AIIM

Association for Information and Image Management

1100 Wayne Avenue, Suite 1100
Silver Spring, Maryland 20910
301/587-8202

Centimeter

Inches

MANUFACTURED TO AIIM STANDARDS
BY APPLIED IMAGE, INC.

O

f

SAND93-3834
Unlimited Distribution
Printed May, 1994

UC-900 Distribution
Category Number

ETPRE User's Manual Version 3.00

Robert J. Roginski
Assessment Technologies Department
Sandia National Laboratories
Albuquerque, New Mexico 87185

Abstract

ETPRE is a preprocessor for the Event Progression Analysis Code EVNTRE. It reads an input file of event definitions and writes the lengthy EVNTRE code input files. ETPRE's advantage is that it eliminates the error-prone task of manually creating or revising these files since their formats are quite elaborate. The user-friendly format of ETPRE differs from the EVNTRE code format in that questions, branch references, and other event tree components are defined symbolically instead of numerically. When ETPRE is executed, these symbols are converted to their numeric equivalents and written to the output files using formats defined in the EVNTRE Reference Manual (SAND88-1607). Revisions to event tree models are simplified by allowing the user to edit the symbolic format and rerun the preprocessor, since questions, branch references, and other symbols are automatically resequenced to their new values with each execution.

ETPRE and EVNTRE have both been incorporated into the SETAC event tree analysis package.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

ACKNOWLEDGMENTS

The author wishes to acknowledge Martin K. Fuentes of Sandia National Laboratories for his significant contribution to the development of ETPRE. He was responsible for the original design of the ETPRE symbolic input format.

Gregory D. Wyss of Sandia National Laboratories provided valuable information that made correcting several problems a relatively easy task. In addition, he defined the method and syntax which interface the LHS code to ETPRE when sampling is to be performed. He also supplied the example problems found in Appendices A through F and the definition of the LHS File Format that appears in Appendix G.

On several occasions, Kevin J. Maloney of Sandia National Laboratories provided valuable assistance in verifying the format of the ETPRE output files.

Contents

1. Introduction	7
2. Description of ETPRE	7
2.1 Input structure	7
2.2 Output Structure	9
2.3 Problem Size	10
2.4 Machine Compatibility	11
3. Input Files	12
3.1 General Considerations	12
3.2 Keyword Input File	12
3.3 Tree Definition File	18
3.3.1 General Description of Tree Definition File	18
3.3.2 Tree Definition Header	24
3.3.3 Question Definitions	24
3.3.4 Binning Information Segment	31
3.3.5 Sorting Information Segment	33
3.3.6 Rebinning Information Segment	34
3.3.7 Dataset Information Segment for LHS Sampling	35
3.3.8 Pointers and Pointer References	36
3.4 LHS-like Sampling Information Files	37
4. Output Files	39
4.1 Tree Definition File	39
4.2 Binning and Sorting Definition File	40
4.3 Rebinning Definition File	40
4.4 Sample Definition File	40
4.5 Sample Vector File	41
4.6 Error Files	42
4.6.1 Fatal Error Message File (ETPRE.ERR)	42
4.6.2 Warning Message File (ETPRE.WNG)	43
4.7 QA Information File	43
5. Debugging Considerations	44
References	45
Appendices	46
Appendix A - Example Problem 1	47
Appendix B - Example Problem 2	52
Appendix C - Example Problem 3	57
Appendix D - Example Problem 4	62
Appendix E - Example Problem 5	67
Appendix F - Example Problem 6	75
Appendix G - LHS File Format	86

Tables

1. ETPRE Version 3.xx Limits	11
2. Boolean Operators Recognized by ETPRE.....	19
3. Comparison Methods Used in Question Types 5 Through 8	21

ETPRE User's Manual Version 3.00

1. Introduction

Probabilistic risk assessment (PRA) studies are conducted for many types of problems at Sandia National Laboratories. The event progression analysis code EVNTRE (see Ref. 1) is a useful tool for processing the complex scenarios that are typical of these studies. Preparing the input for large EVNTRE runs has a tendency to be error-prone, since many components of the tree must be specified in two or more places. Errors may also be introduced when it is necessary to insert a question in the middle of the tree since all subsequent questions and question references must now be manually renumbered to reflect the new question number. In addition to being annoying and time-consuming, even a simple modification to an EVNTRE input file has the potential to become a major quality assurance (QA) problem if the user fails to correct all references affected by the revision. Several users of EVNTRE had mentioned that it would be beneficial if event tree logic could be defined in a format that is more symbolic in nature and if it would also define parameters using position or context wherever possible. These suggestions eventually led to the development of the preprocessor code ETPRE. This code's primary purpose is to eliminate the error-prone portions of the EVNTRE input file preparation/revision process. ETPRE accomplishes this each time it writes (in EVNTRE input format) the output files, because it always translates the symbolic definitions to their numeric counterparts in as many places as required by the EVNTRE input formats.

This manual describes the operation and syntax of the preprocessor code ETPRE, Version 3.xx. The manual also describes the relationship of ETPRE with other codes having one or more file formats that must be either read or written by ETPRE. However, no attempt is made to explain the functionality or operation of any feature(s) supported by these codes. For assistance in this area, the user should consult the documentation for the code in question. All reference material for codes that are directly or indirectly related to ETPRE are listed as References on Page 45 of this manual.

2. Description of ETPRE

ETPRE reads various types of information from one or more input files and writes the converted information to a maximum of eight different output files, depending on options selected and errors encountered. This section gives a general description of each file and its function. Information regarding problem size (i.e., program limits) and machine compatibility is also included in this section.

2.1 Input structure

With very few exceptions, ETPRE input files are read in a free format that is token-oriented. A token is defined as a grouping of one or more contiguous characters

delimited by at least one tab or space character (or any combination of tabs and spaces). In most instances, the type of token required (character string, floating-point constant, or integer constant) is determined by the context of the input. All tokens are verified for correctness of syntax and for other relevant conditions as they are encountered in the current line of input. Although free format is allowed, readability can be significantly improved by using a tabular format as shown in several of the examples in this manual.

ETPRE is controlled through a keyword file whose pathname is specified on the program execution command line. If no keyword pathname is entered on the command line, ETPRE prompts the user to enter one. The keywords contained in this file define input/output file specifications and other program options. The ETPRE keyword file can be read directly by EVNTRE, PSTEVNT (see Ref. 2) and other codes in the SETAC system (see Ref. 3).

Excluding the keyword input file, the following input files can be read by ETPRE:

Tree Definition File

The first segment of this file supplies the event tree structure, default probability, and parameter values. The second segment supplies the definitions of the binning characteristics and the corresponding Boolean expressions. The third segment supplies the sorting input information. The fourth segment supplies rebinning characteristics and the corresponding Boolean expressions. The fifth and final segment supplies additional information that defines LHS-related symbols used in subsequent sampling. Only the first two segments of this file are required.

Sampling Information Files

These files contain the split fractions and parameter values that are used by ETPRE when any of the sampling options are used. The information includes point estimate values and values computed for each observation to be sampled. It also contains the symbolic information that allows ETPRE to reference any and all values by name.

Alternate Rebinning File

This optional file contains sorting and rebinning information that will be used in place of the information found in the

sorting and rebinning segments of the Tree Definition File. Specifying an Alternate Rebinning file makes it convenient to rebin previously sampled results without modifying the original Tree Definition File.

These files are described in more detail in Section 3. Sampling information files are only required if any of the parameter values or split fractions are to be passed on to the sampling facility found in the EVNTRE code.

2.2 Output Structure

As many as eight separate output files can be produced by ETPRE:

Fatal Error Message File	This file (ETPRE.ERR) describes all the fatal errors that ETPRE was able to detect before terminating execution.
Warning Message File	This file (ETPRE.WNG) will describe all the minor problems that ETPRE was able to detect while executing.
Tree Definition File	This file supplies the event tree structure, the default probability, and the parameter values. This file is written in the EVNTRE code input format.
Binning and Sorting Definition File	This file supplies EVNTRE with the definitions of the binning characteristics along with the corresponding Boolean expressions. It also contains sorting information. A default sort is generated if sorting information is not explicitly specified in the sorting segment of the ETPRE Tree Definition File.
Rebinning Definition File	This file supplies rebinning information that can later be used as input to the PSTEVNT code. A default rebin is generated if rebinning information is not explicitly specified in the rebinning segment of the ETPRE Tree Definition input file. This file also contains sorting, binning and

rebinning information that can later be used as input to the PSTEVNT code.

Sample Definition File

This file supplies EVNTRE with the specifications required to set up the sampling modes for the tree. These specifications include the pointers to probabilities and parameters to be sampled.

Sample Vector File

This file supplies EVNTRE with the vectors whose elements are the values to be substituted into the tree for each iteration (observation) in the sample.

QA Information File

This optional output file lists the contents of all tree-related tables (arrays) internal to ETPRE. It is intended as a programmer's code maintenance tool that, in general, will be of little value to the ETPRE user.

These files are described in further detail in Section 4. The Sample Definition and Sample Vector files are only generated if a parameter value or split fraction is to be passed on to the sampling facility found in the EVNTRE code. The Rebinning Definition File is always generated regardless of whether or not the PSTEVNT code is to be executed.

2.3 Problem Size

The size of an event tree, which can be processed with ETPRE (Version 3.xx), is limited by the internal table (array) sizes defined in Table 1. It should be noted that these are the limits for the entire tree and that they are not applicable on a per question basis. For example, the user may define questions having as few as two branches or may define other questions having ten or more branches. This is acceptable as long as none of the total limits found in Table 1 are exceeded.

If there is concern about exceeding any of these limits, the user can refer to the "RESOURCE USAGE" portion of the ETPRE output displayed on the computer screen before the execution terminates. This output will show how many times each item was required for that particular tree. If the tree is very large, a limit may be exceeded even before processing begins. In this case, execution will be terminated with an appropriate error diagnostic. This diagnostic should be brought to the attention of the Sandia point of contact so that the limit can be increased.

Questions	200
Sequences	32
Branches	600
Split Fractions	2400
Expressions	600
Symbols	1800
Reference Parameters	200
Parameter Values	2400
Pointers	200
Reference Types	1800
Split Fraction Comparison Values	1200
Comparison Parameters	600
Parameter Comparison Types	600
Internal Function References	600
Sample Vectors	256
Vector Labels	256
Binning Characteristics	32
Binning Outcomes	128
Binning Symbols	384
Sorts	16
Sort Keys	128
Rebinning Characteristics	32
Rebinning Outcomes	128
Rebinning Symbols	384
LHS Symbolic Names	256
LHS File Paths	16
LHS Sampled Symbols	512

Table 1. ETPRE Version 3.xx Limits

2.4 Machine Compatibility

Version 3.xx of ETPRE runs only on an IBM or compatible personal computer (PC). However, there are two executable files: one is compiled with Version 5.1 of the Microsoft Fortran compiler; the other is compiled with Version 5.01 of the Lahey F77L EM/32 compiler. Ignoring minor differences in the format of floating point numbers in some files (such as in the Sample Vector File), the two executables are functionally equivalent in all respects. However, the hardware requirements are quite different.

The Lahey executable requires a 386 (or higher) central processing unit (CPU) and a 387 numeric coprocessor. The coprocessor is not a concern with 486 machines since it is an integral part of the CPU, but 486SX machines require a 487 or equivalent math chip. The performance of the Microsoft executable is slightly slower than that of the Lahey, but it will also execute on a machine having a 286 CPU. It can also execute without a numeric coprocessor. If a coprocessor is detected at runtime, however, it will be used and will execute faster than it would without a coprocessor installed.

The Microsoft executable currently requires 433K (kilobytes) of conventional RAM (random access memory). The Lahey executable currently requires 174K of conventional RAM and an additional 480K of XMS (extended) RAM. All memory requirements are subject to change with each new release of ETPRE.

Future versions of ETPRE will probably introduce substantially higher limits for the Lahey compiled executable. This is possible since the Lahey compiler can address a large amount of memory by using its protected mode capabilities. The Microsoft version, however, will retain smaller limits so that event trees of small to medium sizes can still be processed on older DOS machines equipped with non-state-of-the-art CPUs that contain only conventional RAM.

Future requirements may also dictate the need to port ETPRE to additional platforms such as UNIX, Windows NT, etc. Conversion to these systems is not difficult because ETPRE is written using portable Fortran 77 statements in all but a few instances.

3. Input Files

3.1 General Considerations

In general, ETPRE input files consist of free-formatted records that can be up to 2000 characters in length. The few exceptions to the free-format rule are explained where applicable in Sections 3.3 and 3.4. Rules governing the syntax of specific input files are explained in Sections 3.2 through 3.4. The records consist of tokens that are delimited by at least one space or tab character or any combination of these two characters.

3.2 Keyword Input File

ETPRE obtains its execution options and instructions from a series of keywords in a file whose pathname is specified on the program execution command line. If no pathname is specified, ETPRE will prompt the user to enter one interactively. Some keywords set flags while others indicate that a value (integer or character) is to be read. In this case, a missing or illegal value will cause ETPRE to abort execution and to write a fatal error message to the error file ETPRE.ERR.

Some of the keywords acknowledged by ETPRE are also valid keywords for the EVNTRE code. This is necessary since some of the ETPRE output files become EVNTRE input files when ETPRE terminates successfully. Keywords that have no meaning to ETPRE are simply ignored, so the same keyword file can be used for both codes. The EVNTRE code performs a validity check for all keywords recognized by both codes; no attempt is made to duplicate keyword validation effort in the preprocessor.

The remainder of this section describes the keywords currently recognized by ETPRE. The keywords are shown here in upper case, but will be accepted in any combination of upper of lower case, as long as the spelling agrees. As in the EVNTRE code, self-documenting keyword files can be produced by using the '\$' mode of commenting the rest of the line (see Appendix A of Reference 1).

ADDSEQ

Instructs ETPRE to insert the question "Which sequence?" as Question Number 1 in the event tree. This feature makes it easy for the user to isolate sequence-specific output since the output files will be modified as described here.

The inserted question will be a Type 1 question having as many branches as the number of user-defined sequences. The branch labels for the question will be copies of the respective sequence labels. The split fractions for the new question will only equal 1.0 where the branch number equals the sequence number; it will equal 0.0 in all other branches. In other words, the diagonal of this square matrix will consist of 1s with 0s in the triangular regions.

