum a unique PID for Ego and Ego's parents, and a code designating sex may be transformed into Master File format. Some pre-processing may be required if records are incorrectly coded or formatted or contain items that more appropriately belong in a Data File. Program INDEX, and adherence to the conventions of the **CODE.STD** file provide guides to data preparation for Master Files.

Most programs require that Master Files be named **MASTER.<EXT>**, where .<EXT> is an extension identifying the population. For example **MASTER.XMP** is the name of the Example Master File distributed with the PEDSYS package.

Twenty standard items of a Master File are formatted as shown below.

Notes on Master File Record Format:

- Mnemonics are invariant, although allowances are made for differences between human and non-human genealogical terminology ("father" vs. "sire", etc.).

- Items 1 - 12 are fixed in position. Items 1-11, 16, 19 and 20 are invariant in length.

- Items 12, 14, 15 - Maximum length of Permanent IDs may range from 3 to 9 characters, but must be the same in all records of a given Master File. Actual length of entry may not exceed length specified in Code File (see below).

- Items 13, 17 - See date specifications in Data Types section above.

- Item 17 - Because Exit Dates are not available for some populations, this is not a required item in the Master File. Nonetheless, its inclusion is strongly advised, and must follow specifications as listed in the table below.

Item No.	Length	Item name	Mnemonic	Type
1	5	Ego Sequential ID	SEQ	I
2	5	Father's SEQ	FSEQ (or SSEQ)	I
3	5	Mother's SEQ	MSEQ (or DSEQ)	I
4	5	First Offspring SEQ	KID1	I
5	5	Next Paternal Sib SEQ	PSIB	I
6	5	Next Maternal Sib SEQ	MSIB	I
7	5	Next Full Sib SEQ	FSIB	I
8	3	Number of Offspring	NKID	I
9	3	Paternal Sibship Size	PSIBSZ	I
10	3	Maternal Sibship Size	MSIBSZ	I
11	3	Full Sibship Size	FSIBSZ	I
12	3 - 9	Ego Permanent ID	EGO (or ID)	C
(13)	3 - 8 *	Birth Date (Yr,Mo,Da)	BIRTH (or BIR)	D
(14)	3 - 9	Father's Perm. ID	FA (or SIRE)	C
(15)	3 - 9	Mother's Perm. ID	MO (or DAM)	C
(16)	1	Sex (M,F,U or 1,2,3)	SEX	C
(17)	3 - 8 *	Exit Date (Yr,Mo,Da) **	EXIT	D
(18)	1 - 3	Exit Code **	EXCODE	C
(19)	5	Pedigree ID Number	PEDNO	I
(20)	5	Generation Number	GEN	I
...				

( ) position not fixed * Eight-digit dates preferred ** Optional

- Item 18 - The need to specify cause of exit may vary depending on the population in question. In some studies it may be sufficient to distinguish the living from the dead, which in some cases can be signified by the presence or absence of an Exit Date (with no Exit Code needed), or by a simple binary Exit Code (i.e., **A** for alive, **D** for dead) alone or in combination with an Exit Date. In other cases, a "lost to follow-up" code may be needed, while in others Exit Codes may specify detailed causes of death, etc. See Dyke and Mamelka (1989) for a demographic analysis package that uses this coding scheme. A set of standardized Exit Codes has been proposed for nonhuman primate populations (Dyke 1993). In any case, Exit Codes are optional, and may be defined arbitrarily.

- Items 19 and 20 - Pedigree Identification Number and Generation Number are added automatically to Master File records by program INDEX.

- **...** signifies that a Master File may contain additional items.

- Many pedigree analysis programs require that records of parents appear before those of their offspring in population files. This order is automatically enforced when producing Master Files with program INDEX or the ALTMASTR- SETMASTR sequence.

3. Data Files

PEDSYS uses Data Files for storage of information not kept in the Master File. PEDSYS programs assume that for each individual represented in a Data File there is a corresponding entry in the Master File to which it is formally linked, but not vice versa -- that is, there need not be Data File entries for all individuals in a Master File. There are two types of Data Files:

a. *Single-entry Data Files* - contain items for which a single value is recorded for each individual (birth weight, age at first mating, etc.). These files may be created and manipulated with many

PEDSYS programs quite independently of a Master File (or in fact, without the existence of a Master File). When a Data File is linked to a Master File, the first item in each record must be the Ego Permanent ID (mnemonic EGO or ID), formatted exactly as it is in the Master File. Subsequent items may be defined arbitrarily. Single-entry Data Files are normally created and updated with the PEDSYS program ALTERDAT, but may be produced with any text editor, or may be imported from other database systems.

b. *Multiple-entry Data Files* - (sometimes called Historical Files) contain items for which more than one value is recorded during the lifetime of the individual (body weight, cage location, etc.). A separate record is added to the end of this file each time an entry is made for the individual. The first item in each record must be the Ego Permanent ID (mnemonic EGO or ID), formatted exactly as it is in the Master File; subsequent Data items may be defined arbitrarily. The last item is an internal pointer (a number designating the location in the file of the previous entry for the individual). This pointer takes the value 0 for the individual's first (or only) entry.

The internal pointer is an exception to the principle of including only data in PEDSYS Data Files. However, these fields may easily be ignored when using PEDSYS programs, and may be skipped when information is exported to other programs, because they are the last items in the record and are explicitly identified in screen displays.

Multiple-entry Data Files are normally created and updated with the PEDSYS program ALTERDAT, although any text editor can be used providing that pointers are maintained correctly (a risky assumption). Programs ALTERDAT and DATALINK assign and process internal pointers automatically.

Linked Data Files conventionally are named **<FILENAME>.<EXT>**, where <FILENAME> is an arbitrarily chosen name identifying the data set (upper case only), and .<EXT> is an extension identifying the population. For example **MARKERS.XMP** is the name of one of the Example Data Files distributed with the PEDSYS package.

4. Code Files

PEDSYS programs require information about the structure of the records they process (that is, the format, description, and data types of the items they contain). This information is passed to the programs from a special purpose Code File associated with each Master and Data File. Although we refer to Master Code Files and Data Code Files, the basic composition of the two types is the same. The first record of a Code File contains a label that identifies the file defined by the Code File. Subsequent records contain descriptors that define the format and contents of items in the Master File or Data File. All records in a Code File must be exactly 54 characters long.

Code Files normally are created with the PEDSYS programs CODE or ITEMIZE, although any text editor can be used if care is taken to observe format requirements.

Although the label contained in the first Label Record may be the full 54 characters long, a limit of about half this length is preferred. This label may be the file name, or (preferably) a more descriptive title. Subsequent Descriptor Records are formatted as follows:

```
Key    (a)          (b)           (c)    (c)    (c)    (c) (d)
Map    ••o••••••••••••••••••••o••••••o••••••o••••••o••••••o•
       |  |                    |      |      |      |      |
Start  1  4                   26     33     40     47     54
Format    (I2,1X,A21,4(1X,A6),1X,A1)
```

Where • represents a character in a field read by PEDSYS programs.
o represents a single blank used as a visual separator ignored by PEDSYS programs.
| is a visual marker pointing to the first character in each field. Numbers below this marker represent the starting column of each field.

Notes on Code File Record Format:

(a) The first field in the Descriptor Record (columns 1-2) contains the field length of the data item.

(b) The second field is an Item Label (columns 4-24) containing a description of the data that is used in screen and print display. Program operation is not affected if the field is left blank.

(c) The next four fields (columns 26-31, 33-38, 40-45, 47-52) contain item mnemonic "words". Mnemonics are used by PEDSYS programs to locate standard items in records and as labels in printed listings. In some cases, the mnemonic words are specified explicitly (see **a. The Master Code File** and **d. Code Files for Genetic Marker Data Files** below). Otherwise, other mnemonics may be chosen arbitrarily, with the restriction that the first mnemonic word should not be left blank. It is usually best not to duplicate the first and second mnemonic words within a given Code File, as well.

(d) The last field (column 54) contains the data type code (C, D, I, and R). Character data is assumed if this field is left blank; however, it is a better idea to define this type (Code C) explicitly, since it is difficult be sure that the record is the correct length without a visible character in column 54.

The Master Code File

Master Code Files are named **CODE.<EXT>**, where .<EXT> (upper case only) is an extension identifying the population. Here is an example of the Example Master Code File (**CODE.XMP**):

```
Example Master File
 5 EGO SEQUENTIAL ID     SEQ                    I
 5 SIRE'S SEQ            SSEQ                   I
 5 DAM'S SEQ             DSEQ                   I
 5 FIRST OFFSPRING SEQ   KID1                   I
 5 NEXT PATERNAL SIB SEQ PSIB                   I
 5 NEXT MATERNAL SIB SEQ MSIB                   I
 5 NEXT FULL SIB SEQ     FSIB                   I
 3 NUMBER OF OFFSPRING   NKID                   I
 3 PATERNAL SIBSHIP SIZE PSIBSZ                 I
 3 MATERNAL SIBSHIP SIZE MSIBSZ                 I
 3 FULL SIBSHIP SIZE     FSIBSZ                 I
 5 EGO Permanent ID      EGO                    C
 8 Birth Date (YYYYMMDD) BIRTH                  D
 5 Sire's Permanent ID   FA                     C
 5 Dam's Permanent ID    MO                     C
 1 Sex (M, F or U)       SEX                    C
 8 Exit Date (YYYYMMDD)  EXIT                   D
 1 Exit Code             EXCODE                 C
 5 Mate's Permanent ID   MATE                   C
 5 PEDIGREE NUMBER       PEDNO                  I
 5 GENERATION NUMBER     GEN                    I
```

Important note: mnemonic words defining some of the items in a Master File are invariant (see Master File requirements above, and the Standard Mnemonics File below).

Code Files for Linked Data Files

If Data Files are formally linked to Master Files (see the description of Data Pointer Files, and instructions for PEDSYS program DATALINK), the associated Code File must be named **<FILENAME>.CDE**, where <FILENAME> is the name of the associated Data File. For example, **SAMPLES.CDE** is the name of the Code File associated with the Example Data File **SAMPLES.XMP** distributed with the PEDSYS package.

Code Files for Unlinked Data Files

There are no fixed rules about naming unlinked Data Code Files, but they conventionally take the form **<filename>.cde**, where **<filename>** is an arbitrarily chosen name that identifies both the population and the relevant data set. It is a good idea to make sure that the first item on each record is a PID if there is any chance that they will be linked to a Master File in the future.

Code Files for Genetic Marker Data Files

Data Files that contain genetic marker phenotypes and/or genotypes require no special configuration, but if genetic data are to be displayed and analyzed correctly by programs GENCHECK, GENEFREQ and INFER, some mnemonic words defining the phenotypes may have reserved forms as follows:

Mnemonic Word 1

This word consists of the name of the locus in full, or in abbreviated form. Examples: **ABO**, **ADA**, **APRT**, **G6PD**, etc.