If the number of defined sequences is greater than 1, a new characteristic identified as "Sequence" will be inserted as binning/rebinning Characteristic Number 1. The new characteristic will have as many outcomes as there are user-defined sequences. The outcome labels for this new characteristic will refer to the branch labels for the inserted Question Number 1; as previously stated, these were defined as the respective sequence labels for the event tree.

Sorting information is also modified for multiple-sequence event trees. A new sort key identified as "Sequence" will be inserted as the primary (first) key for each sort, regardless of whether the sort was user-defined or was created by default. This applies to both the Rebinning

Definition File and the sorting information that will be written at the end of the Binning Definition File.

Example: ADDSEQ

BININ *binfile* Specifies *binfile* as the name of the ETPRE binning output file. This file will become the binning input file for the EVNTRE code. If the keyword BININ is not found in the keyword input file, ETPRE will assume a filename that uses the same path and name as the Tree Definition File specified by keyword PRETRIN, but that has a file extension of .BIN. If BININ is not specified, the keyword file cannot be fed directly to EVNTRE. The parameter *binfile* may specify a full path up to 50 characters in length.

Example: BININ C:\ETPRE\PROBLEM1.BIN

ENDKEY Specifies the logical end of the keyword file. Any keywords following the optional keyword ENDKEY will be ignored.

Example: ENDKEY

LHSOBS *nobs* Specifies *nobs* as the number of observations (iterations) to be sampled. If LHSOBS is specified, the number of observations in each sampling input file must equal *nobs*.

Example: LHSOBS 200

LHSOUT *lhsoutfile* Specifies *lhsoutfile* as the name of the file that will be the LHS output file. The parameter *lhsoutfile* may specify a full path of 50 characters in length. ETPRE reads this file to obtain values associated with distributions calculated by the LHS code.

Example: LHSOUT C:\ETPRE\LHSOUT.DAT

LHSPOST *lhspstfile* Specifies *lhspstfile* as the name of the file that will be the LHS postprocessor output file. The parameter *lhspstfile* may specify a full path of 50 characters in length. ETPRE reads this file to obtain the values of expressions that are evaluated by the LHS postprocessor code.

Example: LHSPOST C:\ETPRE\LHSPOST.DAT

NEWPOST *altrbfile* Instructs ETPRE to use *altrbfile* as an alternative source for reading its sorting and rebinning information. The file specified by *altrbfile* contains sorting and rebinning information that will be substituted for any sorting and rebinning information found in the Tree Definition File. Specifying Keyword NEWPOST allows the user to rebin an existing event tree without modifying the original ETPRE input file. This is often required for QA purposes. The parameter *altrbfile* may specify a full path up to 50 characters in length.

Example: NEWPOST C:\ETPRE\REBIN-1.DAT

NOSORT Instructs ETPRE not to generate sorting information in the Binning and Sorting Definition File that is subsequently read by the EVNTRE code. Using NOSORT will also suppress the default sort that is assumed if sorting information is not explicitly specified. If NOSORT is specified and sorting information is found in the Tree Definition File (or Alternate Rebinning File), a diagnostic is written to the Warning Message File ETPRE.WNG. Keyword NOSORT will always take precedence over any sorting definition information found.

Example: NOSORT

POSTFL *rebinfile* Specifies *rebinfile* as the name of the ETPRE Rebinning Definition File. This ETPRE output file will become the rebinning input file for the PSTEVNT code. If the keyword POSTFL is not found in the keyword input file, ETPRE will assume a filename that uses the same path and name as the Tree Definition File specified by keyword PRETRIN, but that has a file extension of .RBN. If POSTFL is not specified, the keyword file cannot be fed directly to EVNTRE. The parameter *rebinfile* may specify a full path up to 50 characters in length.

Example: POSTFL C:\ETPRE\REBINOUT.RBN

PRETRIN *questfile* Specifies *questfile* as the name of the Tree Definition File for the preprocessor. Absence of this keyword will cause ETPRE to abort with a fatal diagnostic, since this is a

mandatory input file. The parameter *questfile* may specify a full path up to 50 characters in length.

Example: PRETRIN C:\ETPRE\PROBLEM1.ETI

QAOUT *qafile* Specifies *qafile* as the name of the QA information output file. This optional file will list the contents of all tree-related tables (arrays) internal to ETPRE. If the keyword QAOUT is not found in the keyword input file, the QA information file will not be generated. The parameter *qafile* may specify a full path up to 50 characters in length.

Example: QAOUT C:\ETPRE\PROBLEM1.QAF

SAM1IN *svfile* Specifies *svfile* as the name of the sample vector output file. This file will become the sample vector input file for the EVNTRE code. If the keyword SAM1IN is not found in the Keyword Input File, ETPRE will assume a filename that uses the same path and name as the Tree Definition File specified by keyword PRETRIN, but that has a file extension of .SVF. The parameter *svfile* may specify a full path up to 50 characters in length.

Example: SAMDIN C:\ETPRE\PROBLEM1.SVF

SAMALLB Is not actually used by ETPRE. Presence of SAMALLB in the Keyword File instructs the EVNTRE code to "sample all branches" of sampled questions. ETPRE merely verifies the presence of SAMALLB in the Keyword File. If any sampling is to be performed and SAMALLB was not specified, then a diagnostic is written to the Fatal Error Message File and ETPRE execution is aborted.

Example: SAMALLB

SAMDIN *sdfile* Specifies *sdfile* as the name of the sample definition output file. This file will become the sample definition input file for the EVNTRE code. If the keyword SAMDIN is not found in the Keyword Input File, ETPRE will assume a filename that uses the same path and name as the Tree Definition File specified by keyword PRETRIN, but that has a file extension of .SDF. The

parameter *sdf* may specify a full path up to 50 characters in length.

Example: SAMDIN C:\ETPRE\PROBLEM1.SDF

SAMMOD *modeval* Defines the sampling mode to be the integer constant value *modeval*. A value of 10 indicates no sampling or use of LHS. A value of 11 indicates that LHS is to be used for a full uncertainty analysis. A value of 12 instructs the preprocessor not to generate the sample definition/vector files even if values to be sampled (i.e., LHS symbols enclosed in parentheses) are found in the Tree Definition File. Point estimate values for LHS symbols used in SAMMOD 12 are taken from the DATA: statements and from the DATASET: file segment found in the Tree Definition File. A value of 13 for *modeval* functions the same as 12 except that the point estimate values are taken from the point estimate (first) section of the LHS-like sampling files. If the keyword SAMMOD is not found, ETPRE will use the default starting value of 10. The user should be aware that the sampling mode will automatically switch from 10 to 11 if any LHS sampling symbols are found in the Tree Definition File.

Example: SAMMOD 11

TREEIN *treefile* Specifies *treefile* as the name of the ETPRE tree output file. This file will become the tree definition input file for the EVNTRE code. If the keyword TREEIN is not found in the keyword input file, ETPRE will assume a filename that uses the same path and name as the Tree Definition File specified by keyword PRETRIN, but that has a file extension of .ETO. If TREEIN is not specified, the keyword file cannot be fed directly to EVNTRE. The parameter *treefile* may specify a full path up to 50 characters in length.

Example: TREEIN C:\ETPRE\PROBLEM1.ETO

Appendices A through F contain complete keyword input file listings of example Event Trees 1 through 6, respectively. These are the first files listed in each Appendix.

3.3 Tree Definition File

3.3.1 General Description of Tree Definition File

The Tree Definition File read by ETPRE contains up to five segments. The first segment contains information defining the event tree logic. This information consists of the tree definition header followed by the various questions that make up the tree. All question types available in the EVNTRE code are supported by ETPRE; however, the syntax of each one is quite different in ETPRE. The last four file segments respectively contain binning, sorting, rebinning and sampling information. It was necessary to incorporate all types of information into a single file since the preprocessor requires access to all previously defined symbol tables in order to perform the translation to the EVNTRE format. All five segments of the file use symbols whenever possible to define the various components of the event tree.

Appendices A through F contain complete listings of tree definition files in ETPRE format for six example problems. These are the second of several files listed in each Appendix.

3.3.1.1 Comments

Comments can be placed anywhere in the Tree Definition File using any of three different methods. A trailing comment may be added to a line by appending a '\$' at the end of the desired line. In this case, however, the '\$' must be preceded by at least one space or tab character so that it is not interpreted as part of the previous token. ETPRE will then ignore the '\$' and everything that follows it on that line. Whole line comments are allowed if a '\$' is the first non-blank, non-tab character on a line. However, the '\$' must not start beyond Column 30 for technical reasons. The third method of commenting is the use of totally blank lines that may contain only spaces or tab characters. Unlike the EVNTRE code, ETPRE does not require comment lines anywhere in the Tree Definition File.

3.3.1.2 Continuation Lines

Continuation lines are also possible in the Tree Definition File. Any line can be continued by placing a continuation line character ('#' or '%') as the last character on the line. The continuation character must be preceded by at least one space or tab character. When ETPRE detects the continuation character as the last token on the line, it will read the next line from the file and continue processing tokens. Continuations are particularly useful if the user must prepare the input file with a text editor having a line size limit which is smaller than 2000 characters. Two thousand characters is the maximum line size allowed by ETPRE. Trailing comments are legal on continuation lines as long as there is at least one tab or space character between the continuation character and the '\$' that indicates the beginning of the trailing comment.

3.3.1.3 Numeric Constants

Numeric constants are required input at several places in the Tree Definition File. The required constant may be either an integer or a real number. In places where an integer value is expected, the decimal point, with or without trailing digits, must not be contained in the constant. If a real value is expected, ETPRE will accept the value in any of three formats: fixed point, floating point (E-format), or integer. In this case, the value will be converted to a Fortran double precision real variable regardless of the format used.

3.3.1.4 Column One Usage

Input beginning in Column 1 of the file is reserved for the event tree identifier, the text which begins a new question definition, or the reserved keywords that separate the various file segments. Specific rules regarding the use of Column 1 are explained where applicable in the various sections that follow.

3.3.1.5 Double Quote Characters

Many users may elect to create the Tree Definition File using one of the commercially available spreadsheet programs such as Microsoft Excel. On occasion and without warning, the spreadsheet program may enclose character strings within a pair of double quote characters while saving the file in ASCII format. For this reason, ETPRE will generally ignore the double quote character. The only exception to this is the character string that defines a long binning or rebinning descriptor.

3.3.1.6 Boolean Expressions

Boolean expressions are allowed in even-numbered question types where branch point probabilities are dependent on previous branches. Expressions may also be used in the binning and rebinning segments of the file. With only minor differences, the Boolean expressions are governed by those rules that apply in the EVNTRE code. These minor differences are explained in this manual. ETPRE Boolean operators are functionally equivalent to their EVNTRE counterparts, except that some additional forms of syntax are available to the user. The operators allowed in an ETPRE Boolean expression are shown in Table 2.

<u>Valid ETPRE Operators</u>			
Function	(1)	(2)	(3)
AND	*	&	AND
OR	+		OR
NOT	-	/	NOT

Table 2. Boolean Operators Recognized by ETPRE

Expressions shown in the examples that illustrate the various question types may contain spaces or tabs surrounding the operators. These are required, however, where necessary to prevent interpretation of the operator as part of an adjacent token. For example, a space or tab should precede and follow the third form (case-insensitive alpha) of any operator. A space or tab must also precede all forms of the NOT operator. General rules that apply to ETPRE Boolean expressions are essentially the same as those that apply to the EVNTRE code. However, they are listed here for user convenience. Operators referenced in the rules below can be in any of the three forms shown in Table 2.

General Rules for Boolean Expressions

- A. NOT operator is valid only immediately preceding a branch label, but not before an open parenthesis.
- B. AND is evaluated before OR.
- C. Expressions enclosed in parentheses are evaluated before AND and OR.
- D. The AND operator is the default and is optional. Therefore, a blank space between branch labels in a Boolean expression is interpreted as AND.
- E. Parentheses can be nested up to five levels.

3.3.1.7 Calculated Branch Split Fractions

In question Types 1 through 4, all branch split fractions are explicitly specified. In question Types 5 through 8, the user must supply the information required to calculate branch split fractions. This information consists of the names of the previously defined variables (reference parameters) that are to be combined, a method for combining them, a comparison type that will be used to calculate the branch split fractions, and finally, the values to be compared against. The functionality of this information is well defined in the EVNTRE Reference Manual and will not be repeated here. However, since the syntax required to represent these items in ETPRE format is significantly different, this section is provided for user assistance.

The following lines describe the general format in which ETPRE expects to read the information for these question Types 5 through 8. All items are case-insensitive. Spaces are shown surrounding the parentheses and following the comma, but are not required.

*comparison_method combination_type (variable1[, variable2]...)
boolean_expression comparison_values*

The *comparison_method* in the above syntax should be one of the four described in Table 3. The table also repeats a few syntax rules that must be observed in order to set up a Type 5 through 8 question in ETPRE format. Each method is explained in detail starting on Page 21 of the EVNTRE Reference Manual.

For *combination_type* in the above syntax, the user must substitute one of the following valid types: ADD, AND, MAX, MIN, MULT, FUN-xxxxxx or SUB-xxxxxx. These combination types are explained on Page 20 of the EVNTRE Reference Manual. As shown, the syntax requires the names of one or more variables that must be defined in a preceding question or questions of Type 3, 4, 7, or 8.

The input item identified as *boolean_expression* should be present only for question Types 6 and 8. The syntax for these expressions is defined on Page 19.

The input item identified as *comparison_values* should contain the correct number of values for the method chosen. Table 3 defines the required number of values for each comparison method.

Comparison Method	Required No. of Branches for this Question	Required No. of Comparison Values (Parameters)
EQUAL	2	0
NEQUAL	2	0
NNEQUAL	2	0
NORMAL	2	2
NNORMAL1	2	2
NNORMAL2	2	2
THRESH	2 or more	(No. of Branches) - 1
GETHRESH	2 or more	(No. of Branches) - 1
NTHRESH	2 or more	(No. of Branches) + 1

Table 3. Comparison Methods used in Question Types 5 through 8

The user must be aware that in a Type 5 or 7 question that uses any of the EQUAL, NEQUAL, or NNEQUAL comparison methods, the entire second line must be omitted. The reason is because neither a *boolean_expression* or any *comparison_values* is required. For Type 6 or 8 questions using any of these comparison methods, only the *boolean_expression* should be found on the second line.

3.3.1.8 Data Statements for LHS Sampling

The "Data:" statement is the one of two methods for defining data items that are related to LHS sampling. The second is the "Dataset:" segment of the Tree Definition File that is described in Section 3.3.7 beginning on Page 35 of this manual. Examples of "Data:" statements can be found in Example Problem 6 in Appendix F.

The "Data:" statement is a single line definition of one named data item related to the LHS code. ETPRE allows "Data:" statements only in the following locations in the Tree Definition Input File.

- A. Following the Tree Definition Header, but preceding the text defining Question 1.
- B. Following the last sequence of each case of each question, but preceding any variable definitions.
- C. Following the last sequence of each case in the variable definition section of Question Types 3, 4, 7 and 8.

There are four different syntax forms of the "Data:" statement. The first three parameters of a "Data:" statement are identical in the first three syntax forms. The first parameter is the case-insensitive token "DATA:". The second parameter specifies the name of the LHS symbol. This symbol consists of a combination of 1 to 16 case-insensitive characters that cannot be interpreted as a numeric constant under the ANSI Fortran standard. The LHS symbol must also be unique and begin with an alpha (A-Z or a-z) character. The optional third parameter specifies a numeric constant that is the value used when sampling is not performed (i.e., SAMMOD 12). In this scenario, the parameter *point_value* is mandatory unless alternate rebinning is in effect (see Keyword NEWPOST on Page 15).

Syntax 1: DATA: *name* [*point_value*] FILE *path_name*

Examples: DATA: Sample_1 0.429 File C:\STUDY1\LHSFILE1.DAT
DATA: Sample_2 0.265 File C:\STUDY1\LHSFILE1.DAT
DATA: IMP_ANGLE FILE FLDATA.DAT

In Syntax 1 above, presence of the case-insensitive parameter FILE indicates that the next parameter to be read (*path_name*) will be the name of an LHS-like file that must be subsequently read by ETPRE. The path or file associated with *path_name* may be repeated in other "DATA:" statements (as shown above) if two or more LHS symbols are defined in that particular file. However, the number of unique paths or files specified by *path_name* is currently limited to 16. The parameter *path_name* is case-insensitive and may be 1 to 50 characters in length. If *path_name* specifies only a filename, ETPRE will expect to find the file in the current drive and directory.

Syntax 2: DATA: *name* [*point_value*] *dist_name* *param1* *param2*

Examples: DATA: RLOSP-1 0.05 Lognormal 0.02 3.00
DATA: RLOSP-2 0.05 Beta 0.0 0.5 1.0 2.0
DATA: X-Range Normal 75.0 5.5

In Syntax 2 above, the parameter *dist_name* is assumed to be the name of a distribution for LHS (see Ref. 4). The parameters that follow *dist_name* must be appropriate for the distribution selected. ETPRE currently does not perform any validity checks on *dist_name* or the parameters that follow. In this syntax, the LHS symbol identified by *name* must be found in the LHS Output File (see Keyword LHSOUT on Page 14).

Syntax 3: DATA: *name* [*point_value*] = *lhs_expression*

Examples: DATA: RL-Fails 0.10 = RL1 + RL2 + RL3 + RL4
DATA: Max-X-Range 75.0 = X-Maximum + X-Tolerance
DATA: Diagonal = SQRT (Length ** 2 + Width ** 2)

In Syntax 3 above, the required '=' (equal sign) specifies that the next parameter to be read is an expression containing at least one previously-defined LHS symbol. As shown, the *lhs_expression* may contain ANSI standard Fortran numerical operators and intrinsic function references. Each numerical operator in the *lhs_expression* must be preceded and followed by at least one space or tab character. ETPRE currently does not perform any validity checks on the *lhs_expression*; this occurs in the LHS postprocessor code at evaluation time. When Syntax 3 is used, ETPRE expects to find the LHS symbol *name* defined in the LHS Postprocessor Output File (see Keyword LHSPOST on Page 14). ETPRE also expects to find a value of the associated *lhs_expression* for each observation when sampling is performed (i.e., SAMMOD 11).

Syntax 4: DATA: CORRELATE *item1_name* *item2_name* *corr_value*

Examples: DATA: Correlate Alpha-Dist StmExp-Lop 0.999
DATA: Correlate Alpha-Dist StmExp-Hip 0.999

In Syntax 4 above, the case-insensitive parameter CORRELATE allows the user to define a correlation between two LHS data items. The two data items are LHS symbols that must be defined in other "Data:" statements as LHS distributions using Syntax 2. Using Syntax 1 or 3 to define either correlated item will cause LHS to abort with a fatal diagnostic. The parameter *corr_value* specifies the value that defines the correlation between the two items. The definition of *corr_value* and other details regarding correlation are explained in Reference 4.

Additionally, the following rules apply to all syntax forms of the "Data:" statement.

- A. The statement must not begin in Column 1.
- B. The statement may be continued in two or more lines using a '#' or '%' (see Section 3.3.1.2 on Page 18).
- C. Trailing comments are permitted following any line in the statement (see Section 3.3.1.1 on Page 18).

3.3.2 Tree Definition Header

An example of a Tree Definition Header in the ETPRE input format is shown below. This header is taken from example problem Number 1 in Appendix A.

```
Business travel example with simple uncertainty.  
Optimistic      0.10  
BestGuess       0.80  
Pessimistic     0.10
```

The Event Tree Identifier (title for the tree) is shown here on the first line. However, ETPRE will assume that the first non-blank, non-comment line in the file contains the identifier, which is limited to 76 characters. This example requires three sequences (i.e., multiple evaluations of the same event tree). Each sequence definition line must contain a sequence label and an initial frequency value. A tabular format as shown is suggested for improved readability, but is not required. The Event Tree Identifier may begin in Column 1 as shown, although this also is not required. The sequence definition lines which follow must not begin in Column 1. The user should be aware that sequence labels longer than 16 characters will be truncated to 16 characters in order to be consistent with the EVNTRE code.

Unlike the EVNTRE code, the user is not required to specify the number of sequences or questions in the Tree Definition Header portion of the file. ETPRE will determine the sequence count from the number of sequence definition lines found. The question count will be determined automatically as ETPRE counts the questions that follow the Tree Definition Header.

3.3.3 Question Definitions

The definitions of the questions that define the event tree logic immediately follow the Tree Definition Header. This section will provide an example of each of the eight question types along with text describing what is unique about each one. First of all, it is necessary to describe the first two lines of a question definition, since the order and syntax are common to all eight types.

The first line of any question definition is the question text or verbal definition of the question. Question text is currently limited to 76 total characters. However, to be recognized as the start of a new question definition, the question text must also satisfy one or both of the following conditions:

- A. Question text must begin in Column 1.
- B. Question text must be terminated with a '?' (question mark).

The second line of a question definition specifies the branch labels for the question. This line must not begin in Column 1. Each branch label found on this input line must observe all of the following rules:

- A. A branch label is case-insensitive.
- B. A branch label must begin with an alpha (a-z or A-Z) character.
- C. A branch label must not exceed 16 characters in length.
- D. A branch label must not contain tabs or spaces.
- E. A branch label must be unique within the entire tree.

The second line of a question definition may also contain one of three special tokens which defines how ETPRE must validate user-supplied split fractions in Question Types 1 through 4. This allows the user to specify split fractions that are not required to sum to unity (1.0). If present, the token must immediately precede the first branch label. These case-insensitive tokens are defined as follows:

PROB=1	Instructs ETPRE to verify that the split fractions in all branches of the question sum to unity within a tolerance of $\pm 1.0E-06$. Individual split fractions must be in the range of zero to one inclusive. This is the default method assumed if none of the three special tokens are found.
PROB<1	Instructs ETPRE to verify that the split fractions do not exceed unity plus a tolerance of $1.0E-06$. Individual split fractions must be in the range of zero to one inclusive.
PROB>0	Instructs ETPRE to verify that the split fractions do not sum to less than zero. Individual split fractions must be greater than or equal to zero. Any positive sum of the fractions is valid.

The user must be aware that the method is selected or defaulted on a per-question basis and cannot change from case to case within the same question. If PROB<1 or PROB>0 is encountered in a question whose type is not 1 through 4, the token is ignored and a diagnostic is written to the Warning Message File ETPRE.WNG.

3.3.3.1 Type 1 Question

A Type 1 question is one whose split fractions do not depend on previous occurrences within the event tree. An example of a Type 1 question in ETPRE input format is shown below. This is Question 1 taken from Example Problem 1 in Appendix A.

```
Do I get to the airport on time?
  Airport_OT      No_Airport
  0.99            0.01
  0.95            0.05
  0.75            0.25
```

As described above, the question text occupies the first line of the question definition with branch labels following on the second line. In the example shown above, the branch point probabilities must sum to unity (1.0) for each sequence defined. It is not necessary to specify the number of branches and associated probabilities; however, ETPRE will expect a probability value for each branch label found in the question. In this example, a row of split fractions is supplied for each sequence defined in the Tree Definition Header. This is necessary since the fraction values differ in each sequence. Generally speaking, only the row for the first sequence is required. Split fraction values will be assumed from the preceding sequence if a row of values for any trailing sequence is omitted.

The user can also select a method for validating the supplied split fraction values. For example, replacing the second line of the above question with the following line will instruct ETPRE to allow split fractions that sum to less than 1.0.

```
PROB<1      Airport_OT      No_Airport
```

There are three different methods for validating user-supplied split fraction values. The details of these methods are described in Section 3.3.3 beginning on Page 24.

3.3.3.2 Type 2 Question

A Type 2 question is one whose split fractions depend on which branches have been taken as results of previous questions within the event tree. An example of a Type 2 question in ETPRE input format is shown below. This is Question 3 taken from Example Problem 1 in Appendix A.

```
Does the connecting flight leave on time?
  Plane_Late + Plane_NA      Conn_OT      Conn_Late      Conn_NA
  0.00                      0.00          0.00          1.00
  0.00                      0.00          0.00          1.00
  0.00                      0.00          0.00          1.00
  Otherwise                  0.90          0.10          0.00
  0.80                      0.20          0.00
  0.33                      0.67          0.00
```

The question shown has three branches and two cases defined. A case is a set of probabilities that will be selected if a certain combination of previous branches have been taken. In the above example, Case 1 is selected if either of the branches "Plane_Late" or "Plane_NA" were taken previously. If neither of these branches were taken, then the "Otherwise" case is selected by default. The last case is always treated as the "Otherwise" case by ETPRE; even if a Boolean expression is detected, it will be ignored and the "Otherwise" keyword will be used for the last case in a multiple-case question. A diagnostic is written to the Warning Message File if this condition is detected. As illustrated, the terms of an ETPRE Boolean expression (for Question Types 2, 4, 6, and 8) consist of branch labels which must be previously defined in other questions. General information regarding Boolean expressions can be found on Page 19 of this manual.

The above question is taken from a three sequence event tree. In case Number 1, the branch split fractions are identical for all sequences. If this is the desired input, an alternative syntax can be used. The question that follows is the functional equivalent of the Type 2 question shown above. Both questions will produce identical output when processed with ETPRE.

Does the connecting flight leave on time?			
	Conn_OT	Conn_Late	Conn_NA
Plane_Late + Plane_NA	0.00	0.00	1.00
Otherwise	0.90	0.10	0.00
	0.80	0.20	0.00
	0.33	0.67	0.00

As illustrated, branch split fraction values have been omitted for sequences two and three of case Number 1. In general, a row of values is optional for all but the first sequence. Only trailing sequences can be omitted and their values will be assumed from the last defined sequence.

The user can also select a method for validating the supplied split fraction values. For example, replacing the second line of the above question with the following line will instruct ETPRE to allow split fractions that sum to less than 1.0.

PROB<1	Conn_OT	Conn_Late	Conn_NA
--------	---------	-----------	---------

There are three different methods for validating user-supplied split fraction values. The details of these methods are described in Section 3.3.3 beginning on Page 24.

3.3.3.3 Type 3 Question

As in the Type 1 question, the Type 3 question inputs the branch point probabilities that are independent of previous branches. In addition, it supplies one or more parameters which will be used in a later question for internal calculation of branch

point probabilities (Question Types 5, 6, 7, or 8). An example of a Type 3 question in the ETPRE input format is shown below. This is Question 1 taken from Example Problem 2 in Appendix B.

```

At which lake does the incident occur?
      Lake1  Lake2  Lake3  Lake4
0.25  0.25  0.25  0.25
Variable: Notification
      7      7      2      2
Variable: Travel-Time
      2      3      5      8

```

The question shown defines four branches; it also defines two variables (referred to as parameters in EVNTRE). The variables are assigned labels of "Notification" and "Travel-Time" in order of occurrence. Each variable must also supply a value for each branch defined in the question; a row of values is required for each sequence in the event tree. The keyword "Variable:" and the label that follows are both case-insensitive. The variable label must not contain spaces or tab characters and should be 1 to 16 characters in length; truncation will occur without warning if this limit is exceeded. The label must also be unique among all other variable labels in the event tree.

The user can also select a method for validating the supplied split fraction values. For example, replacing the second line of the above question with the following line will instruct ETPRE to allow split fractions that sum to less than 1.0.

```

PROB<1      Lake1  Lake2  Lake3  Lake4

```

There are three different methods for validating user-supplied split fraction values. The details of these methods are described in Section 3.3.3 beginning on Page 24.

3.3.3.4 Type 4 Question

The Type 4 question is similar to Type 2 in that branch point probabilities depend on previous branches. However, it is also similar to Type 3 since the user supplies variables (reference parameters) to be used later (in Question Types 5, 6, 7, or 8) for internal calculation of branch point probabilities. An example of a Type 4 question in ETPRE input format is shown below. This is Question 2 taken from Example Problem 2 in Appendix B.

```

How do possible unusually cold temperatures hamper the response effort?
      Usual  Extra-Cold
Lake1 + Lake2  0.90  0.10
Otherwise      0.90  0.10
Variable: Cold-Time
Lake1 + Lake2  0      1
Otherwise      0      2

```

The question shown has two branches and two cases defined. As shown, each row of probabilities or variable values is preceded by a Boolean expression. General

information regarding Boolean expressions can be found on Page 19 of this manual. In Version 3.xx of ETPRE, the user must repeat (as shown) the Boolean expression for each variable using the same order. If additional variables were required in the above question, lines 5 through 7 would be repeated with different variable names and values.

The user can also select a method for validating the supplied split fraction values. For example, replacing the second line of the above question with the following line will instruct ETPRE to allow split fractions that sum to less than 1.0.

```
PROB<1      Usual   Extra-Cold
```

There are three different methods for validating user-supplied split fraction values. The details of these methods are described in Section 3.3.3 beginning on Page 24.