Mnemonic Words 2 - 4

The locus is assumed to be in *phenotypic* form unless any one of the the next three mnemonic words is **GENO** or **G'TYPE**, in which case the marker is in *genotypic* form.

Here are some examples of mnemonic word sets for genetic markers:

Word 1	2	3	4	Interpretation
ABO	P'TYPE			phenotype
ADA	PHENO	TYPE		phenotype
APOB	Pvu2a	GENO	TYPE	genotype
CETP	Geno	Type		genotype
LPL	G'type			genotype

5. The Master Pointer File

As explained above, Sequential ID numbers in PEDSYS Master File records serve as pointers to the records of family members in the file. Although these numbers make processing of pedigrees very efficient, they are difficult to remember, and it is usually more convenient to identify individuals in a data set by their Permanent IDs. The Master Pointer File contains a record for each individual in the Master File. The two items of this record are Ego's Permanent and Sequential IDs separated by a single

blank character used as a visual separator. The FORTRAN format for the record is (A<f>,1X,I5), where <f> is the length of the Permanent ID as specified in the Code File. The Master Pointer file is sorted by Permanent ID, resulting in an ordered list or index that can be searched quickly to find any Permanent ID and its corresponding Sequential ID number. The Sequential ID points to the location of Ego's Master File record, from which pointers to other relatives can be used to assemble Ego's extended pedigree.

Master Pointer Files are automatically produced by the PEDSYS programs INDEX and SETMASTR, but also may be created from a Master File by the appropriate use of SORT and REFORMAT.

Master Pointer Files are named **MPOINT.<EXT>**, where .<EXT> is an extension identifying the population. For example, **MPOINT.XMP** is the name of the Example Master Pointer File distributed with the PEDSYS package.

6. The Data Pointer File

Although PEDSYS Master Files and Data Files are in most cases independent of one another, it is often necessary to access information for the same individual, but kept in separate files. There are several ways to do this, but the most flexible is to make sure that the Data Files are linked to the Master File through a Data Pointer File. A Data Pointer File contains a record for each individual in the Master File. This record is made up of a variable number of fields, the first of which must be Ego's Sequential ID number, followed by data pointers to records in Data Files that contain information about Ego (one data pointer for each linked Data File). The FORTRAN format of this record is (<n>I5), where <n> is the number of linked Data Files plus 1. In the case of a Multiple-entry Data File, the data pointer identifies the record containing the last entry for Ego. Previous entries can then be identified by means of the internal pointers described above. If there is no entry for Ego in a Data File, the data pointer will have a value of 0. Data Pointer Files are normally created with the PEDSYS program DATALINK. A new field is added to all records in the Data Pointer File each time another Data File is linked to a Master File.

Data Pointer Files are named **DPOINT.<EXT>**, where .<EXT> is an extension identifying the population. For example, **DPOINT.XMP** is the name of the Example Data Pointer File distributed with the PEDSYS package.

7. Reusable Output Files

Most PEDSYS programs direct their output to disk files. An important distinction between output files and other Data or Code Files is that they are designated as reusable files, that is, files that are erased and rewritten each time a particular program is run. This means that data from the previous run of the program is lost when the program is executed again unless the data has been copied into other files, or the output files have been renamed. However, output files from one program may be used as input to a different program without copying or renaming, and some programs permit recursive use of output files (for example, file **infer1.out** may be used as input to program INFER). PEDSYS programs produce several types of reusable output data files.

a. Data Output Files

Most PEDSYS output consists of records for individual population members. These files share most of the characteristics of unlinked Single-entry Data Files. They are named **<progname>.out** (upper case on MS-DOS machines), where **<progname>** is the name of the program that produces the output and **.out** is an extension that identifies the file as a Data Output File. When a Data Output File is produced by any PEDSYS program, an accompanying Code File is generated and saved on the disk, as well. Such a Code File is named **<progname>.cde**, (**.CDE** in MS-DOS), where **<progname>** is the name of the program that produces the output and **.cde** is an extension that identifies the file as an output

Code File. Some programs produce more than one Data Output File, in which case file names are displayed on the screen when processing is done. Examples are **nomerge1.out** and **nomerge2.out** which are produced in addition to **merge.out** by the program MERGE.

b. Error Files

Several PEDSYS programs check for internal consistency of information in individual records, or in pedigrees. When found, errors are usually displayed on the screen and saved in an output file. Error Files may have one of two configurations. The most common is a simple text file with the extension **.err**, consisting of records bearing an error message of varying length and format. No accompanying Code File is provided. PEDSYS programs that produce Error Files of this sort are AGE, ANCESTOR, COMBINE, COUNTPED, DATALINK, DELRECS, GENEFREQ, INDEX, MAKEPED, MERGE, NEWITEM, PREPDRAW, SETDATA, SETMASTR, and SIBSHIP. The other configuration is a PEDSYS file consisting of strictly formatted error messages, and which has an accompanying Code File. These files are named **<progname>err.out** and **<progname>err.cde**, respectively, where <progname> is a PEDSYS program name or its abbreviation. Program INFER produces an Error File of this kind.

c. Log Files

In many File Management Programs it is useful to keep an account of changes and updates made to individual records. These changes are saved in an output text file which has the extension **.log**, consisting of records bearing a message of varying length and format. No accompanying Code File is provided. PEDSYS programs that produce Log Files of this sort are ALTERDAT, DELRECS, INFER and SETMASTR.

d. Tabular Output Files

A few programs compute summary statistics from population or family data. Output from these programs has a tabular format, which is saved (without an accompanying Code File) in an output file having the extension **.tab**. PEDSYS programs that produce Tabular Output Files are ANCESTOR, CODE, GENCHECK, GENEFREQ, LIST, NEWITEM, PREPDRAW, SEGCHECK, TALLY and TRANSLAT.

e. Miscellaneous Output Files

Some PEDSYS programs produce special-purpose output files in addition to standard **.cde**, **.out**, **.err** and **.log** files:

Program INDEX produces a file named **index.mpt** which, when renamed, becomes a Master Pointer File.

Program ALTERDAT produces files named **<filename>.UPD** and **<filename>.DEL** that are used in turn as input to programs SETDATA and DELRECS, respectively. The ALTERDAT - SETDATA program sequence is used to update records, and the ALTERDAT - DELRECS sequence is used to delete records from Data Files.

Program ALTMASTR produces two output files named **UPDATE.<EXT>** and **DELETE.<EXT>** that are used in turn as input to programs SETMASTR and DELRECS, respectively. The ALTMASTR - SETMASTR / DELRECS program sequence is used to make changes in Master Files.

The S.A.G.E.(REGC) option of program TRANSLAT produces files named **fsp.par** and **sageregc.fmt**, each of which contains a single FORTRAN format statement used as input to S.A.G.E.(REGC).

Names of these special-purpose output files are displayed on the screen when processing is done. Associated Code Files are not produced.

Note: See also Section 14, **Temporary System Files** starting on page xxxx below.

8. Backup Files

Several of the File Management Programs produce output files that are updated or edited versions of pre-existing Data or Master Files. These programs automatically replace the old files with new versions, but for safety's sake, they also save the old file with the extension **.OLD**. For example, new population members might be added to the Example Master File **MASTER.XMP** with the programs ALTMASTR and SETMASTR. ALTMASTR produces **UPDATE.XMP**, which is used in turn as input to SETMASTR. SETMASTR generates an updated version of **MASTER.XMP**, and also saves the file as it was before new population members were added under the name **MASTER.OLD**. Likewise, **UPDATE.XMP** is automatically renamed **UPDATE.OLD**.

9. The Directory File

A PEDSYS Master File and its associated Data Files are normally contained in an independent file directory (see Chapter II, SETUP and INSTALLATION). Information about the files that PEDSYS programs are to process is provided in a file that is associated with such a directory. This file contains the name of one or more Master Files, the numbers and names of linked Data Files, access privileges, and variables that control the way in which program ALTERDAT displays data to be edited.

Directory Files may be created with the PEDSYS program DBFILER, or with any ASCII editor. The Directory File for any directory must be named **DBFILES**.

An example of a Directory File for the Example database is shown below:

```
  1 Master Files
XMP          Example Master File       EXAMPLE
  3 Linked Data Files
XMP QUANTGEN Markers + Quant. P'types   1 F T
XMP SAMPLES  Blood Sample Inventory     2 T T
XMP HEMOGLOB Hemoglobin Levels          3 T T
  4 Unlinked Data Files
EXAMPED.XMP
PEDTRTEST.XMP
TRANSTEST.XMP
WEIGHTS.XMP
  1 Sub-directories (1999/02/15)
LOCUS
```

Records in this file are formatted as follows:

Record 1. Master File Count Record (a)

```
Key    (b)                                  (c)
Map    •••oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
       |  |
Start  1  4
Format  (I3)
```

Record 2. Master File Descriptor Record(s) (d)

```
Key    (e)                    (f)              (g)     (h) (h) (h) (h)
Map    •••oooooooooo•••••••••••••••••••••••••o••••••••••o••••••••••••••
       |           |                         |          |   |   |   |
Start  1           14                        40         52  56  60  64
Format  (A3,10X,A25,1X,A11,1X,4A4)
```

Record 3. Linked Data File Count Record (i)

```
Key    (j)                              (c)
Map    •••ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
       |  |
Start  1  4
Format  (I3)
```

Records 4 - 6 Linked Data File Descriptor Record(s) (k)

```
Key    (e)   (l)             (m)           (n)(o,p)     (q) (q) (q) (q)
Map    •••o••••••••o••••••••••••••••••••••••o••o•o•ooooo•••••••••••••••
       |   |        |                        |  | |      |   |   |   |
Start  1   5       14                       40 43 45     52  56  60  64
Format  (A3,1X,A8,1X,A25,1X,I2,1X,2(A1,1X),5X,4A4)
```

Record 7. Unlinked Data File Count Record (r)

```
Key    (j)                              (c)
Map    •••ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
       |  |
Start  1  4
Format  (I3)
```

Record 8. Unlinked Data File Descriptor Record (s)

```
Key                                (t)
Map    •••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••ooo
       |                                                               |
Start  1                                                              65
Format  (A64)
```

Where • represents a character in a field read by PEDSYS programs.
o represents a single blank ignored by PEDSYS programs, often used as a visual separator.
| is a visual marker pointing to the first character in each field. Numbers below the vertical markers represent the starting column of each field.

- **Notes on DBFILES Record Formats:**

(a) The Master File Count Record is always the first record in **DBFILES**. Maximum length of this (and all records in **DBFILES**) is 67 characters.

(b) The first field processed in this record is an integer (in columns 1-3) indicating the number of Master Files included in the Data Directory (see Chapter II, SETUP and INSTALLATION). The maximum number of Master Files per directory is 5, but in most cases we have found it advantageous to keep a single Master File and its associated Data Files in a separate directory (or subdirectory).

(c) It is often useful to include a note here reminding the user of the database name, date of creation, etc. The note will not interfere with subsequent values since everything in this field is ignored.

(d) There must be a separate Master File Descriptor Record for each Master File specified in the Master File Count Record. Total length is 67 characters.

(e) This field (columns 1-3) contains the file extension identifying the population. Entries must be left-justified if less than three characters in length. In the example above, **XMP** identifies the Example population.

(f) This field (columns 14-38) contains a file label that identifies the Master File for screen and print displays. **EXAMPLE MASTER FILE** is shown in the example above.

(g) A population label is included here (columns 40-50), mostly as a means of determining the appropriate terminology for screen displays and other output. The name **Example** is shown in the example.

(h) Normally, users have read-only access to all PEDSYS files. Four four-character strings may be entered here (columns 52-55, 56-59, 60-63 and 64-67) to serve as passwords of individuals who are given write access, that is, who are authorized to make changes to the Master File using program ALTMASTR.

(i) The Linked Data File Count Record specifies the total number of Linked Data Files per Master File included in the Data Directory. Maximum length is 67 characters.

(j) The only field processed in this record is an integer (in columns 1-3) indicating the number of (Linked or Unlinked) Data Files in the current directory. The maximum number of Linked Data Files per Master File is 50.

(k) There must be a separate Linked Data File Descriptor Record for each Linked Data File specified in the Linked Data File Count Record. Total length is 67 characters.

(l) This field (columns 5-12) contains a Data File name without its extension (**MARKERS** and **SAMPLES** in the example). The extension described in (e) above is appended to this name when the files are accessed by PEDSYS programs.

(m) This field (columns 14-38) contains a label that identifies the Data File for screen and print displays. The **Example Marker Phenotypes** and **Blood Inventory** files are given in the example above.

(n) The number contained in this field (columns 40-41) refers to the position of the data pointer in Data Pointer File records, not counting Ego's Sequential ID which always appears in the first position. See **6. The Data Pointer File** above.

(o) This field (column 43) contains a logical variable specifying whether the Data File is a Multiple-entry file having internal pointers (**T** = yes, **F** = no).

(p) This field (column 45) contains a logical variable specifying the default status of the Log File generated by program ALTERDAT (**T** = Log normally on, **F** = Log normally off). This default may be overridden when running ALTERDAT.

(q) Four four-character strings may be entered here (columns 52-55, 56-59, 60-63 and 64-67) to serve as passwords of individuals who are given write access, that is, who are authorized to make changes to the Data File using program ALTERDAT.

(s) There must be a separate Unlinked Data File Descriptor Record for each unlinked Data File specified in the Unlinked Data File Count Record. Unlinked Data Files need not include an extension, but they cannot be accessed by PEDSYS programs unless they have an accompanying Code File in the Data Directory named **<filename>.cde,** where <filename> is the same as the name of the Data File, ignoring any extension.

(t) Maximum length of the Descriptor Record is 64 characters so that the directory path may be included as part of the file name.

10. The Shell Menu Files

a. **Sun Microsystems Unix**

Program PEDSYS reads four pairs of text files corresponding to the function groups described in Chapter .

General Data Management Programs	**menu.A**	**prog.A**
File Management Programs	**menu.B**	**prog.B**
Pedigree Management Programs	**menu.C**	**prog.C**
Genetic Data Management Programs	**menu.D**	**prog.D**

The program displays the contents of a **menu** file which contains a list of program names and descriptions, and depending on the function selected from the keyboard, calls the matching PEDSYS program listed in the accompanying **prog** file.

b. **MS-DOS**

Batch file PEDSYS.BAT calls program PEDSYSP.EXE. This program displays the contents of a file **PEDMENU** which contains a list of program names and descriptions. Depending on the function selected from the keyboard, PEDSYSP.EXE calls the appropriate PEDSYS program.

c. **Macintosh OS**

PEDSYS program names are kept in a file named **PEDSYS.RSRC** on the Macintosh. These names are displayed in the program shell menu, and are also used to call the appropriate PEDSYS program.

Although any of these files may be modified with the appropriate editor, there should be no need to do so.

11. The Device Control File

To assure that PEDSYS listings are properly formatted, PEDSYS programs LIST, REPORT and TALLY control printer characteristics (such as line length and character size) with codes read from a

Device Control File named **DEVICES** located in the PEDSYS Program Directory. Codes are specified by the manufacturer of the printer. Here is an example of **DEVICES** configured for the Epson 286e printer attached to an MS-DOS computer. Letters in parentheses on the right-hand side of the figure refer to notes that follow, and do not actually appear in the file:

```
 2  Number of devices                          (a)
PRN:    4 Epson 286e                           (b)
136 050 <018> <027> "P"                        (c)
163 050 <018> <027> "M"                        (c)
233 050 <015> <027> "P"                        (c)
272 050 <015> <027> "M"                        (c)
<012> <027> "@"                                (d)
CON:    1 Console screen                       (b)
080 024 <000>                                  (c)
<000>                                          (c)
```

Notes:

(a) The first record in file **DEVICES** (I2 format) specifies the number of output devices to which program LIST may send printed output. In this example, two devices have been specified. Columns 3 through 80 of this record are ignored by program LIST, and it is useful to use this space to define the purpose of the record, as has been done in this example.

(b) These records begin with a device name which, in the case of MS-DOS, must be one of the Reserved Names (CON:, PRN:, LPT1:, LPT2:, LPT3:, AUX:, COM1:, COM2:, or NUL:). This is followed by an integer specifying the number of character fonts available to the device (maximum nine). Format of the record is (A7,1X,I1,1X,A30). *The console screen should be the last device listed in* **DEVICES**.

In this example, PRN: (the first parallel printer) with four fonts, and CON: (the console screen) with one font, have been selected. Columns 11 through 40 of this record are used as entries in the device control menu displayed by programs LIST, REPORT and TALLY.

(c) Following each record specifying a device name comes one or more records (one for each font) that control page format and font setup. This record consists first of two integers specifying the maximum number of characters in the printed line and the maximum number of records per page, respectively. These numbers are followed by a control string (72 characters maximum) that can be used to initialize the printer and font. Elements of this string are printable ASCII characters enclosed in double quotes, and decimal equivalents of ASCII characters represented by a three-digit integer surrounded by angle brackets <>. The actual content of the control string is specified by the manufacturer of the printer. A single blank must be left between elements. Format of these records is 2(I3,1X),A72.

In this example, the first control record specifies a 136-character line (the maximum number of pica-font characters that will fit on standard 14-inch computer paper), and a limit of 50 records per page. Although it may be desirable at times to fit a few more lines on a standard page, we have found the 50-record limit to be convenient. Next follows ASCII character 18 (DC2 or CTRL-R) which is the Epson control code for standard uncompressed print, and then the sequence ASCII character 27 (ESC) and the letter P, which together specify the Pica font. The fourth control record specifies a line length of 272 characters, a page length of 50, ASCII character 15 (SI or CTRL-O) which sets compressed print, and the sequence ESC M which specifies the elite font, etc. Line length 80 and 24 records are specified for device CON:, but ASCII character 0 signifies that no font control is required. *Note: The console screen should be the last device listed in* **DEVICES**.

(d) The record following the last control record for a given device contains control codes that LIST sends to the printer after a PEDSYS listing is finished. This sequence resets the printer to the configuration it had before LIST was used. Rules for composition of the reset string are the same as for the control string described above. Record format is A80.

In this example, a carriage return character <012> is sent to the printer, followed by the Epson 286e reset sequence ESC @. ASCII character 0 signifies that no reset sequence is required for the console screen.

Control codes for more advanced printers can be quite complex. Here is an example of a Device Control File set up for an OTC TriMax 850XL dot matrix line printer, a Hewlett-Packard LaserJet III, a Sun LaserWriter PostScript printer, and the console screen, all attached to a Sun Microsystems computer:

```
13
5 PGenLab-LJ5                     Main Lab LaserJet 5
091 064 "/pedsys/bin/nenscript  -R -B -L64"
135 054 "/pedsys/bin/nenscript  -1r -B -fCourier9 -L54"
151 060 "/pedsys/bin/nenscript  -1r -B -fCourier8 -L60"
174 074 "/pedsys/bin/nenscript  -1r -B -fCourier7 -L74"
204 080 "/pedsys/bin/nenscript  -1r -B -fCourier6 -L80"
<000>
4 PGenLab-LP                      Main Lab Line Printer
136 050 <018> <027> <080>
162 050 <018> <027> <077>
226 050 <015> <027> <080>
247 050 <015> <027> <077>
<018>
1 CON:                            Console Screen
000 000 <000>
<000>
4 PGenLab-Fuj                     Main Lab Line Printer
5 PGenLab-LW                      Main Lab LaserWriterII
078 050 <027> "&l0O" <027> "(8U" <027> "&a2L" <027> "(s0p10h12v0s0b3T"
170 050 <027> "&l1o8D" <027> "(8U" <027> "&a2L" <027> "(s0p16.6h8.5v0s-1b0T"
<012>
```

```
 4  Number of Devices
lp1     4 OTC Line Printer
136 050 <018> <027> <080>
162 050 <018> <027> <077>
226 050 <015> <027> <080>
247 050 <015> <027> <077>
<018>
HP1     2 HP Laserjet III
080 050 <027> "&l0O" <027> "(8U" <027> "&a2L" <027> "(s0p10h12v0s0b3T"
170 050 <027> "&l1o8D" <027> "(8U" <027> "&a2L" <027> "(s0p16.6h8.5v0s-1b0T"
<012>
lw1     4 Sun LaserWriter
080 064 "/usr/local/bin/nenscript  -R -B -L64"
135 054 "/usr/local/bin/nenscript  -1r -B -fCourier9 -L54"
151 060 "/usr/local/bin/nenscript  -1r -B -fCourier-Bold8 -L60"
```

```
166 064 "/usr/local/bin/nenscript  -1r -B -fCourier-Bold7 -L64"
<000>
CON:    1 Console Screen
080 014 <000>
<000>
```

Note that the line printer and console codes are similar to those in the MS-DOS Epson configuration shown above, but those for the HP LaserJet and the Sun LaserWriter are quite different. The LaserJet uses HP Printer Control Language sequences, which can be found in the User's Manual accompanying any Hewlett-Packard printer. The LaserWriter uses PostScript control sequences which are provided by **nenscript**, a public domain Postscript translation utility described in Appendix D.

The Device Control File provided for the Macintosh computer appears as follows:

```
2
MAC     1 Macintosh printer
110  60
<000>
CON:    1 Console
 0    0 <000>
<000>
```

File **DEVICES**, must be present in the PEDSYS Program Directory (see Chapter II, SETUP and INSTALLATION). The file should be created with any text editor at the time PEDSYS is installed.