3.3.3.5 Type 5 Question

A Type 5 question is one whose split fractions are calculated based on the variables defined in previous questions, but are not directly influenced by the outcomes of previous questions in the tree (i.e., a Type 5 question has no Boolean expressions). An example of a Type 5 question in ETPRE input format is shown below. This is Question 4 taken from Example Problem 2 in Appendix B.

```
Does the victim survive?
                                Death   Survival
Normal Add( Notification, Travel-Time, Cold-Time, Rescue-Time )
      16      4
```

This Type 5 question will use the Add combination type to sum the values of four variables (referred to as reference parameters in the EVNTRE manual). These variables (identified as "Notification", "Travel-Time", "Cold-Time", and "Rescue-Time") must be defined in one or more preceding questions. The question has the required two branches defined in order to use the normal comparison method. All supported comparison methods are described in detail starting on Page 21 of the EVNTRE Reference Manual. The last line of this question definition specifies the comparison parameters for this method. The first parameter (16) is the mean value of the normal distribution; the second parameter (4) is the standard deviation. All Type 5 through 8 questions supply the required information to calculate branch split fractions. Additional information describing the syntax of these question types can be found on Page 20 of this manual.

3.3.3.6 Type 6 Question

A Type 6 question is essentially the same as a Type 5 question with one notable exception: it has multiple cases defined. This is the only difference between a Type 5 and Type 6 question. An example of a Type 6 question in ETPRE input format is shown below. This is Question 5 taken from Example Problem 3 in Appendix C.

```

Does the victim survive?
      Death      Survival
Normal Add( Notification, Travel-Time, Cold-Time, Rescue-Time )
Child      21      5

Normal Add( Notification, Travel-Time, Cold-Time, Rescue-Time )
Adult      16      4

Normal Add( Notification, Travel-Time, Cold-Time, Rescue-Time )
Otherwise   10      3

```

Although not required, the question above specifies that the same comparison method, combination type, and variable names are to be used for each case. The user can specify different variable names, combination types, and comparison methods for each case if desired. If no change in methodology (variables, combination types, and comparison methods) is required, an alternative syntax can be used. The question shown below is an abbreviated form of the Type 6 question shown above. The two questions are functionally equivalent and will produce identical output when processed by ETPRE.

```

Does the victim survive?
      Death      Survival
Normal Add( Notification, Travel-Time, Cold-Time, Rescue-Time )
Child      21      5
Adult      16      4
Otherwise   10      3

```

The "function line" (Line 3 in the question above) must be specified for the first case. A new "function line" is required only if the methodology differs in the case that follows. Omission of the "function line" will instruct ETPRE to use the methodology of the preceding case. Additional information describing the syntax details of Type 5 through 8 questions can be found on Page 20 of this manual.

3.3.3.7 Type 7 Question

The Type 7 question is a Type 5 question that defines one or more variables to be used in subsequent questions. This difference is the only difference between a Type 5 and Type 7 question. An example of a Type 7 question in ETPRE input format is shown below. This is Question 5 taken from Example Problem 4 in Appendix D.

```

Does an explosion occur?
      Big-Boom      Explosion      Big-Fire      Small-Fire      NoFire
GEThresh Add ( Pump1-Flow, Pump2-Flow, Pump3-Flow, Pump4-Flow )
              70.0      50.0      25.0      10.0

Variable: Fire-Temp      $ Fire temperature, degrees Kelvin
      1000      900      800      600      293

```

The portion that determines split fractions (Lines 1 through 4 in the example above) uses the same syntax as a Type 5 question. Variables are defined in the same manner as in a Type 3 question (last 2 lines of the example above). Additional information

describing the syntax of these question types can be found on Page 20 of this manual. Syntax details regarding the definition of a variable are explained on Page 27 in Section 3.3.3.3.

3.3.3.8 Type 8 Question

The Type 8 question is a Type 6 question that defines one or more variables to be used in subsequent questions. This difference is the only difference between a Type 6 and Type 8 question. An example of a Type 8 question in ETPRE input format is shown below. This is Question 8 taken from Example Problem 5 in Appendix E.

```

Does he make his budget?
                                LT-Save   Budget-OK   Budget-NotOK   LT-Borrow

GEThresh Add( Client1, Client2, Client3, Client4, Client5, Client6 )
High-Pay                                4300   4000           3700

GEThresh Add( Client1, Client2, Client3, Client4, Client5, Client6 )
Nominal-Pay                             3900   3600           3300

GEThresh Add( Client1, Client2, Client3, Client4, Client5, Client6 )
Otherwise                              3300   3000           2700

Variable: Interest-Rate
High-Pay      8.0           8.5       9.5           11.0
Nominal-Pay   8.5           9.0      10.0          12.0
Otherwise     9.5          10.0     11.0          13.0

```

Like the Type 6 question described on Page 29, the Type 8 question can also use an abbreviated format if the methodology does not change with each case. Additional information describing the syntax of these question types can be found on Page 20 of this manual. Syntax details regarding the definition of a variable are explained on Page 27 in Section 3.3.3.3.

3.3.4 Binning Information Segment

As stated in Section 2.1, the binning segment of the Tree Definition File supplies the definitions of the binning characteristics and the corresponding Boolean expressions. The binning information segment is mandatory and must immediately follow the question definition segment in the ETPRE Tree Definition File. The following is an example of a typical binning information segment. This segment was taken from Example Problem 6 in Appendix F.

```

Binning: Gas station example
'Pump1 = Pump 1 Condition'
1 'Pump1-OK = OK'      Pump1-OK
2 'Pump1-Bad = Hit'    Pump1-Bad
'Pump2 = Pump 2 Condition'
1 'Pump2-OK = OK'      Pump2-OK
2 'Pump2-Bad = Hit'    Pump2-Bad
'Pump3 = Pump 3 Condition'
1 'Pump3-OK = OK'      Pump3-OK
2 'Pump3-Bad = Hit'    Pump3-Bad

```

```

'Pump4 = Pump 4 Condition'
1 'Pump4-OK = OK'      Pump4-OK
2 'Pump4-Bad = Hit'    Pump4-Bad
Result
1 'Explosion = Explode' Explosion
2 'Fire = Fire'        Fire

```

As shown, the start of the binning segment (or end of the question definition segment) is defined by the presence of the keyword "Binning:". This keyword is case-insensitive and may begin in Column 1, although this is not required. An optional binning identifier (like a binning title) follows the keyword on the same line. This identifier has a limit of 70 characters and must be preceded by at least one space or tab character.

Users should refer to the EVNTRE Reference Manual for detailed definitions of characteristics, outcomes, and bins. However, briefly stated, characteristics are the topics for which the user wants EVNTRE to save information, and the outcomes are the possible values that the topic can take on. It is as if a bin characteristic is a "summary question" for the event tree, and the list of outcomes is the set of possible answers to that question.

The above example defines five binning characteristics; Pump1, Pump2, Pump3, Pump4 and Result, and each characteristic has two possible outcomes. As illustrated, the first four characteristics have been assigned an optional long descriptor that will be subsequently used in the PSTEVNT code. The labels and long descriptors are defined using any of the three syntax forms shown below.

```

Syntax 1:  'label = long_descriptor'
Syntax 2:  'label'
Syntax 3:  label

```

The characteristic labels consists of 1 to 16 case-insensitive characters and must begin with an alpha character. In all three syntax forms, *label* may not contain tabs or spaces. Each characteristic label must also be unique within the tree. The *long_descriptor* is a string of 1 to 80 alphanumeric or special characters that may include spaces, tabs or double quotes. In the second and third syntax forms, the *long_descriptor* will default to *label* converted to upper case characters. If a *long_descriptor* is specified using Syntax 1, ETPRE will not perform a case conversion when the Rebinning Definition File is written.

Each characteristic in the example Binning Information Segment defines two binning outcomes. As illustrated, the binning order for each outcome is supplied in integer format. Fixed or floating point formats are also acceptable, but ETPRE will convert the binning orders to integer values for internal usage.

The binning outcome label and optional long descriptor follow the binning order. These two parameters follow the same syntax rules (defined above) that respectively describe the characteristic label and its associated long descriptor.

The last item found on the outcome line is a Boolean expression that defines the outcome. The expression contains references to one or more branch labels (which must be previously defined in the tree definition segment) with operators where required. A detailed explanation of Boolean expressions can be found on Page 19 of this manual.

Users should be aware that the presence of keyword ADDSEQ in the Keyword Input File will instruct ETPRE to alter the translation of binning information to EVNTRE format. Example Problem 6 in Appendix F uses this keyword and illustrates the resulting modifications. All modifications resulting from the use of ADDSEQ are explained in detail on Page 13 of this manual.

3.3.5 Sorting Information Segment

The sorting segment of the Tree Definition File defines the number of sorts and the sort keys associated with each one. The sorting information segment is optional, but if present, must immediately follow the binning information segment in the ETPRE Tree Definition File. The following is an example of a typical sorting information segment. This segment was taken from Example Problem 6 in Appendix F.

```

Sorting:
      Pump1   Pump2   Pump3   Pump4   Result
      Result

```

As shown, the start of the sorting segment (or end of the binning information segment) is defined by the presence of the keyword "Sorting:". This keyword is case-insensitive and may begin in Column 1, although this is not required.

This example defines two sorts. The first sort has five sort keys specified; Pump1, Pump2, Pump3, Pump4 and Result. The second sort, however, specifies only the single key Result. The first key encountered in a line is the primary key; the second key becomes the secondary key, and so on. A sort key is not valid unless it references a defined characteristic in the binning information segment and observes all the rules pertaining to characteristic labels. These rules are explained in Section 3.3.4 which begins on Page 31 of this manual. (This example is a valid sorting input based on the example binning input described in Section 3.3.4).

Since the sorting information segment is optional, ETPRE will assume a default sort if sorting information is not explicitly provided. This default will consist of a single sort having the same number of keys as characteristics defined in the binning information segment. The effect of the default sort for Example Problem 6 in Appendix F, is illustrated when ETPRE automatically generates sorting information for EVNTRE just as if the following sorting information segment had been provided.

```

Sorting:
      Pump1   Pump2   Pump3   Pump4   Result

```

The user has two methods of using the default sorting option. The first is to place the keyword "Rebinning:" after the keyword "Sorting:", without supplying any sorting information. The second method is to omit the keyword "Sorting:" entirely, and simply follow the binning information segment with whatever information comes next (the rebinning segment, the dataset segment, or the end of the file).

The user should note that no sorting information is generated in the Binning and Sorting Definition File for Example Problem 6. This is due to the presence of Keyword NOSORT in the Keyword Input File DEMO.EK1. However, using NOSORT will not suppress any sorting information written to the Rebinning Definition File that will be subsequently read by the PSTEVNT code. The details regarding the usage of NOSORT are explained on Page 15.

Users should also be aware that the presence of keyword ADDSEQ in the Keyword Input File will instruct ETPRE to alter the translation of sorting information to EVNTRE format. Example Problem 6 in Appendix F uses this keyword and illustrates the resulting modifications. All modifications resulting from the use of ADDSEQ are explained in detail on Page 13 of this manual.

3.3.6 Rebinning Information Segment

The rebinning segment of the tree definition file supplies the definitions of the rebinning characteristics and the corresponding Boolean expressions. The rebinning information segment is optional and, if specified, must immediately follow the sorting information segment in the ETPRE tree definition input file. The following is an example of a typical rebinning information segment (based on the example binning input from Section 3.3.4):

```
Rebinning:
  Result
  1 'Explosion = Explode'   Explosion
  2 'Fire = Fire'          Fire
```

As shown, the start of the rebinning segment (or end of the sorting information segment) is defined by the presence of the keyword "Rebinning:". This keyword is case-insensitive and may begin in Column 1, although this is not required. An optional rebinning ID (like a rebinning title) may follow the keyword "Rebinning:" on the same line. This identifier has a limit of 70 characters and must be preceded by at least one space or tab characters.

The rebinning segment format is identical to that of the binning information segment with one noteworthy exception. The terms (operands) of the Boolean expression are not references to defined branch labels, but references to defined binning outcome labels; they refer to binning outcomes previously defined in the binning information segment. A fatal error diagnostic will be written to file ETPRE.ERR if an undefined

binning outcome label is detected in a rebinning Boolean expression. All other components of the rebinning information segment must adhere to the same rules that apply to their counterparts in the binning information segment. These rules are explained in Section 3.3.4 which begins on Page 31 of this manual.

Quite often, it is desired to perform rebinning that does not modify the binning that was done by the EVNTRE code. In a scenario such as this, the number of rebinning characteristics is the same as the number of binning characteristics and so on. Since this is a frequently chosen method of rebinning, it has been implemented as the default rebinning method assumed by ETPRE. The default is assumed whenever the keyword "Rebinning:" is not followed by rebinning information, or is omitted entirely. The first five example problems in Appendices A through E use the default rebinning method. It should be noted that if the user supplies rebinning input and omits a characteristic from the original EVNTRE rebinning, then that characteristic is omitted from the rebinning process.

Users should be aware that the presence of keyword ADDSEQ in the file KEYWORD.DAT will instruct ETPRE to alter the translation of rebinning information to PSTEVNT input format. Example Problem 6 in Appendix F uses this keyword and illustrates the resulting modifications. All modifications resulting from the use of ADDSEQ are explained in detail on Page 13 of this manual.

3.3.7 Dataset Information Segment for LHS Sampling

The "Dataset:" Information Segment is the second of two methods for defining data items that are related to LHS sampling. As you may recall, the first method is the "Data:" statement that is described in Section 3.3.1.7 beginning on Page 21 of this manual. An example of a "Dataset:" Information segment can be found in this section and also in Example Problem 6 in Appendix F.

The "Dataset:" Information Segment is a multi-line information block consisting of the file segment marker "Dataset:" followed by statements defining one or more data items related to the LHS code. The following is an example of a "Dataset:" Information Segment.

```
Dataset:
  Gas-Flow-2  20.0  File D:\ETPRE\EXAMPLE\LHS.LSP
  Gas-Flow-3  20.0  File D:\ETPRE\EXAMPLE\LHS.LSP
  CD  0.2  Normal  0.2  0.03
```

This "Dataset:" Information Segment is taken from the Tree Definition File of Example Problem 6 in Appendix F. Although not used in this example, comment lines and continuation lines are both supported in this segment.