12. The Standard Mnemonics File

A permanent input file containing standardized mnemonics and field lengths named **CODE.STD** must be present in the PEDSYS Program Directory. The first two columns of this file contain invariant mnemonics appropriate for non-human (animal), and human pedigrees, respectively. The last two columns contain the lower and upper range of field lengths of required Master File items, respectively. The contents of this file, which is shown below, should not be altered in any way.

```
   SEQ                  SEQ                  5  5
   SSEQ                 FSEQ                 5  5
   DSEQ                 MSEQ                 5  5
   KID1                 KID1                 5  5
   PSIB                 PSIB                 5  5
   MSIB                 MSIB                 5  5
   FSIB                 FSIB                 5  5
   NKID                 NKID                 3  3
   PSIBSZ               PSIBSZ               3  3
   MSIBSZ               MSIBSZ               3  3
   FSIBSZ               FSIBSZ               3  3
   ID                   EGO                  3 32
   BIRTH                BIR                  3  8
   SIRE                 FA                   3 32
   DAM                  MO                   3 32
```

```
SEX                         SEX                          1  1
EXIT                        EXIT                         3  8
```

CODE.STD is required input for programs ALTERDAT, BROWSE, ALTMASTR, COUNTPED, DATES, DELRECS, EXCLUDE, INDEX, MERGE, NUCFAM, REPORT, SETMASTR, SHOW, SHOWDATA, SIBSHIP and VERIFIER.

13. The Genotype-Phenotype Mapping File

Relationships between genotypes and the phenotypes they express are defined in a file named **GENOMAP.<ext>**. This mapping is required for X-linked loci and for autosomal loci exhibiting dominance. It is optional for autosomal codominant phenotypes whose nomenclature adheres to PEDSYS conventions listed below. This file is used by programs GENCHECK, GENEFREQ, and INFER, and is normally created with program GENOMAP. An example showing entries for the ABO blood group is given below:

```
ABO    PHENO  TYPE           3   4   6 AUDR
A      B      0

                | | |A|
              A|B|0|B|
A      A      |+| | | |
A      B      | | | |+|
A      0      |+| | | |
B      B      | |+| | |
B      0      | |+| | |
0      0      | | |+| |
```

Records in this file are formatted as follows:

Record 1. Locus Descriptor Record (a)

```
Key                 (b)              (c) (d) (e) (f)
Map    ••••••o••••••o••••••o••••••o•••o•••o•••o••••
       |      |      |      |      |   |   |   |
Start  1      8      15    22     29  33  37  41
Format   (4(A6,1X),3(I3,1X),A4)
```

Record 2. Allele Descriptor Record (g)

```
Key    (h) ...
Map    •••••o•••••o•••••o•••••o ...
       |     |     |     |
Start  1     7    13    19 ...
Format   (<na>(A5,1X))
```

Records 4 - 5. Phenotype Label Records (i)

```
Key                 (j) ...
Map    ooooooooooooo•|•|•|• ...
                    |
Start               14
Format   (13X,<np>(A1,'|'))
```

Records 6 - <nc>. Genotype-Phenotype Mapping Record (k)

```
Key      (l)         (m)
Map    •••••o•••••o|•|•|•| ... |
       |     |     |
Start  1     7    13
Format   (2(A5,1x),<np>('|',A1), ... ,'|')
```

Where • represents a character in a field read by PEDSYS programs.
o represents a single blank ignored by PEDSYS programs, often used as a visual separator.
| is a visual marker pointing to the first character in each field (also appears between single quotes as a data constant in records 4 and higher).

Numbers below the vertical markers in the diagrams above represent the starting column of each field.

- **Notes on GENOMAP.<ext> Record Formats:**

(a) A Locus Descriptor Record is always the first record in the group of records defining the relationships of phenotypes and genotypes for a given locus. Length of this record is fixed at 41 characters.

(b) The first four fields of this record contain a unique locus name which must be the same as the first four mnemonic words describing the locus in the Code File associated with the Data File containing phenotypic data.

(c) The fifth field contains an integer n_a enumerating the alleles found at this locus.

(d) The sixth field consists of an integer n_p enumerating the phenotypes expressed at this locus.

(e) The seventh field consists of an integer n_c enumerating all possible genotypes.

(f) The last field contains a four-character alphabetic code specifying mode of inheritance. These codes are:

```
AUCD     Autosomal co-dominant
AUDR     Autosomal dominant/recessive
XLCD     X-linked co-dominant
XLDR     X-linked dominant/recessive
YL       Y-linked
```

(g) The Allele Descriptor Record contains a list of the n_a alleles at this locus. Its length is $6n$.

(h) Strings representing alleles may be up to five printable characters in length. Alphabetic symbols are case-sensitive.

The third record is blank.

(i) The fourth and up to the 13th records carry labels for each of the n_p phenotypes. Length of these records is $2n_p + 13$. There may be up to 10 such records in the file, depending on the number of characters in the string making up the longest phenotype label.

(j) Characters in the strings making up phenotype labels are arranged vertically, with each single character followed by a stroke (|) to provide visual separation between labels.

(k) A Genotype-Phenotype Mapping Record is repeated for each of the n_c unique combinations of allele pairs (disregarding parental origin).

$n_c = n_a (n_a + 1)/2$ for autosomal loci, $n_c = n_a (n_a + 3)/2$ for X-linked loci.

(l) The first two fields of this record consist of a pair of allele labels from the list described in (g) above.

(m) One of the next n_p fields contains a **+** if the phenotype found in the corresponding position in the Phenotype Label Record is produced by the allele pair. Remaining fields contain a blank.

- Loci having any mode of inheritance may be listed in the Genotype-Phenotype Map File, but inclusion is required only if they are phenotypes that are X-linked and/or exhibit dominance. Inclusion of autosomal codominant loci is necessary only if a distinction is to be made between allele and phenotype symbols (lower vs. upper case, for example).

14. Temporary System Files

Several PEDSYS programs create one or more files with the extension **.tmp** (or uppercase **.TMP**) which are used to save data during program execution, and which are erased upon successful completion of the task. The **.tmp** extension is reserved for PEDSYS system use and should not be used in naming Master Files or Data Files. In addition, some PEDSYS programs that direct their output to disk files use **.tmp** files as input to programs run subsequently. These files contain information that helps keep track of prior manipulations performed on a data set. The files are

LASTFILE.TMP - Which consists of two records containing the name of the output file most recently produced, and its associated Code File. This file is produced by PEDSYS programs that produce Data Output Files (with extension **.out**). The file name read from **LASTFILE.TMP** is automatically included in the opening screen display when the user is asked to choose a file for processing.

SORTIN.TMP - Which is produced by program SORT, consists of a variable number of records that (a) identify the information contained in the file, (b) list the items sorted in order of significance, and (c) give the name of the file that has most recently been processed by the SORT program. This information is automatically printed on the cover page of listings produced by the program LIST. Here is an example of **SORTIN.TMP**

```
SORTINFO
 Records sorted in following order (major sort first):
```

```
 VARIABLE  16   (SEX   )   SEX
 VARIABLE  12   (ID    )   ANIMAL ID

0/pedsys/example/MASTER.XMP
```

The **0** preceding the file name informs LIST that this is the only information about processing history that is to be printed on the cover page.

SUBIN.TMP - Produced by program SUBSET, consists of a variable number of records that (a) identify the information contained in the file, (b) list the criteria by which the subset was made, and (c) give the name of the file that has most recently been processed by the SUBSET program. This information is automatically printed on the cover page of listings produced by the program LIST. Here is an example of **SUBIN.TMP**

```
SUBINFO
 Subset constructed with the following commands:

 INCLUDE SEX EQ  <1>

0/pedsys/example/MASTER.XMP
```

To better illustrate the function of the number preceding the file name, here is an example of **SORTIN.TMP** that is produced after the file **MASTER.XMP** has been processed by program SUBSET immediately prior to being sorted:

```
SORTINFO
 Records sorted in following order (major sort first):

 VARIABLE  12   (SEX   )   SEX
 VARIABLE   8   (ID    )   ANIMAL ID

SUBINFO
 Subset constructed with the following commands:

 INCLUDE SEX EQ  <1>

1/pedsys/example/MASTER.XMP
```

SORTIN.TMP now includes information from **SUBIN.TMP**. The 1 preceding the file name informs program LIST that an expanded processing history of **MASTER.XMP** is to be printed on the cover page and page headings.

Occasionally when a program is interrupted or ends prematurely, one or more temporary system files will not be erased. Any **.tmp** files remaining in the working directory may be deleted manually, although this is usually unnecessary. No attention need be paid to these files, other than to understand their effects. They are produced anew each time a program is run, and are read automatically.

15. The Data Path File

Unless otherwise directed, PEDSYS programs read data from, and send their output to the current (open) Data Directory. To change this default, a short optional file containing disk drive and directory specifications may be placed in each working directory (Unix and MS-DOS) or Data Directory (Macintosh). PEDSYS programs use the routings specified in this file (if it is present) rather than the default. The file must be named **PATHWAY**, and may contain one or two records defining input and output data paths, as needed. This file is best created with program **setpath**, although with care, any text editor may be used.

For example, a Data Path File containing the following records:

```
/pedsys/example/
/home/users/bdyke/
```

would direct PEDSYS programs to read input from the Example Data Directory , and send output files to `/home/users/bdyke/.`

Note: Temporary System File **LASTFILE.TMP** *is always directed to the current Data Directory, and is not affected by the output path specified in the Data Path File.*

References:

Dyke B 1993 Basic data standards for primate colonies. *American Journal of Primatology* 29:125-143.

Dyke B, Mamelka PM 1989 *ACMP, an Animal Colony Management Package User's Guide.* Population Genetics Laboratory Technical Report No. 3, Southwest Foundation for Biomedical Research, San Antonio, TX 78228

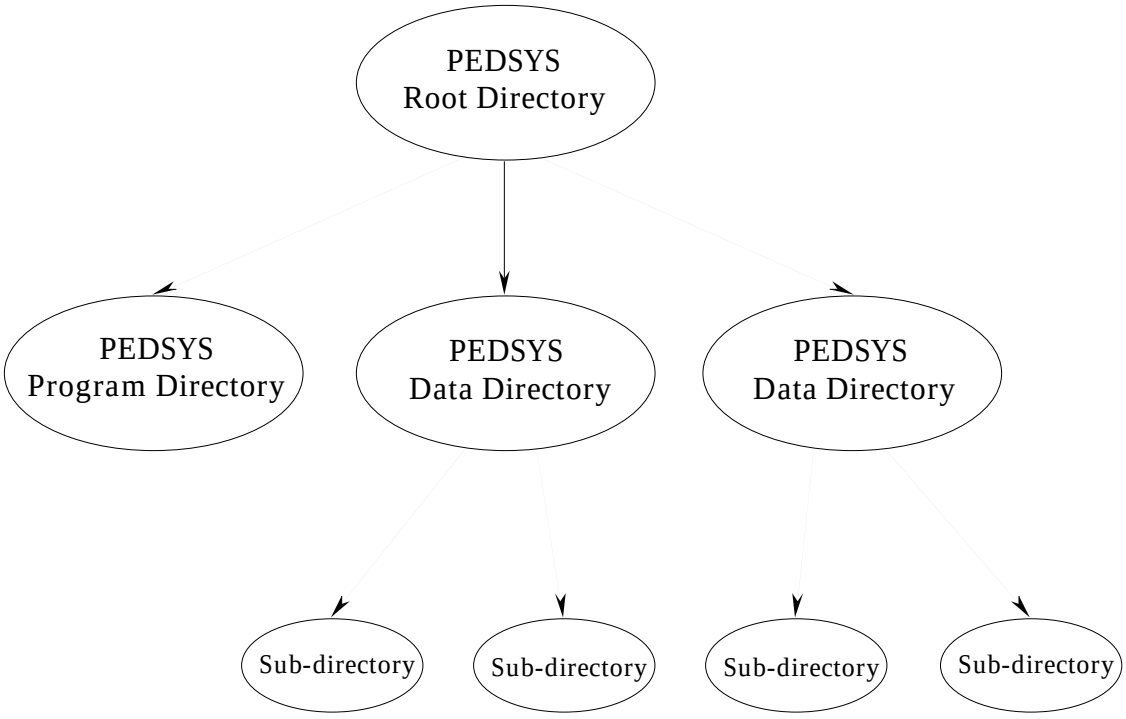
Appendix E. INSTALLATION and SETUP

Sun Microsystems Unix (Solaris 2.5)

In this appendix, we describe directory organization, installation procedures, and file access and permissions for Sun Microsystems Solaris 2.5 (compatible with later versions of this operating system).

1. Directory Organization

The basic scheme assigns executable program files to an independent Program Directory, which is kept separate from any number of Data Directories, which in turn may contain one or more sub-directories. These directories are set at the same hierarchical level below a PEDSYS root directory, as shown in the following diagram:



a. The PEDSYS root directory

Typical contents of the Unix PEDSYS root directory are as follows:

```
./          CODE.STD          DEVICES          example/
../         DBDIRECT          bin/             menu/
```

where **CODE.STD** and **DEVICES** are the Standard Mnemonics File and Device Control File, respectively (see Appendix A), **DBDIRECT** is a file used as input to program **setpath** containing a list of pathways to the various Data Directories (see Chapter III), **bin/** is the name of the Program Directory, and **menu/** is the name of a directory containing text files used for display of program functions and file access. The only Data Directory shown is **example/**, which reflects the state of the root directory at

the time of its installation.

b. The PEDSYS Program Directory (bin/)

Files typical of the Program Directory are shown in the display below (this list may change as new programs are added or removed in the course of PEDSYS development):

```
../             countped        genefreq        nucfam          show
./              datalink        genomap         pedsys          showdata
age             dates           index           pedtrim         sort
alterdat        dbfiler         infer           peelseq         subset
altmastr        delrecs         itemize         prepdraw        tally
ancestor        downcode        kinship         reformat        translat
append          duplic          list            report          triplets
browse          exclude         makeped         segcheck        verifier
calc            foundrep        merge           setdata
code            fprint          nenscript       setmastr
combine         fprnen          newitem         setpath
```

Files **fprint** and **nenscript** generate Unix printer commands from program output. File **setpath** contains a program that controls file access pathways, and is described in Chapter III, *Setting Directories and Running Pedsys Programs.*

c. PEDSYS Data Directories

PEDSYS Master Files and Data Files are kept separately from PEDSYS programs in their own independent Data Directories. Usually (although not necessarily) only one Master File is kept in a Data Directory. The **example** Data Directory contains the following entries:

```
./              EXAMPED.CDE     LOCUS/          QUANTGEN.CDE    TRANSTEST.XMP
../             EXAMPED.XMP     MASTER.XMP      QUANTGEN.XMP    WEIGHTS.CDE
CODE.XMP        GENOMAP.XMP     MPOINT.XMP      SAMPLES.CDE     WEIGHTS.XMP
DBFILES         HEMOGLOB.CDE    PEDTRTEST.CDE   SAMPLES.XMP     examped.tab
DPOINT.XMP      HEMOGLOB.XMP    PEDTRTEST.XMP   TRANSTEST.CDE
```

A copy of the PEDSYS Directory File **DBFILES** is also included in each Data Directory. DBFILES is created by program DBFILER (see Chapter IV, *Creating a PEDSYS Database,* and Appendix A, *File Types*). Also listed in the Data Directory may be one or more Subdirectories. Included here is the LOCUS subdirectory:

```
./              AllLoci.cde     D1S515          D4S1582.cde     DYS391.cde
../             CA1             D1S515.cde      D5S1466         ErrorDemo
ABO             CA1.cde         D2S1329         D5S1466.cde     ErrorDemo.cde
ABO.cde         CandLoci        D2S1329.cde     D6S1036         G6PD
ADA             CandLoci.cde    D2S326          D6S1036.cde     G6PD.cde
ADA.cde         D1S1656         D2S326.cde      D7S23           GGAT1A4
APOBpvu2a       D1S1656.cde     D3S1229         D7S23.cde       GGAT1A4.cde
APOBpvu2a.cde   D1S3721         D3S1229.cde     DBFILES
AllLoci         D1S3721.cde     D4S1582         DYS391
```

Note that names of Master Files, Linked Data Files, and System Management Files are in upper case to distinguish them from less permanent files that may be generated by PEDSYS programs. We have found it best to limit files kept in PEDSYS Data Directories to those used as more or less stable input to PEDSYS programs, and to keep them separate from working areas where output files accumulate. This structure is enabled by the installation procedures given below.

d. The PEDSYS Menu Directory

The Menu Directory contains input data for the PEDSYS program menu seen when running program PEDSYS.

```
 ./      menu.A      menu.C      prog.A      prog.C
../      menu.B      menu.D      prog.B      prog.D
```

2. Installation

The Unix version of PEDSYS is provided on six compressed tar distribution files named **PEDSYS<n>-2.0.tar.gz**, where <n> is an integer from 1 - 6 identifying each of the files. These files are available on CD-ROM (ISO 9660 format), and from our FTP site (ftp://www.sfbr.org/pub/pedmgt) or the Southwest Foundation Web Page (http://www.sfbr.org). Installation proceeds as follows (note that write permission is assumed for the directory where PEDSYS is to be installed):

Step 1. Create a PEDSYS root directory **`/pedsys`**, and within this directory, create sub-directories **`bin`**, **`menu`** and **`example`**. Then within the **`example`** sub-directory, create the **`LOCUS`** subdirectory.

Step 2. Copy or download the compressed PEDSYS tar files to the location where the PEDSYS root is to reside.

Step 3. Uncompress each of the five file with the command:

```
gunzip PEDSYS<n>-2.0.tar.Z
```

Step 4. Extract the PEDSYS applications and data files from each of the the now uncompressed distribution files with the command(s):

```
tar xvf PEDSYS<n>-2.0.tar
```

This will automatically place files in their appropriate directories and sub-directories.

Step 5. Set the PEDSYS environment variable by adding the following line to the .cshrc initialization file:

```
setenv PEDSYS ~pedsys_root
```

where `~pedsys_root` is the complete path name for the pedsys root directory (assumes use of the Unix **csh** shell)

Step 6. Set the path to include the ~pedsys_root/bin directory in the .cshrc file.

```
set path=($path ~pedsys_root/bin)
```

This makes it possible to run PEDSYS from any working directory. To see the program menu, enter the command 'pedsys'. The first time this command is issued, program **setpath** is run (see below), from which the Example directory should be selected for input. The PEDSYS program menu is then displayed.

- **Note:** Executable program files currently require over 85Mb of disk storage on the Sun SPARC-Station, exclusive of data requirements.

3. Unix File Access and Permissions

Since Unix is a multi-user operating system, issues of access and security are complex, and require careful implementation. In general, the strategy is to control access and security by assigning standard Unix file permissions to the various files and directories. File permissions are denoted by a 10-character Permissions Field that appears at the beginning of each file descriptor record. This field can be seen by displaying the contents of a directory in long format (that is, by entering Unix command **`ls -l`**).

The first character of the field will be either the letter **`d`**, indicating that the descriptor record applies to a directory, or a dash **`-`**, indicating that record describes a file. The next nine characters fall into three sets, the first of which specifies permissions given to the **owner** of the file or directory. The second and third sets specify permissions for a **group** (of users), and the "**world**" (all other users), respectively. Permissions within each set are ordered as follows:

	Given	Denied
read	**r**	**-**
write	**w**	**-**
execute	**x**	**-**

Ownership, group membership and permissions are usually assigned by the Unix Systems Manager. Examples of typical Unix directory entries are shown below.

a. Program Directories

The owner of a PEDSYS Program Directory and its files is usually the individual who has the major responsibility for installing and maintaining these files, and who thus must have unrestricted read, write and execute privileges in the PEDSYS account. Additional selected users may be assigned a group that has some set of privileges, whereas most other users are given only read and execute access. A typical entry for a Program Directory might be

```
drwxrwxr-x  8 tom      staff        3702 Jan  6 11:19 ./
```

The initial **`d`** indicates that the descriptor line applies to a directory. The characters **`rwx`** are repeated twice, indicating that the owner (**`tom`**) and group (**`staff`**) have full access to this directory. Here, read access refers to permission to list the files in the directory, write access makes it possible to add a file to the directory, while the execute privilege makes it possible to open the directory for access. The last set of three characters (**`r-x`**) specifies that other users may open the directory and list existing files, but may not add new files to it.

A typical Program Directory entry for program ALTERDAT might be

```
-rwxrwxr-x  1 tom      staff     2039088 Jan  6 11:19 alterdat
```

The initial **-** indicates that the descriptor line applies to an ordinary file. The characters **rwx** are repeated twice, indicating that the owner (**tom**) and group (**staff**) have full privileges, while the last three characters (**r-x**) specify that other users may read and execute, but not write the files.

b. Data Directories

The owner and group associated with PEDSYS Master Files and Data Files need not be the same as those for Program Files. A typical Data Directory entry for the Example Master File might be

```
-rw-r--r--   1 bdyke    genetics    3072 Jul 19  1998 MASTER.XMP
```

The initial **-** indicates that the descriptor line applies to an ordinary file. Read and write privileges (**rw-**) are given to the owner (**bdyke**) and group (**genetics**), while only read access is given to other users (**r--**). Because this is a Master File, execute privileges are not needed, and a dash appears in the third position of each set of characters.

Note that PEDSYS Master Files and Data Files also can be assigned simple passwords with program DBFILER. The security of these passwords depends in part on the security of the Directory File **DBFILES** in which they are kept.

c. Data File Locking

PEDSYS uses the Unix utility SCCS (Source Code Control System), or equivalent, for file locking, that is, to keep more than one user with Data Directory write privileges from trying to update a Master File, Data File or Code File at the same time. SCCS protection may be implemented after installation. What follows here is a brief description of the concepts involved.