Syntactically, the statements that define the LHS symbols are nearly identical to the "Data:" statements described on Page 21. The only difference is the absence of the

token "Data:" at the beginning of each statement. The token "Data:" is not required (or accepted) here because the file marker "Dataset:" defines the beginning of this segment of the Tree Definition File. Apart from this single notable exception, the syntax of these statements is defined in Section 3.3.1.7 beginning on Page 21.

The "Dataset:" Information Segment may define as many LHS symbols as required within the current limits of ETPRE (see Table 1 on Page 11). However, the user must be aware that ETPRE does not differentiate in any manner LHS symbols defined in the "Dataset:" Information Segment from those defined in preceding "Data:" statements. The two methods are provided only to offer flexibility in defining the event tree. In a large event tree, it may be convenient to define a symbol in the question and case that is most relevant. Some users may prefer to place all LHS symbols in the "Dataset:" Information Segment. A combination of the two methods is also permitted and highly recommended. The only restriction regarding the "Dataset:" Information Segment is that it must be the last segment in the Tree Definition Input File.

3.3.8 Pointers and Pointer References

Pointers provide the user a shorthand method of referencing a previously defined row of branch split fractions or comparison parameters that have been identified (labeled) for this purpose. The following Type 2 question (not found in any of the Appendices) defines two pointers in Case 1 and references them in Case 5.

How many other ships are there in the immediate area?				
	Ships-0	Ships-1	Ships-M	
Port * I-Collisn	0.0000	0.0010	0.9990	!repeat1
	0.0000	0.0010	0.9990	!repeat2
Port * NOT I-Collisn	0.0050	0.0050	0.9900	
	0.0050	0.0050	0.9900	
Restricted * I-Collisn	0.0000	0.0100	0.9900	
	0.0000	0.0100	0.9900	
Restricted * NOT I-Collisn	0.1000	0.1000	0.8000	
	0.1000	0.1000	0.8000	
Open_Sea * I-Collisn	!repeat1			
	!repeat2			
Open_Sea * NOT I-Collisn	0.5000	0.2500	0.2500	
	0.5000	0.2500	0.2500	

As illustrated, the pointer consists of an exclamation point followed by 1 to 16 case-insensitive alphanumeric characters. Although shown here, it is not a requirement that the pointer reference be made in the same question in which it is defined. However, the reference must be made in the same sequence in which it was defined. Also, the number and type of values must agree if the reference is not made within the same question. As shown, the pointer definition follows the last value in the row; the pointer reference replaces all values for the row in which it is found. Each pointer defined must be unique, but can be referenced as many times as necessary. Unreferenced pointers will be brought to the attention of the user by a message written to the warning file ETPRE.WNG. If any unreferenced pointers are reported, they should be removed since they are counted against the limit of defined pointers.

Users familiar with the EVNTRE code input format may be interested in knowing that the ETPRE pointer reference is translated to a series of EVNTRE pointers. The following question shows the preceding question after it has been translated from ETPRE to EVNTRE format:

```

5 How many other ships are there in the immediate area?
3 SHIPS-0 SHIPS-1 SHIPS-M
2
6
2      2      1
      1      * 2
      PORT      I-COLLISN
      0.00000E+00 1.00000E-03 9.99000E-01
      0.00000E+00 1.00000E-03 9.99000E-01
2      2      1
      1      * -2
      PORT      I-COLLISN
      5.00000E-03 5.00000E-03 9.90000E-01
      5.00000E-03 5.00000E-03 9.90000E-01
2      2      1
      2      * 2
      RESTRICTED I-COLLISN
      0.00000E+00 1.00000E-02 9.90000E-01
      0.00000E+00 1.00000E-02 9.90000E-01
2      2      1
      2      * -2
      RESTRICTED I-COLLISN
      1.00000E-01 1.00000E-01 8.00000E-01
      1.00000E-01 1.00000E-01 8.00000E-01
2      2      1
      3      * 2
      OPEN_SEA   I-COLLISN
      15 1 1      15 1 2      15 1 3
      15 1 1      15 1 2      15 1 3
      OTHERWISE
      5.00000E-01 2.50000E-01 2.50000E-01
      5.00000E-01 2.50000E-01 2.50000E-01

```

3.4 LHS-like Sampling Information Files

ETPRE supports the sampling capabilities of the EVNTRE code by using the LHS (Latin Hypercube Sampling) program. LHS is used to perform a simulation that generates an output file containing sampling results for each iteration (referred to as an observation in the EVNTRE Reference Manual) of the simulation. The values corresponding to an LHS symbolic name defined as an LHS distribution will be found in the LHS Output File. These symbolic names and distributions must be defined using Syntax 2 of the "Data:" statement or "Dataset:" segment of the Tree Definition File. This syntax is described on Page 23. The name of the LHS Output File is specified by keyword LHSOUT in the Keyword Input File. Use of the keyword LHSOUT is explained on Page 14.

The LHS postprocessor program can also be used to evaluate LHS expressions defined in the ETPRE Tree Definition File. The postprocessor output file is an LHS-like file that uses the same format and syntax. The LHS postprocessor reads the ETPRE Tree Definition File and the LHS output file, evaluates all LHS expressions, then writes

these values to an LHS-like file whose name is specified by keyword LHSPPOST in the Keyword File. Use of the keyword LHSPPOST is explained on Page 14. The user must be aware that an LHS expression is defined in the "Data:" statement or in the "Dataset:" segment of the Tree Definition File. LHS expressions must conform to Syntax 3 of the "Data:" statement (or "Dataset:" segment) described on Page 23.

Additionally, the user may specify up to 16 other LHS-like files from which ETPRE must read sampling information. These files (or file paths) are also specified in the "Data:" statements or in the "Dataset:" segment of the Tree Definition File. An LHS symbolic name whose sampling values are to be read from one of these "User" files, must be specified using Syntax 1 of the "Data:" statement (or "Dataset:" segment) described on Page 22.

The user is not generally concerned with examining any of the LHS-like sampling files since ETPRE will automatically read them and generate the two files (sample definition and sample vector) that are required as input to EVNTRE if sampling is to be performed. However, the LHS-like sampling information files required in Example Problem 6 are listed in Appendix F for user information or reference. The exact format of an LHS-like file is defined in Appendix G.

In the process of defining an event tree, the user must inform ETPRE which split fractions or parameter values are to be sampled. Specifying sampling is accomplished by enclosing an LHS symbolic name in parentheses as shown in the following question. A fatal error diagnostic will be issued if any LHS sampled symbolic name has not been defined in a "Data:" statement or in the "Dataset:" file segment.

```
Is pump #1 Damaged?
  Pump1-OK      Pump1-Bad
    (A)          (Excess)
Variable: Pump1-Flow
  0.0           (Gas-Flow-1)

Data: Gas-Flow-1 20.0 File D:\ETPRE\EXAMPLE\LHS.LSP
Data: A1 0.6 LogUniform 0.4 0.8
Data: Crash-Factor 1.0 Lognormal-B 0.8 1.2
Data: A 0.6 = A1 * Crash-Factor
```

This is Question 1 taken from Example Problem 6 in Appendix F. For the LHS symbolic name "A" in the third line above, corresponding values for each observation will be read from the LHS postprocessor output file instead of the LHS output file. This occurs because "A" is defined as an expression (in the last "Data:" statement) that requires evaluation by the LHS postprocessor. The LHS symbolic name "Excess" has special meaning. It instructs ETPRE to sample the split fraction for the second branch substituting a value of one minus the total of the fractions for all other branches. In this situation, ETPRE will calculate and substitute a different "Excess" value for each observation. If a fraction or parameter is not to be sampled for a given branch, a valid numeric constant should appear in that branch position. An example of an unsampled branch value appears in the fifth line in the example above.

There are times when the user may choose not to sample many observations, but use a single point estimate value for each LHS symbolic name that was identified for sampling (i.e., enclosed in parentheses). If this is desired, the user need only change the "SAMMOD" value in the Keyword File to 12 or 13. The details of keyword SAMMOD are explained on Page 17 of this manual.

Another option that is sometimes necessary is to perform full sampling on some of the LHS symbolic names, but to always use point estimate values for others. To accomplish this, the user simply encloses the symbolic name in square brackets whenever the point estimate value is to be used. An example of this is shown in the question that follows.

```
$ This question samples the explosion threshold in the first two
$ sequences, and uses a constant threshold for the third sequence.

How big is the spill?
      Large      Small      No-Spill
GEThresh Add(Pump1-Flow, Pump2-Flow, Pump3-Flow, Pump4-Flow)
      (Exp-Thresh) 0.1
      (Exp-Thresh) 0.1
      [Exp-Thresh] 0.1
Data: Exp-Thresh 50.0 File D:\ETPRE\EXAMPLE\LHS.LSP
```

This is Question 5 taken from Example Problem 6 in Appendix F. In this problem, the point estimate value used for symbolic name "Exp-Thresh" is sampled during the evaluation of the first two sequences. For the third sequence, a point estimate value is taken from the Point Estimate (first) Section of the LHS-like file that defines this LHS symbolic name. In this particular example, the complete pathname for the file is "D:\ETPRE\EXAMPLE\LHS.LSP".

4. Output Files

4.1 Tree Definition File

The Tree Definition File written by ETPRE is the complete event tree after translation to EVNTRE input format. All syntax rules described in the EVNTRE Reference Manual are strictly enforced. Users should consult the EVNTRE Manual for syntax details. It is necessary, however, to address the issue of the required comment records in EVNTRE input. ETPRE will output these records where necessary when writing the Tree Definition File. Also, it will attempt to make these records meaningful by inserting branch labels in them. These labels refer to the terms of a Boolean expression that immediately precedes the required comment record. If the preceding expression requires more than a single EVNTRE input record to describe, ETPRE will insure that a label for each term in the expression is written by inserting optional comment records in the Tree Definition File. These records are easily identified by the '\$' found in Column 1.

Appendices A through F contain complete listings of Tree Definition Files in EVNTRE format for six example problems. These file listings are the third of several files listed in each Appendix.

4.2 Binning and Sorting Definition File

The Binning and Sorting Definition File contains information taken from the binning and sorting segments of the ETPRE Tree Definition File. However, the information is translated to a format that is acceptable as input to the EVNTRE code. The format of this file is well-described in the EVNTRE Reference Manual starting on Page 31. As in the Tree Definition File described in Section 4.1, labels are also inserted in required comments (where applicable) to describe the terms of a preceding Boolean expression.

Appendices A through F contain complete listings of Binning and Sorting Definition Files in EVNTRE format for six example problems. These files are the fourth of several files listed in each Appendix.

4.3 Rebinning Definition File

The Rebinning Definition File contains information taken from the rebinning segment of the ETPRE Tree Definition File. As previously described on Page 34, the information will assume default values if the rebinning segment is not specified. However, the information is translated to a format that is acceptable as input to the PSTEVNT code. The format of this file is described in the PSTEVNT Reference Manual starting on Page 16. As in the Tree Definition Output File described in Section 4.1, labels are also inserted in required comments (where applicable) to describe the terms of a preceding Boolean expression.

Appendices A through F contain complete listings of Rebinning Definition Files in PSTEVNT format for six example problems. These file listings are the fifth of several files listed in each Appendix.

4.4 Sample Definition File

This file supplies EVNTRE with the specifications required to set up the sampling modes for the tree. These modes include the pointers to probabilities and parameters to be sampled. When ETPRE writes the sample definition file, it uses information found in both the Tree Definition File and any required LHS-like Sampling Information Files to form the correct number and type of EVNTRE pointers. These pointers, along with the sampling definition input blocks, are written strictly in accordance with the syntax described on Pages 35 through 39 of the EVNTRE Reference Manual. Since the

Sample Definition File is normally very small, the following Sample Definition File is listed here in addition to appearing in Appendix F as File Number 6.

```
FEB 10, 1994 @12:34
15 1
13
P2 1 1 2 TTT M2 1 1 2 TTT P3 1 1 2 TTT M3 1 1 2 TTT
P4 1 1 2 TTT M4 1 1 2 TTT P5 1 1 2 TTT M5 1 1 2 TTT
C6 1 1 TFF C6 1 1 FTF P7 1 1 0 TTT P7 1 2 0 TTT
P7 2 2 0 TTT
```

This is the Sample Definition File DEMO.EPI generated by ETPRE for Example Problem 6 in Appendix F. The Sample Definition File will not be generated unless the user has identified for sampling (enclosed in parentheses) one or more LHS symbolic names and unless an LHS simulation has been previously performed, thereby producing at least one Sampling Information File that is required input for ETPRE.

4.5 Sample Vector File

This file supplies EVNTRE with the vectors whose elements are the values to be substituted into the tree for each iteration (observation) in the sample. The format of this file is the same as that generated by the original LHS code and that specified in Reference 1. The values written to the Sample Vector File are taken from the Sampling Information File produced by the LHS program. The following data illustrates a typical Sample Vector File; it is also listed in Appendix F as File 7.