The SCCS control program uses an SCCS subdirectory (created in each Data Directory) containing a special copy of the files to be protected. Each of these copies starts out with the original data, but accumulates a dated history of all changes made as it is modified over time. These **history** files (as they are called) keep the names of the original files, but prefix them with an **s.** (for example, **MASTER.XMP** becomes **s.MASTER.XMP**). History files are assigned read only access and may be updated only with the SCCS **edit** command, which creates a single editable working copy of the history file. A user with SCCS write privileges may modify this copy (for example, with program ALTERDAT). The history file itself is automatically locked while the working copy is being updated. When changes have been completed, the **s.** version is updated, and the working copy automatically deleted.

The SCCS **get** command is then used to create a non-editable (read only) working copy of the history file. This copy is the one accessed by PEDSYS users who do not have Data File write privileges. In addition to the file locking feature, SCCS also makes it possible to audit past changes made to a file. An excellent short explanation of SCCS has been written by Peter Collinson in the October 1991 issue of *SunExpert* magazine (pp. 34-39).

4. Setting Input/Output Directories

Before PEDSYS can be used, an Input Data Directory must be selected and attached to the programs, and the output directed appropriately. Input and output directories may be set in several ways, but the most convenient is to use program **setpath**, which is started by entering the command **setpath** from any directory, which results in the following display, from which Input and Output Directories may be selected.

At the top of the screen is a list of directories to which input and/or output paths may be changed. This list always includes the **Current Working Directory** and the **Other Unlisted** directory. Additional Input Data Directories are read from the **setpath** directory file **DBDIRECT** (described below). The **example** directory is the only Data Directory that has so far been added to the directory list:

```
 1. Current
 2. Other Unlisted
 3. EXAMPLE

Enter Input Data Directory number (1-26), "O" to change the Output Directory,
"A" to add a new directory, "D" to delete a directory, "H" for help,
or "Q" to quit.

>3




Current Working Directory:  /home/users/bdyke/
         Output Directory:  /home/users/bdyke/
     Input Data Directory:  /home/users/bdyke/
```

At the bottom of the screen are listed paths to three directories:

Current Working Directory is the one that the user is accessing at the moment, that is, the one from which the PEDSYS shell has been invoked. This directory is fixed once PEDSYS programs have been started, and may be changed only with a system command (**cd** in Unix).

Output Directory is the one to which output from PEDSYS programs will be directed. Normally, the output directory will be your current working directory.

Input Data Directory is the directory in which shared PEDSYS Master and Data files for a particular population reside.

In this example, all three directories and the pathways to them are initially the same (directory **bdyke**, and path **/home/users/**).

a. Changing the Input Data Directory

Note that in the display above, a **3** has been entered. This changes the Input Data Directory to

```
/pedsys/example/
```

when the RETURN key is hit, leaving the Current Working Directory and the Output Directory unchanged. Entry of **q** exits the pathway program and returns control to the Current Working Directory level.

b. Adding Directories to DBDIRECT

If you have the appropriate access permission, you may add directories to the **setpath** directory file **DBDIRECT** by entering **a** at the prompt shown in the display above, which produces the following display:

```
 1. Current
 2. Other Unlisted
 3. EXAMPLE

Enter the pathname of the new data directory, or RETURN to exit without
adding a new directory.

>/pedsys/beagles

Enter the species name (1 to 11 characters), or RETURN without
adding a new directory.

>beagles
```

Here, a new path (**pedsys/beagles**) will be be added to file **DBDIRECT**, and the **beagles** directory will be listed in the **setpath** display.

c. Deleting Directories from DBDIRECT

Directories may be deleted from file **DBDIRECT** by entering **d** at the initial **setpath** prompt, which produces the following display (showing the **beagles** directory added above):

```
 1. Current
 2. Other Unlisted
 3. EXAMPLE
 4. BEAGLES

Enter the number of the directory to be deleted, or press RETURN to cancel.

>4
```

Here, a **4** has been entered which will delete the **pedsys/beagles** path from file **DBDIRECT**, and remove the **beagles** name from the **setpath** list of directories.

Notes:

- The operation of **setpath** creates (or modifies) a file named **PATHWAY** in the Current Working Directory. **PATHWAY** contains either one or two records, the first of which is always the path to the Input Data Directory. The second record, if it is present, contains the path to an Output Directory. If a second record is not present, output is automatically directed to the Current Working

Directory. An example of records in a **PATHWAY** file might be:

```
/pedsys/example/
home/users/bdyke/temp/
```

Here, the **example** directory has been selected for input, while output will be sent to the subdirectory named **temp**. Both input and output data bypass the Current Working Directory in this case.

- Program **setpath** modifies the **PATHWAY** file if one is present in the Current Working Directory; otherwise it creates and saves a new copy.

- You may have more than one **PATHWAY** file (each with different contents) on your system. This is because copy of **PATHWAY** is created in any directory used as the Current Working Directory.

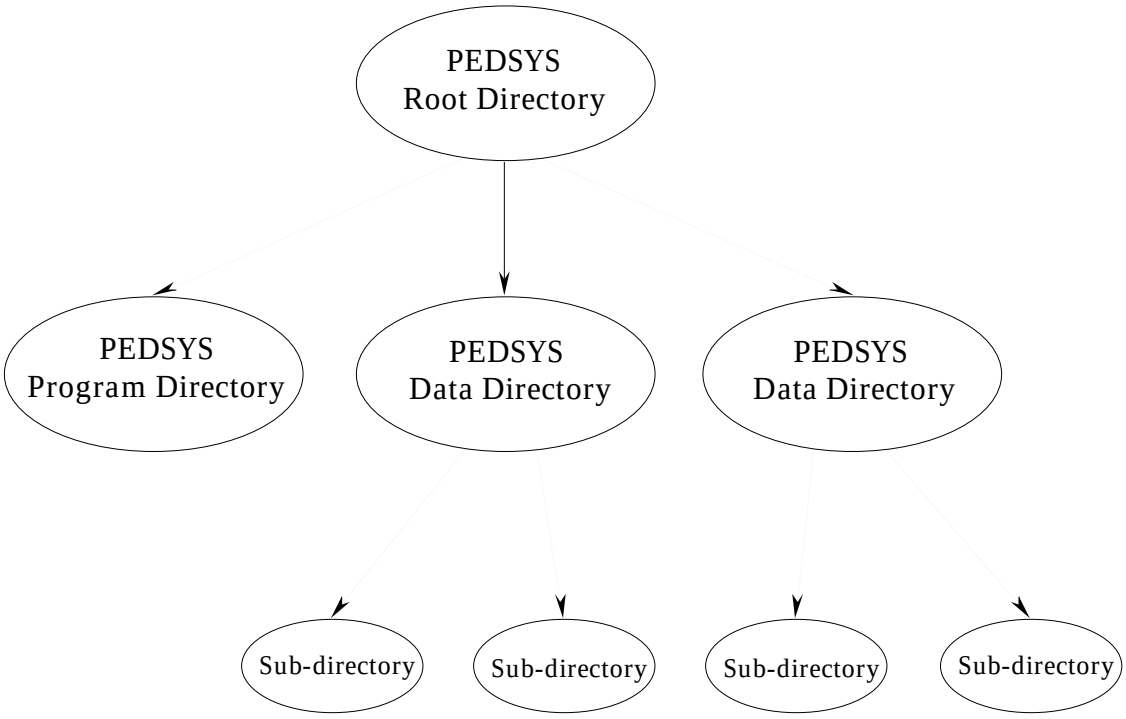
Appendix F. INSTALLATION and SETUP

Apple Macintosh OS

In this appendix, we describe directory organization, installation procedures, and file access and permissions for the Apple Macintosh OS (7.1 - 8.1).

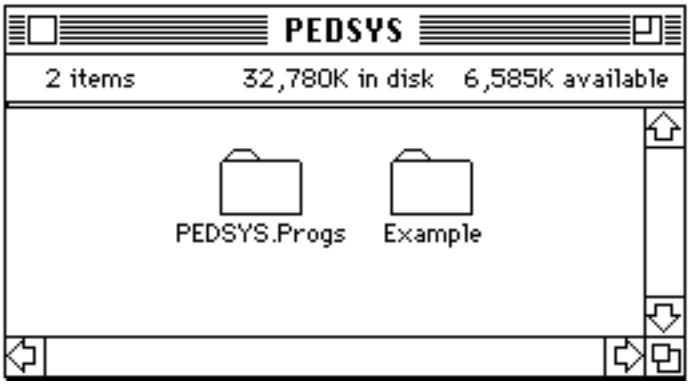
1. Directory Organization

The basic scheme assigns executable program files to an independent Program Directory, which is kept separate from any number of Data Directories, which in turn may contain one or more sub-directories. These directories are set at the same hierarchical level below a PEDSYS root directory, as shown in the following diagram:



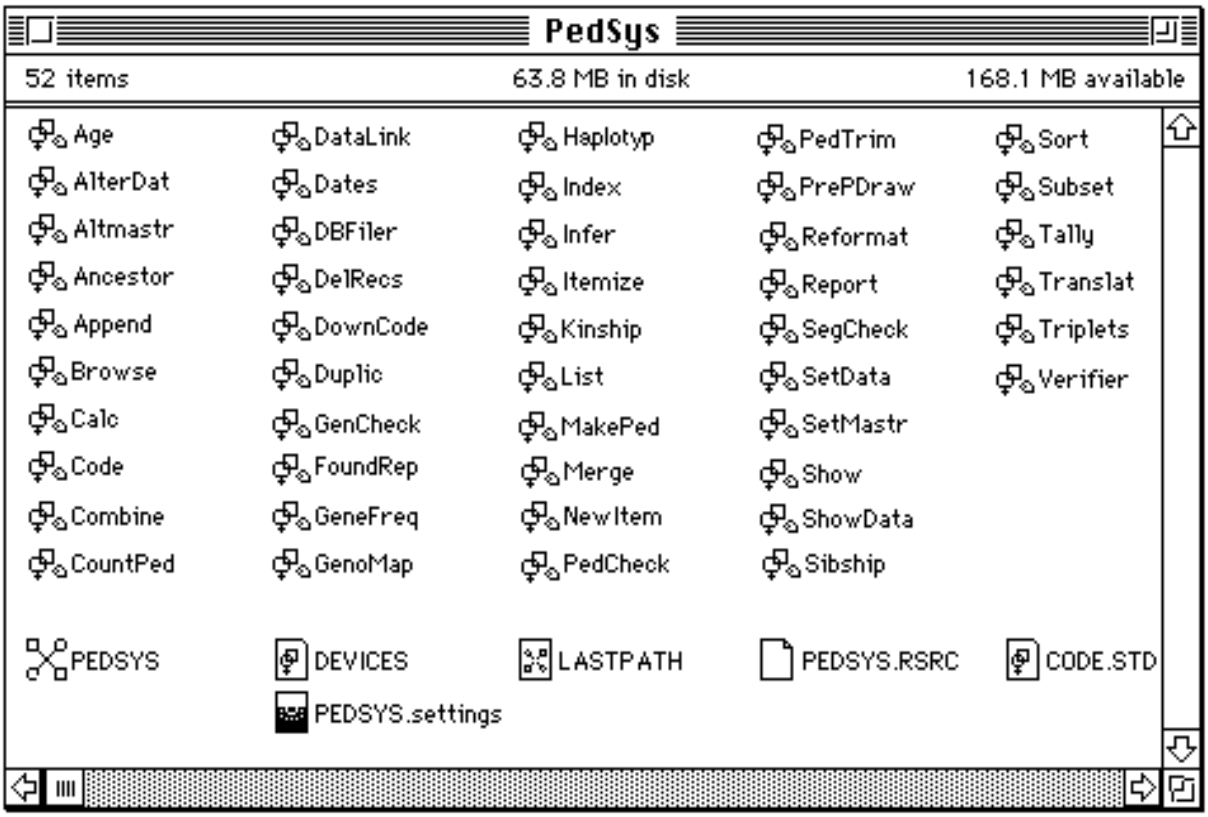
a. The PEDSYS Folder

The Macintosh folder corresponding to the PEDSYS root directory contains only the Program Folder and one or more Data Folders (here the **Example** folder):