1	13	3.784800E-01	10.040600	3.887540E-01
21.720500	2.332350E-01	20.361900	2.332350E-01	
8.817790	72.956900	72.956900	8.804870E-01	
1.195130E-01	4.091540E-01			
2	13	4.875210E-01	42.806900	4.636910E-01
32.923600	1.856510E-01	15.830600	1.856510E-01	
16.466600	34.946100	34.946100	9.147510E-01	
8.524900E-02	6.098890E-01			
3	13	6.356060E-01	16.082000	5.722520E-01
14.305500	2.449650E-01	12.009800	2.449650E-01	
26.127800	55.351100	55.351100	7.911870E-01	
2.088130E-01	7.328630E-01			
4	13	4.896430E-01	34.159800	5.904340E-01
19.902500	2.032290E-01	42.385600	2.032290E-01	
50.298600	74.397100	74.397100	9.690480E-01	
3.095200E-02	3.878710E-01			
5	13	6.034290E-01	21.038000	7.532000E-01
7.864010	1.413190E-01	29.076400	1.413190E-01	
11.307400	20.511300	20.511300	5.603190E-01	
4.396810E-01	5.002000E-01			
6	13	5.780570E-01	17.854400	4.432720E-01
22.525800	1.953350E-01	21.627800	1.953350E-01	
25.984100	30.720100	30.720100	6.031910E-01	
3.968090E-01	4.735270E-01			
7	13	4.361840E-01	8.015580	4.853430E-01
15.377800	1.874880E-01	14.804400	1.874880E-01	
16.070300	46.326400	46.326400	4.571050E-01	
5.428950E-01	7.054150E-01			
8	13	7.298090E-01	14.078100	5.411220E-01
10.082400	1.729140E-01	13.108300	1.729140E-01	
13.142400	234.315000	234.315000	7.348100E-01	
2.651900E-01	7.770310E-01			
9	13	4.961050E-01	23.659400	6.031240E-01

17.196400	2.017340E-01	31.785000	2.017340E-01
20.548500	51.118600	51.118600	8.389010E-01
1.610990E-01	5.638170E-01		
10	13	6.803330E-01	16.613100 7.576270E-01
18.097200	2.568540E-01	24.065800	2.568540E-01
14.553300	40.542700	40.542700	6.778050E-01
3.221950E-01	5.579320E-01		
11	13	4.830380E-01	12.793300 4.501640E-01
25.391000	1.771940E-01	22.695300	1.771940E-01
32.387700	27.208800	27.208800	4.919360E-01
5.080640E-01	2.091510E-01		
12	13	4.405000E-01	24.067800 4.696260E-01
13.676400	2.239050E-01	10.453500	2.239050E-01
19.179000	48.982000	48.982000	4.064470E-01
5.935530E-01	6.607690E-01		
13	13	6.008650E-01	20.312500 5.497220E-01
40.319900	2.161140E-01	18.789500	2.161140E-01
18.017700	60.149100	60.149100	6.655290E-01
3.344710E-01	2.583870E-01		
14	13	8.386230E-01	27.455900 7.870090E-01
27.039800	2.089160E-01	17.306700	2.089160E-01
23.278800	41.664100	41.664100	5.491960E-01
4.508040E-01	3.042730E-01		
15	13	6.038150E-01	13.706300 6.363400E-01
12.261500	1.633420E-01	9.418760	1.633420E-01
12.747300	36.558000	36.558000	8.178740E-01
1.821260E-01	3.469850E-01		

This is Sample Vector File DEMO.EVI generated by ETPRE for Example Problem 6. The Sample Vector File will not be generated unless the user has identified for sampling (enclosed in parentheses) one or more LHS symbolic names and an LHS simulation has been previously performed, thereby producing at least one Sampling Information File that is required input for ETPRE.

4.6 Error Files

Errors detected during execution are identified by diagnostics written to the files that are described in the following sections. If either or both of these files exist, they will be deleted at the beginning of every ETPRE execution. For this reason, both error files should be considered as highly volatile. This insures that any error file written applies to the last execution of ETPRE.

4.6.1 Fatal Error Message File (ETPRE.ERR)

Fatal error diagnostics will be written to a file named ETPRE.ERR. As many errors as possible will be listed in this file; however, certain problems encountered by ETPRE preclude further processing. In this scenario, correcting the malfunction will allow ETPRE to continue beyond that point and simply report the next serious problem. A complete set of output files will not be generated as long as fatal errors exist in the input files. The following is an example of a typical Fatal Error Message File.

```

***** Error No.    1 *****
      UNDEFINED BRANCH LABEL REFERENCE OR ILLEGAL SPLIT FRACTION
            AT LINE NO.    24
Steamin 1.0000  0.0000

```

As shown, the diagnostic is followed by the line number where the problem was detected, and by a copy of the offending line. For added convenience to the user, the first four fatal messages are also displayed on the computer screen just before ETPRE terminates execution. The user must examine the contents of this file if more than four fatal errors are detected. An audible tone (beep) will also be generated if one or more fatal errors are detected. ETPRE currently checks for 114 different fatal conditions. In the example above, the referenced branch label "Steamin" is obviously undefined since the split fractions are valid and sum to unity (1.0).

4.6.2 Warning Message File (ETPRE.WNG)

Warning diagnostics will be written to a file named ETPRE.WNG. The types of problems reported are not serious and will not abort execution. This file should be examined, however, since the warning messages may expose a condition not intended by the user. The following is an example of a typical Warning Message File.

```

***** Error No.    1 *****
      Warning ----- UNREFERENCED POINTER "labelBB"
            Defined at Question  8, Case  1, Sequence  2

***** Error No.    2 *****
      Warning ----- NO SORTS DEFINED; USING DEFAULT SORT

***** Error No.    3 *****
      Warning ----- NO RE-BINS DEFINED; USING DEFAULT RE-BIN

```

As shown and where applicable, the diagnostic is followed by the line number where the problem was detected, and by a copy of the offending line. The user must examine the contents of file ETPRE.WNG to view these warnings since they are not automatically displayed as are the first four fatal error messages. The audible tone (beep) is not issued if only warning diagnostics are written. ETPRE currently checks for 12 different warning conditions.

4.7 QA Information File

The QA Information File is generated only when the keyword QAOUT is encountered in the Keyword Input File. Use of this keyword is described on Page 16 of this manual. The primary use of this file was intended as a programmer's debugging tool; so generally speaking, the QA Information File will be of little value to the ETPRE user. The file lists the values of all tree-related tables (arrays) built by ETPRE in a format that makes the entire internal structure easy to understand. The QA Information File is a large file, even if the event tree has only a few questions. For this reason, an example of this file has not been provided in this manual. Of course, the file can

always be generated and printed if the user is curious about the internal structure used. A printout of the QA Information File should always be generated and presented to the author if a malfunction (bug) in ETPRE is suspected.

Generating the file simply requires the addition of a line to the keyword input file as previously stated. However, since the file contains records up to 200 characters in length, the user must set a Laserjet printer for Landscape Mode and 16.5 characters per inch before printing to avoid truncation of records.

5. Debugging Considerations

Debugging an event tree using ETPRE is straightforward. The user must observe all the syntax rules when preparing the main tree definition input file. More than one iteration of corrections and reruns may be required in order to locate and correct all fatal errors in the tree definition. Once all errors have been corrected, ETPRE will generate files that are always syntactically correct input for both the EVNTRE and PSTEVNT codes. If the user should suspect this not to be the case, the problem should be reported to the Sandia point of contact so that a suitable remedy can be applied to a subsequent maintenance release version of ETPRE.

Once the ETPRE and EVNTRE executions are successful, additional data analysis considerations may be required. These are defined in the EVNTRE Reference Manual (see Ref. 1) starting on Page 65. As previously stated, there should be no syntax errors reported by the EVNTRE code once the analysis reaches this stage.

References

1. J. Michael Griesmeyer, L. N. Smith, A Reference Manual for the Event Progression Analysis Code (EVNTRE), NUREG/CR-5174, SAND88-1607, Sandia National Laboratories, September 1989.
2. Sarah J. Higgins, A User's Manual for the Postprocessing Program PSTEVNT, NUREG/CR-5380, SAND88-2988, Sandia National Laboratories, November 1989.
3. Gregory D. Wyss, A Tutorial for the SETAC Event Tree Analysis Software, SAND94-0818, Sandia National Laboratories, April 1994.
4. Ronald L. Iman, Michael J. Shortencarier, A FORTRAN 77 Program and User's Guide for the Generation of Latin Hypercube and Random Samples for Use With Computer Models, NUREG/CR-3624, SAND83-2365, March 1984.

Appendices

Appendices A through F define six example event tree problems that illustrate the various features of ETPRE. One complete problem is contained in each appendix. The appendix begins with an example execution sequence for the problem. With the exception of error files, all ETPRE input and output files associated with each problem will be listed in the appropriate appendix. All example problems used the Microsoft Fortran version of the ETPRE executable code.

Appendix G defines the format of LHS-like Sampling Information Files that must be read by ETPRE.

Appendix A - Example Problem 1

ETPRE Execution Sequence

D:\ETPRE\EXAMPLE>ETPRE PROB1.EKI

ETPRE VER. 3.00 02/08/94
AUTHOR: Bob Roginski ORG. 12333

Run date & time = 02/08/94 15:43:05
Compiler Used = Microsoft V 5.1

***** INPUT FILES *****

NOTE: KEYWORD INPUT FILE IS PROB1.EKI
NOTE: TREE DEF. INPUT FILE IS PROB1.PRE

***** OUTPUT FILES *****

NOTE: "EVNTRE" INPUT WRITTEN TO FILE PROB1.TRE
NOTE: BINNING INPUT WRITTEN TO FILE PROB1.BIN
NOTE: PSTEVNT INPUT WRITTEN TO FILE PROB1.RBN

***** RESOURCE USAGE *****

67	TOTAL INPUT LINES READ
4	OF 200 MAX QUESTIONS USED
3	OF 32 MAX SEQUENCES USED
12	OF 600 MAX TOTAL BRANCHES USED
66	OF 2400 MAX TOTAL SPLIT-FRACTIONS USED
6	OF 600 MAX TOTAL EXPRESSIONS USED
10	OF 1800 MAX TOTAL SYMBOLS USED
0	OF 200 MAX TOTAL REF. PARAMS. USED
0	OF 2400 MAX TOTAL PARAM. VALUES USED
0	OF 200 MAX TOTAL POINTERS USED
21	OF 1800 MAX TOTAL REFERENCE TYPES USED
0	OF 1200 MAX TOTAL S.F. COMPARISON VALUES
0	OF 600 MAX TOTAL COMPARISON PARAMETERS
0	OF 600 MAX TOTAL PARAM. COMPARISON TYPES
0	OF 600 MAX TOTAL INTERNAL FUNCTION REFS.
0	OF 256 MAX TOTAL SAMPLE VECTORS
0	OF 256 MAX TOTAL VECTOR LABELS
1	OF 32 MAX TOTAL BINNING CHARACTERISTICS
3	OF 128 MAX TOTAL BINNING OUTCOMES
5	OF 384 MAX TOTAL BINNING SYMBOLS
1	OF 16 MAX TOTAL SORTS
1	OF 128 MAX TOTAL SORT KEYS
1	OF 32 MAX TOTAL RE-BIN. CHARACTERISTICS
3	OF 128 MAX TOTAL RE-BIN. OUTCOMES
3	OF 384 MAX TOTAL RE-BIN. SYMBOLS
0	OF 256 MAX TOTAL LHS SYMBOLIC NAMES
0	OF 16 MAX TOTAL LHS FILE PATHS
0	OF 512 MAX TOTAL LHS SAMPLED SYMBOLS

*** 1 MINOR ERRORS ***
(SEE FILE ETPRE.WNG)

Stop - Program terminated.

Keyword Input File PROB1.EK1 (File 1)

```
PRETRIN PROB1.PRE
MODE 1
TREEIN PROB1.TRE
BININ PROB1.BIN
STATS
PRTINP
BIN
WRTBIN
SAVEBIN
INPUT PROB1-IN.OUT
STATOUT PROB1-STA.OUT
SAMROUT PROB1-PST.OUT
BINOUT PROB1-BIN.OUT
KEEPCUT 5.0E-11
PRTCUT 5.0E-9
RUN
ENDKEY
```

ETPRE Tree Definition Input File PROB1.PRE (File 2)

\$ We use a new header block for the uncertainty analysis because we need
\$ multiple evaluations of the same event tree. Let's assume that the
\$ best estimate is 80% likely, and the others 10% likely each.

Business travel example with simple uncertainty.

Optimistic	0.10
BestGuess	0.80
Pessimistic	0.10

\$ The questions and logic are all the same now. However, we need to
\$ supply three sets of data for every event. For each event, the first
\$ set of data is used in the "optimistic" evaluation, the second in the
\$ "best guess" evaluation, and the third in the "pessimistic" evaluation.

Do I get to the airport on time?

Airport_OT	No_Airport
0.99	0.01
0.95	0.05
0.75	0.25

Does the plane leave on time?

	Plane_OT	Plane_Late	Plane_NA
-Airport_OT	0.00	0.00	1.00
	0.00	0.00	1.00
	0.00	0.00	1.00
Otherwise	0.90	0.10	0.00
	0.80	0.20	0.00
	0.33	0.67	0.00

\$ Note that if values are left out for a particular sequence, the code
\$ uses copies the values for the previous case as shown below.

Does the connecting flight leave on time?

	Conn_OT	Conn_Late	Conn_NA
Plane_Late + Plane_NA	0.00	0.00	1.00
	0.00	0.00	1.00
	0.00	0.00	1.00
Otherwise	0.90	0.10	0.00
	0.80	0.20	0.00

		0.33	0.67	0.00
Do my bags arrive undamaged?				
	Bags_OK	Damaged	Boseman	Bags_NA
-Conn_OT	0.00	0.00	0.00	1.00
	0.00	0.00	0.00	1.00
	0.00	0.00	0.00	1.00
Otherwise	0.75	0.20	0.05	0.00
	0.05	0.75	0.20	0.00
	0.01	0.75	0.24	0.00

\$ The binning input is the same regardless of how many evaluations are performed. It is shown on slide # 44 for this session.

Binning: Binning for the business trip example

Experience		
1	GoodTrip	Bags_OK
3	ReallyBad	Boseman + Bags_NA
2	BadTrip	Damaged

Sorting:

Experience

\$ Here is the end of the input for the example problem.

EVNTRE Tree Definition Input File PROBL.TRE (File 3)

Business travel example with simple uncertainty.

```

4
NQ
3 1.00000E-01 8.00000E-01 1.00000E-01
  OPTIMISTIC BESTGUESS PESSIMISTIC
1 Do I get to the airport on time?
  2 AIRPORT_OT NO_AIRPORT
    1
      9.90000E-01 1.00000E-02
      9.50000E-01 5.00000E-02
      7.50000E-01 2.50000E-01
2 Does the plane leave on time?
  3 PLANE_OT PLANE_LATE PLANE_NA
    2
      2
        1
          1
            -1
              AIRPORT_OT
                0.00000E+00 0.00000E+00 1.00000E+00
                0.00000E+00 0.00000E+00 1.00000E+00
                0.00000E+00 0.00000E+00 1.00000E+00
              OTHERWISE
                9.00000E-01 1.00000E-01 0.00000E+00
                8.00000E-01 2.00000E-01 0.00000E+00
                3.30000E-01 6.70000E-01 0.00000E+00
3 Does the connecting flight leave on time?
  3 CONN_OT CONN_LATE CONN_NA
    2
      2
        2
          2
            2
              PLANE_LATE
                0.00000E+00 0.00000E+00 1.00000E+00
                0.00000E+00 0.00000E+00 1.00000E+00
                0.00000E+00 0.00000E+00 1.00000E+00
              OTHERWISE

```

```

          9.00000E-01  1.00000E-01  0.00000E+00
          8.00000E-01  2.00000E-01  0.00000E+00
          3.30000E-01  6.70000E-01  0.00000E+00
4 Do my bags arrive undamaged?
  4 BAGS_OK DAMAGED BOSEMAN BAGS_NA
    2
    2
    1      3
        -1
      CONN_OT
        0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
        0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
        0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
      OTHERWISE
        7.50000E-01  2.00000E-01  5.00000E-02  0.00000E+00
        5.00000E-02  7.50000E-01  2.00000E-01  0.00000E+00
        1.00000E-02  7.50000E-01  2.40000E-01  0.00000E+00

```

Binning and Sorting Definition File PROB1.BIN (File 4)

```

Binning for the business trip example
1  EXPERIENCE
3  3  GOODTRIP      BADTRIP      REALLYBAD
1  1      4
    1
    BAGS_OK
2  3      4      4
    3      +      4
    BOSEMAN      BAGS_NA
1  2      4
    2
    DAMAGED
1
1      1
    EXPERIENCE

```

Rebinning Definition File PROB1.RBN (File 5)

```

Binning for the business trip example
1  EXPERIENCE
3  3  GOODTRIP      BADTRIP      REALLYBAD
1  1      1
    1
    GOODTRIP
1  3      1
    2
    REALLYBAD
1  2      1
    3
    BADTRIP
1
1      1
    EXPERIENCE
1
3
1  'GOODTRIP'
2  'REALLYBAD'
3  'BADTRIP'

```

```

$
$ This file was written on 02/08/94 at 15:43:05 by
$ ETPRE Version 3.00, compiled with Microsoft V 5.1.

```

```

$ Tree Definition File is PROB1.PRE.
$ Keyword File is PROB1.EK1.
$
@BIN
$$ EVNTRE/PSTEVNT Binning Labels Version 1.00
$$ Title Binning for the business trip example
$$ Dimensions: 1
$$ Dim 1 3 EXPERIENCE
$$ 1 A GOODTRIP
$$ 1 B BADTRIP
$$ 1 C REALLYBAD
$$ Done
@REBIN
$$ EVNTRE/PSTEVNT Binning Labels Version 1.00
$$ Title Binning for the business trip example
$$ Dimensions: 1
$$ Dim 1 3 EXPERIENCE
$$ 1 A GOODTRIP
$$ 1 B BADTRIP
$$ 1 C REALLYBAD
$$ Done

```

Appendix B- Example Problem 2

ETPRE Execution Sequence

D:\ETPRE\EXAMPLE>ETPRE PROB2.EKI

ETPRE VER. 3.00 02/08/94
AUTHOR: Bob Roginski ORG. 12333

Run date & time = 02/08/94 15:43:27
Compiler Used = Microsoft V 5.1

***** INPUT FILES *****

NOTE: KEYWORD INPUT FILE IS PROB2.EKI
NOTE: TREE DEF. INPUT FILE IS PROB2.PRE

***** OUTPUT FILES *****

NOTE: "EVNTRE" INPUT WRITTEN TO FILE PROB2.TRE
NOTE: BINNING INPUT WRITTEN TO FILE PROB2.BIN
NOTE: PSTEVNT INPUT WRITTEN TO FILE PROB2.RBN

***** RESOURCE USAGE *****

59 TOTAL INPUT LINES READ
4 OF 200 MAX QUESTIONS USED
1 OF 32 MAX SEQUENCES USED
11 OF 600 MAX TOTAL BRANCHES USED
11 OF 2400 MAX TOTAL SPLIT-FRACTIONS USED
2 OF 600 MAX TOTAL EXPRESSIONS USED
4 OF 1800 MAX TOTAL SYMBOLS USED
4 OF 200 MAX TOTAL REF. PARAMS. USED
15 OF 2400 MAX TOTAL PARAM. VALUES USED
0 OF 200 MAX TOTAL POINTERS USED
10 OF 1800 MAX TOTAL REFERENCE TYPES USED
2 OF 1200 MAX TOTAL S.F. COMPARISON VALUES
5 OF 600 MAX TOTAL COMPARISON PARAMETERS
1 OF 600 MAX TOTAL PARAM. COMPARISON TYPES
1 OF 600 MAX TOTAL INTERNAL FUNCTION REFS.
0 OF 256 MAX TOTAL SAMPLE VECTORS
0 OF 256 MAX TOTAL VECTOR LABELS
2 OF 32 MAX TOTAL BINNING CHARACTERISTICS
6 OF 128 MAX TOTAL BINNING OUTCOMES
6 OF 384 MAX TOTAL BINNING SYMBOLS
1 OF 16 MAX TOTAL SORTS
2 OF 128 MAX TOTAL SORT KEYS
2 OF 32 MAX TOTAL RE-BIN. CHARACTERISTICS
6 OF 128 MAX TOTAL RE-BIN. OUTCOMES
6 OF 384 MAX TOTAL RE-BIN. SYMBOLS
0 OF 256 MAX TOTAL LHS SYMBOLIC NAMES
0 OF 16 MAX TOTAL LHS FILE PATHS
0 OF 512 MAX TOTAL LHS SAMPLED SYMBOLS

*** 1 MINOR ERRORS ***
(SEE FILE ETPRE.WNG)

Stop - Program terminated.

Keyword Input File PROB2.EKI (File 1)

```
PRETRIN PROB2.PRE
MODE 1
TREEIN PROB2.TRE
BININ PROB2.BIN
STATS
PRTINP
BIN
WRTBIN
SAVEBIN
INPOUT PROB2-IN.OUT
STATOUT PROB2-STA.OUT
SAMROUT PROB2-PST.OUT
BINOUT PROB2-BIN.OUT
KEEPCUT 5.0E-11
PRTCUT 5.0E-9
RUN
ENDKEY
```

ETPRE Tree Definition Input File PROB2.PRE (File 2)

Fire Department Rescue of a Person Falling Through the Ice
Only-Eval 1.00

\$ Determine which lake the incident occurs at, and set variables for how
\$ long it takes to get to the lake and notification time. The lakes are
\$ of approximately equal size and usage, so the incidents are about equally
\$ likely at each lake. Each variable is in minutes.

At which lake does the incident occur?

	Lake1	Lake2	Lake3	Lake4
	0.25	0.25	0.25	0.25
Variable: Notification				
	7	7	2	2
Variable: Travel-Time				
	2	3	5	8

\$ If the temperature is unusually cold, it takes longer to respond. Extra time
\$ is different for the various lakes, so model this using a "type 4 question" in
\$ which the variable value depends on the previously occurring branches.

How do possible unusually cold temperatures hamper the response effort?

	Usual	Extra-Cold
Lake1 + Lake2	0.90	0.10
Otherwise	0.90	0.10
Variable: Cold-Time		
Lake1 + Lake2	0	1
Otherwise	0	2

\$ The time it takes to perform the rescue depends only on where in the lake
\$ the person is. There is no dependence on previous questions.

Where in the lake did the person fall through the ice?

	Near-Shore	Int-Dist	Far-Out
	0.50	0.30	0.20
Variable: Rescue-Time			
	1	3	5

\$ Now total the times and compare to a normal distribution with mean 16
\$ standard deviation 4, to determine the likelihood of survival.

Does the victim survive?

	Death	Survival
Normal Add(Notification, Travel-Time, Cold-Time, Rescue-Time)		

Binning: Binning input for the ice rescue problem

Site		
1	Lake-1	Lake1
2	Lake-2	Lake2
3	Lake-3	Lake3
4	Lake-4	Lake4

Outcome		
1	Success	Survival
2	Failure	Death

Sorting:

Outcome	Site
---------	------

EVNTRE Tree Definition Input File PROB2.TRE (File 3)

Fire Department Rescue of a Person Falling Through the Ice

```

4
NQ
1 1.00000E+00
  ONLY-EVAL
1 At which lake does the incident occur?
4 LAKE1 LAKE2 LAKE3 LAKE4
3
  2.50000E-01 2.50000E-01 2.50000E-01 2.50000E-01
2
1 7.00000 7.00000 2.00000 2.00000
2 2.00000 3.00000 5.00000 8.00000
2 How do possible unusually cold temperatures hamper the response effort?
2 USUAL EXTRA-COLD
4
2
2 1 1
  1 + 2
    LAKE1 LAKE2
    9.00000E-01 1.00000E-01
1
3 .000000 1.00000
  OTHERWISE
  9.00000E-01 1.00000E-01
1
3 .000000 2.00000
3 Where in the lake did the person fall through the ice?
3 NEAR-SHORE INT-DIST FAR-OUT
3
  5.00000E-01 3.00000E-01 2.00000E-01
1
4 1.00000 3.00000 5.00000
4 Does the victim survive?
2 DEATH SURVIVAL
5
4 1 2 3 4
  NOTIFICATION TRAVEL-TIME COLD-TIME RESCUE-TIME
ADD
NORMAL 2 16.0000 4.00000
REQUIRED COMMENT FOR QUESTION 4

```

Binning and Sorting Definition File PROB2.BIN (File 4)

Binning input for the ice rescue problem

2		SITE		OUTCOME		
4	4	LAKE-1		LAKE-2	LAKE-3	LAKE-4
1	1	1				
		LAKE1				
1	2	1				
		LAKE2				
1	3	1				
		LAKE3				
1	4	1				
		LAKE4				
2	2	SUCCESS		FAILURE		
1	1	4				
		SURVIVAL				
1	2	4				
		DEATH				
1						
2		2		1		
		OUTCOME		SITE		

Rebinning Definition File PROB2.RBN (File 5)

Binning input for the ice rescue problem

2		SITE		OUTCOME		
4	4	LAKE-1		LAKE-2	LAKE-3	LAKE-4
1	1	1				
		LAKE-1				
1	2	1				
		LAKE-2				
1	3	1				
		LAKE-3				
1	4	1				
		LAKE-4				
2	2	SUCCESS		FAILURE		
1	1	2				
		SUCCESS				
1	2	2				
		FAILURE				
1						
2		2		1		
		OUTCOME		SITE		
1						
4						
1		'LAKE-1'				
2		'LAKE-2'				
3		'LAKE-3'				
4		'LAKE-4'				
2						
1		'SUCCESS'				
2		'FAILURE'				

```

$
$ This file was written on 02/08/94 at 15:43:27 by
$ ETPRE Version 3.00, compiled with Microsoft V 5.1.
$ Tree Definition File is PROB2.PRE.
$ Keyword File is PROB2.EK1.
$
@BIN
$$ EVNTRE/PSTEVNT Binning Labels Version 1.00
$$ Title Binning input for the ice rescue problem
$$ Dimensions: 2
$$ Dim 1 4 SITE
$$ 1 A LAKE-1
$$ 1 B LAKE-2
$$ 1 C LAKE-3
$$ 1 D LAKE-4
$$ Dim 2 2 OUTCOME
$$ 2 A SUCCESS
$$ 2 B FAILURE
$$ Done
@REBIN
$$ EVNTRE/PSTEVNT Binning Labels Version 1.00
$$ Title Binning input for the ice rescue problem
$$ Dimensions: 2
$$ Dim 1 4 SITE
$$ 1 A LAKE-1
$$ 1 B LAKE-2
$$ 1 C LAKE-3
$$ 1 D LAKE-4
$$ Dim 2 2 OUTCOME
$$ 2 A SUCCESS
$$ 2 B FAILURE
$$ Done

```


Appendix C- Example Problem 3

ETPRE Execution Sequence

D:\ETPRE\EXAMPLE>ETPRE PROB3.EKI

ETPRE VER. 3.00 02/08/94
AUTHOR: Bob Roginski ORG. 12333

Run date & time = 02/08/94 15:43:45
Compiler Used = Microsoft V 5.1

***** INPUT FILES *****

NOTE: KEYWORD INPUT FILE IS PROB3.EKI
NOTE: TREE DEF. INPUT FILE IS PROB3.PRE

***** OUTPUT FILES *****

NOTE: "EVNTRE" INPUT WRITTEN TO FILE PROB3.TRE
NOTE: BINNING INPUT WRITTEN TO FILE PROB3.BIN
NOTE: PSTEVNT INPUT WRITTEN TO FILE PROB3.RBN

***** RESOURCE USAGE *****

62 TOTAL INPUT LINES READ
5 OF 200 MAX QUESTIONS USED
1 OF 32 MAX SEQUENCES USED
14 OF 600 MAX TOTAL BRANCHES USED
14 OF 2400 MAX TOTAL SPLIT-FRACTIONS USED
5 OF 600 MAX TOTAL EXPRESSIONS USED
7 OF 1800 MAX TOTAL SYMBOLS USED
4 OF 200 MAX TOTAL REF. PARAMS. USED
15 OF 2400 MAX TOTAL PARAM. VALUES USED
0 OF 200 MAX TOTAL POINTERS USED
13 OF 1800 MAX TOTAL REFERENCE TYPES USED
6 OF 1200 MAX TOTAL S.F. COMPARISON VALUES
15 OF 600 MAX TOTAL COMPARISON PARAMETERS
3 OF 600 MAX TOTAL PARAM. COMPARISON TYPES
3 OF 600 MAX TOTAL INTERNAL FUNCTION REFS.
0 OF 256 MAX TOTAL SAMPLE VECTORS
0 OF 256 MAX TOTAL VECTOR LABELS
2 OF 32 MAX TOTAL BINNING CHARACTERISTICS
6 OF 128 MAX TOTAL BINNING OUTCOMES
6 OF 384 MAX TOTAL BINNING SYMBOLS
1 OF 16 MAX TOTAL SORTS
2 OF 128 MAX TOTAL SORT KEYS
2 OF 32 MAX TOTAL RE-BIN. CHARACTERISTICS
6 OF 128 MAX TOTAL RE-BIN. OUTCOMES
6 OF 384 MAX TOTAL RE-BIN. SYMBOLS
0 OF 256 MAX TOTAL LHS SYMBOLIC NAMES
0 OF 16 MAX TOTAL LHS FILE PATHS
0 OF 512 MAX TOTAL LHS SAMPLED SYMBOLS

*** 1 MINOR ERRORS ***
(SEE FILE ETPRE.WNG)

Stop - Program terminated.

Keyword Input File PROB3.EKI (File 1)

```
PRETRIN PROB3.PRE
MODE 1
TREEIN PROB3.TRE
BININ PROB3.BIN
STATS
PRTINP
BIN
WRTBIN
SAVEBIN
INPOUT PROB3-IN.OUT
STATOUT PROB3-STA.OUT
SAMROUT PROB3-PST.OUT
BINOUT PROB3-BIN.OUT
KEEPCUT 5.0E-11
PRTCUT 5.0E-9
RUN
ENDKEY
```

ETPRE Tree Definition File PROB3.PRE (File 2)

\$ This example input begins on slide #44 in session 3.