b. The PEDSYS Program Folder

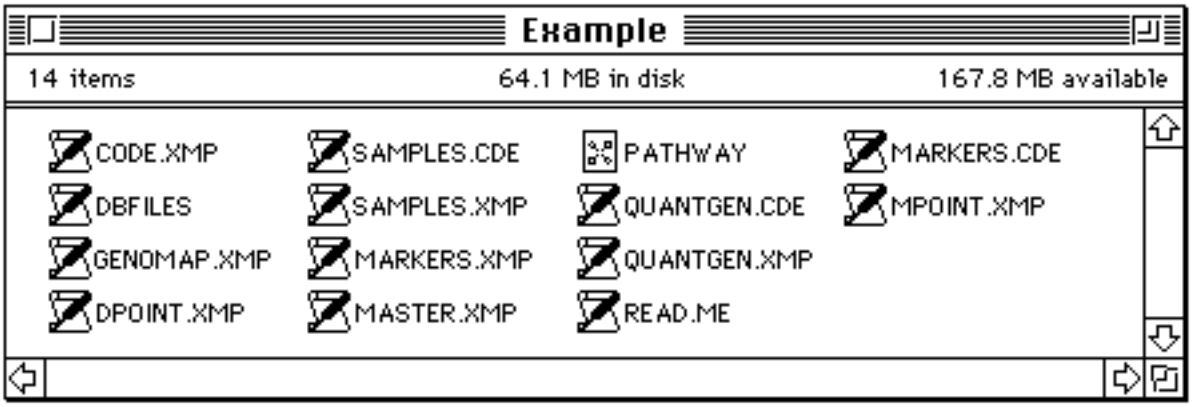
The Macintosh Program Folder contains the following files:



In addition to executable program files, files **CODE.STD** and **DEVICES** are also located in the the Macintosh PEDSYS Program Folder.

c. PEDSYS Data Folders

The **example** Data Folder contains the following files:



d. Minimum System Requirements

- Macintosh OS in the range 7.1 - 8.1.

- Available hard drive space of at least 25 megabytes (MB). PEDSYS applications occupy about 18 MB, and an additional 5-10 MB should be made available for pedigree data files (depending on the size of the files), and for PEDSYS applications to use in allocating temporary work files, some of which may be larger than the source files being processed.

- A PowerPC-based Macintosh, with 16 MB of internal RAM; or a 680x0-based Macintosh with a floating point unit (FPU) and 8 MB of RAM. Virtual Memory may be activated to allow applications requiring more memory to run successfully.

For 680x0-based Macs which do not include a FPU as standard equipment, SoftwareFPU may be used to provide emulation. Version 3.04 has been included on the PedSys.68K release disk; it is offered as shareware by John Neil & Associates for a registration fee of $10. More information is available at *http://www.jna.com*.

The RAM requirements for individual programs are based on a Master file record limit of 30,000 records. This allows most PEDSYS applications to run in a 1-2 MB memory partition, although programs such as *Index* and *Kinship* require a 4-6 MB partition. The Suggested Memory sizes set for each application were determined by experimentation. If a PEDSYS program terminates abruptly when processing large files, or otherwise indicates that too little RAM is available, increase its Preferred Memory allocation under the *Finder* (using **Get Info** command on the **File** menu) and rerun. Use the memory partitions display in the **About this Computer** window as a guide to adjusting the memory required by a particular application.

e. Support Files

- The first two support files are provided in the **PedSys.Progs** folder and should not be deleted or moved:

  CODE.STD defines the standard field mnemonics used to create and verify Code files for a linked Master File record. This file is referenced whenever a Master File is opened by a PEDSYS application.

  DEVICES contains the application display window dimensions set in *PedSys* (as described later in this document), and printer page limits used by the applications *Tally*, *List* and *Report*, all of which generate printed output. Each of these programs enables the **Page Setup** command on the **File** menu, allowing standard printing options such as **Page Orientation** and **Reduction/Enlargement** to be specified when the program is first started, and before each subsequent output file selection. These settings are combined with the selected font size to determine the maximum text line width and number of lines that can be printed on a single page. These limits are stored and referenced the next time *List*, *Tally* or *Report* is run. Default values are Portrait orientation and 100% Reduction/Enlargement.

- The remaining support files are maintained by PEDSYS applications:

  PATHWAY is created or updated by the application *PedSys* when a Data Folder is selected using the **Path IN** button, or when a path to an output folder is set using **Path OUT**. It is stored in the input Data Folder. Double-clicking this file (or its alias) activates the *PedSys* application and sets the I/O paths to the pathnames stored on the clicked **PATHWAY** file.

  LASTPATH contains the input/output pathnames most recently accessed. It is stored in the **PedSys.Progs** folder, and is read by any PEDSYS application when starting up to determine the path names for reading and writing PEDSYS files. Double-clicking this file launches the *PedSys* application, and sets the input/output paths to those specified on **LASTPATH**. If this file is not found, the user is asked to return to *PedSys*, and select an input Data Folder.

PedSys.settings is created and updated by the system to keep track of window placement, font selections, and other system-wide parameters. If this file is deleted, it will be initialized with default values by the next active PEDSYS application. If PEDSYS programs do not start up properly, remove **PedSys.settings** from the **PedSys.Progs** folder, and try again. If the file was updated improperly, or otherwise contains corrupted data, it can interfere with program execution.

2. Installation

PEDSYS for Macintosh can be obtained on CD-ROM (postal delivery), or in self-extracting archive files downloaded from the Southwest Foundation's web site. (Addresses are listed at the end of this document). Installation procedures differ according to how the software is obtained:

- The *PEDSYS CD-ROM* contains editions for Solaris, DOS/Windows and Macintosh, and an Adobe Acrobat-formatted version of the PEDSYS User's Manual. Installation of the Macintosh edition from the CD-ROM consists of locating and opening the **MAC** folder, opening the **PedSys** folder, and dragging the **PedSys.PPC** or **PedSys.68K** folder to your hard drive (depending on whether the target system is a PowerPC- or 680x0-based Macintosh). Each of these folders contains an applications folder, **PedSys.Progs**, and **Example**, a small data folder.

- The *SFBR web site* provides PEDSYS in four Self-Extracting Archive (**.sea**) files for both old and new systems. Archive file names for the PowerMac edition are of the form **PedSysPPC1.sea**, while equivalent archive names for Macintosh 680x0 edition are of the form **PedSys68K1.sea**. The first archive file contains the Data Management applications and example files in a folder called **PedSys**. Open (double-click) the **.sea** file to start the extraction process, and specify a hard drive location where the archive folder **PedSys68K1 Folder** will be installed. When the extraction process is complete for the first archive file, the **PedSys** folder will be present in this archive folder. The **PedSys** folder contains two others: **PedSys.Progs**, which stores applications and support files, and **Example**, a small data set that can be used to test the system. Archive files 2-4 contain the remaining applications, which will also be stored in similarly-named archive folders when extracted. Drag these applications to **PedSys.Progs** to complete the installation. About 18 MB hard drive space is required for the full installation.

  The User Manual is contained in the file **PSManual.pds**. It may be viewed or printed with *Acrobat Reader 2.1/3.0*.

Restrictions

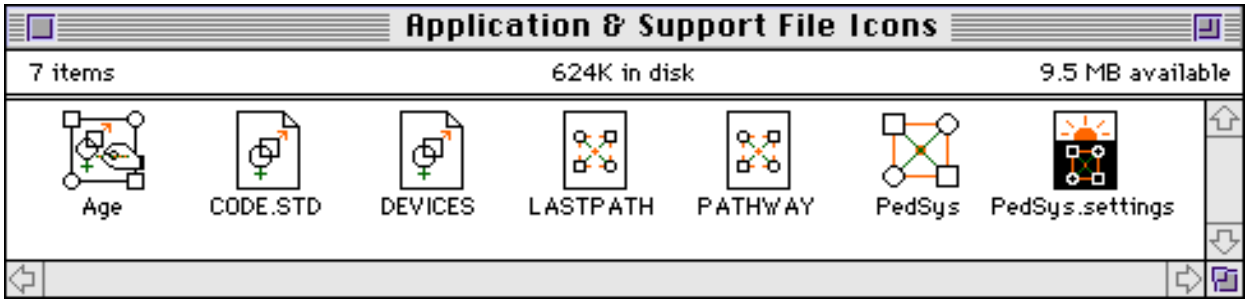
Once installed, the **PedSys** folder may be placed at any level in a hard drive's folder hierarchy, subject to two restrictions:

- The fully-qualified pathname of the folder **PedSys**, formed by joining parent folder names in the form "folder1:folder2:folder3:", must be under 244 characters in length. This pathname length restriction applies to the location of Data Folders as well.

- The fully-qualified pathname of a folder or file accessed by the system must contain no spaces or Mac-only characters. This restriction is required by the standard sort routine used by many PEDSYS applications, which was designed with Unix conventions in mind. Limiting folder names to printable ASCII characters (e.g. 0-9, A-Z, a-z, period, dash and underscore) is recommended.

Enabling Custom Icons

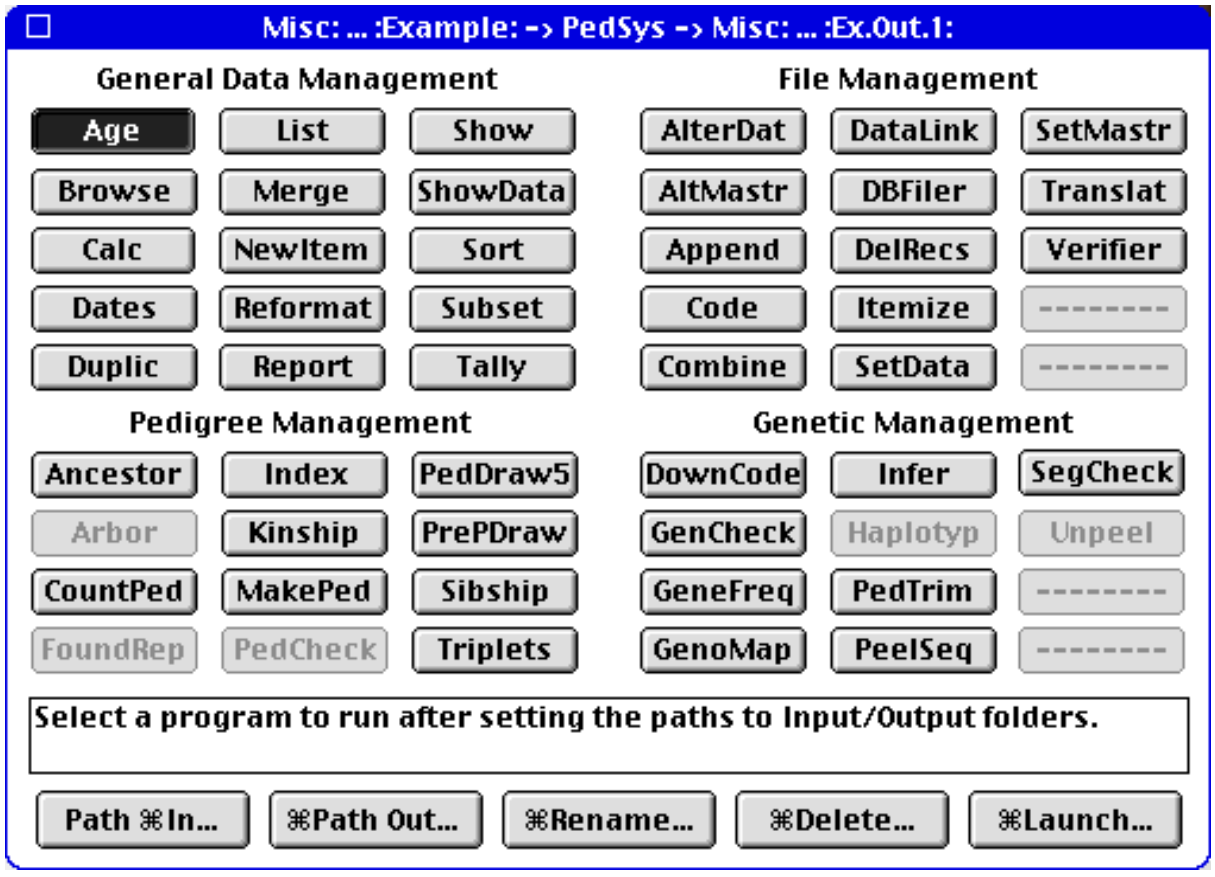
PEDSYS for Macintosh features its own set of customized icons identifying applications and support files which come with the system, or are created by it. The same icon is used for all applications except *PedSys*. Double-clicking any of the document icons causes *PedSys* to be launched.



If PEDSYS's customized icons do not appear when displaying the icon view in the **PedSys** folder, rebuild the DeskTop files on the hard drive volume by restarting the computer, and pressing the Command-Option keys together. Click the OK button when asked if you want to rebuild the DeskTop file of the disk volume containing PEDSYS.

3. Running PedSys

When the application *PedSys* is started, a "Launch" window of named buttons is displayed, with each active button representing an application stored in **PedSys.Progs**. This is the main point of reference for running applications and managing files. An application may be initiated by clicking the appropriate button, by typing the name of the application and pressing RETURN (the first 3-4 letters of a name usually suffice), or by using the up/down arrow keys to highlight a button and pressing RETURN. Functions such as **Path In** or **Rename** are triggered by clicking the button, or by pressing the Command key plus the first letter following the Command key symbol in the button label.



Once a program has started, *PedSys* reverts to background status. It is brought to the foreground again when an application ends, when a visible section of its Launch Window is clicked, or when selected from the active application menu at the far right end of the menu bar.

PedSys may also be launched conveniently by creating an alias for the application file, and moving it to the desktop for easy reference, or by dragging the application icon to the Launcher window (if it is active) to create an alias there.

Instructions for running the individual applications may be found in the *PEDSYS User Manual.*

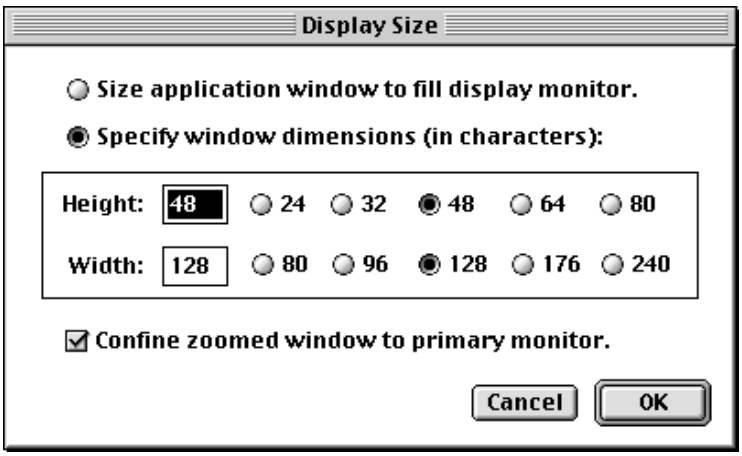
4. Setting Input/Output Data Paths

Before initiating an application, a Data Folder should be selected using the **Path In** button. Subsequently launched applications then will access this folder automatically. The **Path Out** function is used to create or select an Output Folder to receive files created by PEDSYS applications. If no separate folder is designated, output files are sent to the input Data Folder by default. There are no special restrictions on Output Folder location, although it can be useful to keep it near, or within, the Data Folder. Successively designated Output Folders can be used to store results from several analyses of the same file group, as an alternative to renaming **.OUT** files before rerunning a PEDSYS program.

Abbreviated pathnames of selected I/O folders are listed in the Launch Window title bar; while the folder names alone are shown in an application's title bar. Changes to the I/O path designations will not affect an active PEDSYS program; it must be stopped and restarted for the new pathways to be used.

Setting Application Display Window Size

The size of the display window opened by PEDSYS applications is controlled through the **Set Display Size...** command on the **Options** menu of *PedSys*, which opens a dialog titled "Display Size" when invoked.



Window size may be set exactly using the predefined values, or by entering them directly. If **Size ... to fill display monitor** is selected, the window is set to fill the monitor screen automatically when the application is launched. The last option, **Confine zoomed window...** limits the growth of the display window when resized by clicking the zoom box. If this option is unchecked, the window will assume the designated display size, which may be larger than the monitor depending on the dimensions and font size specified. On a system with more than one monitor, the zoomed display window will be located in the monitor which contained most of the original window area.

Multi-tasking

Under certain conditions, most PEDSYS applications can be run concurrently, and share access to the same files in the Data Folder. The primary requirement is that the **Allow Multiprocessing** item in the **Options** menu of *PedSys* be checked (the default); otherwise only one application at a time is allowed to launch. The second requirement is that there be enough available RAM to support multiple program partitions. A third requirement is the program not be one of eight File Management programs (*AlterDat*, *AltMastr*, *Code*, *DataLink*, *DBFiler*, *DelRecs*, *SetData* and *SetMastr*) which update Master, Data, or Support files. These applications require exclusive access to the files they are updating, so they are not allowed to run unless all active PEDSYS applications have terminated.

A program is terminated by selecting **Quit** from the **File** menu, by closing the display window, or by entering "Q" or "X" for Quit/Exit when this option is offered by a program. These methods return control to *PedSys*. Entering Command-. (period), Control-C, or Command-C also ends an application, but returns control to the Finder.

Editing Text Files

The *PedSys* application provides a simple ASCII text editor for viewing files and saving changes. It is invoked by selecting **Open** on the **File** menu and specifying a file name, or by double-clicking a PEDSYS data file from the DeskTop. This editor is based on the text editing routines provided by the Macintosh Toolbox, and thus cannot accommodate all the data in a file containing more than 32,000 characters. If such a file is opened, the first 32,000 characters are loaded, and a warning message is displayed. In these circumstances, if the file is saved under the same name, the contents of the original file beyond 32,000 characters will be lost. The text face and size are taken from the *PedSys* **Font** menu settings.

Printing

When generating printed output with *List*, *Report*, or *Tally*, the standard Macintosh **Print** dialog is presented, allowing standard printer options to be set before anything is actually printed. Any compatible printer selected through the **Chooser**, or attached as a serial port device may be used. The amount of text printed on a single page is governed by the page orientation (portrait or landscape) and the reduction/enlargement (25% - 300%) values, which may be set in the **Page Setup** dialog when the program is started. The greater the reduction, the more text is printed per page. Text is printed in the size and font selected from the **Font** menu, subject to the specified reduction/enlargement value. For example, a text size of 12 printed at 25% would yield printed text of size 4, which is quite readable when printed at resolutions of 600 DPI or more.

Mousing Around

In a modest effort to provide a little of the Macintosh "point and click" convenience to the text-based interface of the applications, mouse-clicks on the application display window may be used to enter a character or a RETURN at the prompt line. Clicking on a screen character is the same as typing it at the keyboard, while clicking an empty space is the equivalent of pressing the RETURN key . For example, in order to enter the standard Quit code, click on the "Q" shown in the display, and then on any empty screen space. This works only if the mouse cursor is clicked and released within the same character space. If the mouse is moved before the button is released, the character or RETURN will not be entered. Currently only one character may be entered at a time this way, and only characters that are visible on the screen. This feature may be elaborated upon in future editions, perhaps with support for a simple Copy & Paste approach to entering filenames.

Comments

If you have comments or problem descriptions to forward, please fill out this page and mail to the address below, or send as e-mail if preferred.

Name: _____

Address: _____
_____
_____

E-mail:_____
Phone : (_____) _____

System Configuration:

MacOS version number _____
Machine model _____
Memory capacity _____
Disk storage available _____
Printers in use _____
Special software in use _____

Problem Description or Comments:

Please send the report to:

Bennett Dyke (bdyke@darwin.sfbr.org), or
Paul Mamelka (paulm@darwin.sfbr.org)

Department of Genetics
Southwest Foundation for Biomedical Research
P. O. Box 760549
San Antonio TX 78245-0549

Voice: (210) 674-1410
FAX: (210) 670-3317

Thanks.

The SFBR web site can be reached by pointing your browser to ***www.sfbr.org***. The same address may be used for anonymous FTP transfers. Future PEDSYS updates will be posted here when available.