Rescue of a Person Falling Through the Ice - Considering Age
Only-Eval 1.00

\$ Determine whether the victim is child, adult, or senior

How old is the victim?

Child	Adult	Senior
0.80	0.19	0.01

At which lake does the incident occur?

Lake1	Lake2	Lake3	Lake4
0.25	0.25	0.25	0.25

Variable: Notification

7	7	2	2
---	---	---	---

Variable: Travel-Time

2	3	5	8
---	---	---	---

Do unusually cold temperatures hamper the response effort?

	Usual	Extra-Cold
Lake1 + Lake2	0.90	0.10
Otherwise	0.90	0.10

Variable: Cold-Time

Lake1 + Lake2	0	1
Otherwise	0	2

Where in the lake did the person fall through the ice?

Near-Shore	Int-Dist	Far-Out
0.50	0.30	0.20

Variable: Rescue-Time

1	3	5
---	---	---

\$ Change the final question to account for the differences

\$ in survival times for children, adults and seniors

Does the victim survive?

Death	Survival
Normal Add(Notification, Travel-Time, Cold-Time, Rescue-Time)	
Child	21 5

Normal Add(Notification, Travel-Time, Cold-Time, Rescue-Time)	
Adult	16 4

```

Normal Add( Notification, Travel-Time, Cold-Time, Rescue-Time )
Otherwise      10      3

```

\$ The binning from the previous version of this example can be used
 \$ if desired, or a dimension can be added to retain information about
 \$ the age of the person being rescued.

Binning: Sample binning input for the ice rescue example.

```

Site
1      Lake-1      Lake1
2      Lake-2      Lake2
3      Lake-3      Lake3
4      Lake-4      Lake4

```

```

Outcome
1      Success      Survival
2      Failure      Death

```

Sorting:

```

Outcome      Site

```

EVNTRE Tree Definition Input File PROB3.TRE (File 3)

Rescue of a Person Falling Through the Ice - Considering Age

```

5
NQ
1  1.00000E+00
  ONLY-EVAL
1 How old is the victim?
3  CHILD  ADULT  SENIOR
1
  8.00000E-01  1.90000E-01  1.00000E-02
2 At which lake does the incident occur?
4  LAKE1  LAKE2  LAKE3  LAKE4
3
  2.50000E-01  2.50000E-01  2.50000E-01  2.50000E-01
2
1  7.00000  7.00000  2.00000  2.00000
2  2.00000  3.00000  5.00000  8.00000
3 Do unusually cold temperatures hamper the response effort?
2  USUAL  EXTRA-COLD
4
2
2
  2      2
  1      +  2
  LAKE1  LAKE2
  9.00000E-01  1.00000E-01
1
3  .000000  1.00000
  OTHERWISE
  9.00000E-01  1.00000E-01
1
3  .000000  2.00000
4 Where in the lake did the person fall through the ice?
3  NEAR-SHORE  INT-DIST  FAR-OUT
3
  5.00000E-01  3.00000E-01  2.00000E-01
1
4  1.00000  3.00000  5.00000
5 Does the victim survive?
2  DEATH  SURVIVAL
6
3
1      1
  1

```

```

CHILD
4 1 2 3 4
  NOTIFICATION TRAVEL-TIME COLD-TIME RESCUE-TIME
ADD
NORMAL 2 21.0000 5.00000
REQUIRED COMMENT FOR QUESTION 5 CASE 1
1 1
  2
ADULT
4 1 2 3 4
  NOTIFICATION TRAVEL-TIME COLD-TIME RESCUE-TIME
ADD
NORMAL 2 16.0000 4.00000
REQUIRED COMMENT FOR QUESTION 5 CASE 2
OTHERWISE
4 1 2 3 4
  NOTIFICATION TRAVEL-TIME COLD-TIME RESCUE-TIME
ADD
NORMAL 2 10.0000 3.00000
REQUIRED COMMENT FOR QUESTION 5 CASE 3

```

Binning and Sorting Definition File PROB3.BIN (File 4)

Sample binning input for the ice rescue example.

```

2 SITE Outcome
4 4 Lake-1 Lake-2 Lake-3 Lake-4
1 1 2
  1
  Lake1
1 2 2
  2
  Lake2
1 3 2
  3
  Lake3
1 4 2
  4
  Lake4
2 2 Success Failure
1 1 5
  2
  Survival
1 2 5
  1
  Death
1
2 2 1
  Outcome Site

```

Rebinning Definition File PROB3.RBN (File 5)

Sample binning input for the ice rescue example.

```

2 SITE Outcome
4 4 LAKE-1 LAKE-2 LAKE-3 LAKE-4
1 1 1
  1
  LAKE-1
1 2 1
  2
  LAKE-2
1 3 1

```

```

      3
    LAKE-3
1  4  1
      4
    LAKE-4
2  2  SUCCESS      FAILURE
1  1  2
      1
    SUCCESS
1  2  2
      2
    FAILURE
1
2      2      1
    OUTCOME      SITE
1
4
1  'LAKE-1'
2  'LAKE-2'
3  'LAKE-3'
4  'LAKE-4'
2
1  'SUCCESS'
2  'FAILURE'
$
$ This file was written on 02/08/94 at 15:43:45 by
$ ETPRE Version 3.00, compiled with Microsoft V 5.1.
$ Tree Definition File is PROB3.PRE.
$ Keyword File is PROB3.EK1.
$
@BIN
$$ EVNTRE/PSTEVNT Binning Labels Version 1.00
$$ Title Sample binning input for the ice rescue example.
$$ Dimensions: 2
$$ Dim 1 4 SITE
$$ 1 A LAKE-1
$$ 1 B LAKE-2
$$ 1 C LAKE-3
$$ 1 D LAKE-4
$$ Dim 2 2 OUTCOME
$$ 2 A SUCCESS
$$ 2 B FAILURE
$$ Done
@REBIN
$$ EVNTRE/PSTEVNT Binning Labels Version 1.00
$$ Title Sample binning input for the ice rescue example.
$$ Dimensions: 2
$$ Dim 1 4 SITE
$$ 1 A LAKE-1
$$ 1 B LAKE-2
$$ 1 C LAKE-3
$$ 1 D LAKE-4
$$ Dim 2 2 OUTCOME
$$ 2 A SUCCESS
$$ 2 B FAILURE
$$ Done

```

Appendix D- Example Problem 4

ETPRE Execution Sequence

D:\ETPRE\EXAMPLE>ETPRE PROB4.EKI

ETPRE VER. 3.00 02/08/94
AUTHOR: Bob Roginski ORG. 12333

Run date & time = 02/08/94 15:44:05
Compiler Used = Microsoft V 5.1

***** INPUT FILES *****

NOTE: KEYWORD INPUT FILE IS PROB4.EKI
NOTE: TREE DEF. INPUT FILE IS PROB4.PRE

***** OUTPUT FILES *****

NOTE: "EVTRE" INPUT WRITTEN TO FILE PROB4.TRE
NOTE: BINNING INPUT WRITTEN TO FILE PROB4.BIN
NOTE: PSTEVNT INPUT WRITTEN TO FILE PROB4.RBN

***** RESOURCE USAGE *****

65 TOTAL INPUT LINES READ
6 OF 200 MAX QUESTIONS USED
1 OF 32 MAX SEQUENCES USED
23 OF 600 MAX TOTAL BRANCHES USED
16 OF 2400 MAX TOTAL SPLIT-FRACTIONS USED
0 OF 600 MAX TOTAL EXPRESSIONS USED
0 OF 1800 MAX TOTAL SYMBOLS USED
5 OF 200 MAX TOTAL REF. PARAMS. USED
21 OF 2400 MAX TOTAL PARAM. VALUES USED
0 OF 200 MAX TOTAL POINTERS USED
11 OF 1800 MAX TOTAL REFERENCE TYPES USED
5 OF 1200 MAX TOTAL S.F. COMPARISON VALUES
7 OF 600 MAX TOTAL COMPARISON PARAMETERS
2 OF 600 MAX TOTAL PARAM. COMPARISON TYPES
2 OF 600 MAX TOTAL INTERNAL FUNCTION REFS.
0 OF 256 MAX TOTAL SAMPLE VECTORS
0 OF 256 MAX TOTAL VECTOR LABELS
2 OF 32 MAX TOTAL BINNING CHARACTERISTICS
7 OF 128 MAX TOTAL BINNING OUTCOMES
7 OF 384 MAX TOTAL BINNING SYMBOLS
1 OF 16 MAX TOTAL SORTS
2 OF 128 MAX TOTAL SORT KEYS
2 OF 32 MAX TOTAL RE-BIN. CHARACTERISTICS
7 OF 128 MAX TOTAL RE-BIN. OUTCOMES
7 OF 384 MAX TOTAL RE-BIN. SYMBOLS
0 OF 256 MAX TOTAL LHS SYMBOLIC NAMES
0 OF 16 MAX TOTAL LHS FILE PATHS
0 OF 512 MAX TOTAL LHS SAMPLED SYMBOLS

*** 1 MINOR ERRORS ***
(SEE FILE ETPRE.WNG)

Stop - Program terminated.

Keyword Input File PROB4.EKI (File 1)

```

PRETRIN PROB4.PRE
MODE 1
TREEIN PROB4.TRE
BININ PROB4.BIN
STATS
PRTINP
BIN
WRTBIN
SAVEBIN
INPOUT PROB4-IN.OUT
STATOUT PROB4-STA.OUT
SAMROUT PROB4-PST.OUT
BINOUT PROB4-BIN.OUT
KEEPCUT 5.0E-11
PRTCUT 5.0E-9
RUN
ENDKEY

```

ETPRE Tree Definition File PROB4.PRE (File 2)

Four branch gas pump example with a type 7 question
Only-Eval 1.00

Is Pump #1 Damaged?

Pump1-OK	Pump1-7gpm	Pump1-12gpm	Pump1-20gpm
0.60	0.20	0.10	0.10
Variable: Pump1-Flow			
0.0	7.0	12.0	20.0

Is Pump #2 Damaged?

Pump2-OK	Pump2-7gpm	Pump2-12gpm	Pump2-20gpm
0.60	0.20	0.10	0.10
Variable: Pump2-Flow			
0.0	7.0	12.0	20.0

Is Pump #3 Damaged?

Pump3-OK	Pump3-7gpm	Pump3-12gpm	Pump3-20gpm
0.60	0.20	0.10	0.10
Variable: Pump3-Flow			
0.0	7.0	12.0	20.0

Is Pump #4 Damaged?

Pump4-OK	Pump4-7gpm	Pump4-12gpm	Pump4-20gpm
0.60	0.20	0.10	0.10
Variable: Pump4-Flow			
0.0	7.0	12.0	20.0

\$ This is now a type 7 question because a parameter is defined as part of what
\$ used to be a type 5 question

Does an explosion occur?

Big-Boom	Explosion	Big-Fire	Small-Fire	NoFire
GEThresh Add (Pump1-Flow, Pump2-Flow, Pump3-Flow, Pump4-Flow)				
	70.0	50.0	25.0	10.0
Variable: Fire-Temp	\$ Fire temperature, degrees Kelvin			
1000	900	800	600	293

\$ Now add a type 5 question to look at the response to the fire temperature

Can the fire department respond to the fire?

	FD-Effective	FD-Ineffective
Thresh Add(Fire-Temp)		
	700	

Binning: Binning for type 7 question

Result		
1	Neighborhood	Big-Boom
2	Station	Explosion
3	Building-Fire	Big-Fire
4	Pump-Fire	Small-Fire
5	NoDamage	NoFire

Fire-Dept		
1	Effective	FD-Effective
2	Ineffective	FD-Ineffective

Sorting: Fire-Dept Result

EVNTRE Tree Definition File PROB4.TRE (File 3)

Four branch gas pump example with a type 7 question

```

6
NQ
1 1.00000E+00
ONLY-EVAL
1 Is Pump #1 Damaged?
4 PUMP1-OK PUMP1-7GPM PUMP1-12GPM PUMP1-20GPM
3
6.00000E-01 2.00000E-01 1.00000E-01 1.00000E-01
1
.000000 7.00000 12.0000 20.0000
2 Is Pump #2 Damaged?
4 PUMP2-OK PUMP2-7GPM PUMP2-12GPM PUMP2-20GPM
3
6.00000E-01 2.00000E-01 1.00000E-01 1.00000E-01
1
.000000 7.00000 12.0000 20.0000
3 Is Pump #3 Damaged?
4 PUMP3-OK PUMP3-7GPM PUMP3-12GPM PUMP3-20GPM
3
6.00000E-01 2.00000E-01 1.00000E-01 1.00000E-01
1
.000000 7.00000 12.0000 20.0000
4 Is Pump #4 Damaged?
4 PUMP4-OK PUMP4-7GPM PUMP4-12GPM PUMP4-20GPM
3
6.00000E-01 2.00000E-01 1.00000E-01 1.00000E-01
1
.000000 7.00000 12.0000 20.0000
5 Does an explosion occur?
5 BIG-BOOM EXPLOSION BIG-FIRE SMALL-FIRE NOFIRE
7
4 1 2 3 4
PUMP1-FLOW PUMP2-FLOW PUMP3-FLOW PUMP4-FLOW
ADD
GETHRESH 4 70.0000 50.0000 25.0000 10.0000
REQUIRED COMMENT FOR QUESTION 5
1
5 1000.00 900.000 800.000 600.000 293.000
6 Can the fire department respond to the fire?

```



```

2  FD-EFFECTIVE  FD-INEFFECTIVE
5
1      5
      FIRE-TEMP
ADD
THRESH      1      700.000
REQUIRED COMMENT FOR QUESTION  6

```

Binning and Sorting Definition File PROB4.BIN (File 4)

Binning for type 7 question

```

2      RESULT      FIRE-DEPT
5  5  NEIGHBORHOOD  STATION      BUILDING-FIRE  PUMP-FIRE
      NODAMAGE
1  1      5
      1
      BIG-BOOM
1  2      5
      2
      EXPLOSION
1  3      5
      3
      BIG-FIRE
1  4      5
      4
      SMALL-FIRE
1  5      5
      5
      NOFIRE
2  2  EFFECTIVE      INEFFECTIVE
1  1      6
      1
      FD-EFFECTIVE
1  2      6
      2
      FD-INEFFECTIVE
1
2      2      1
      FIRE-DEPT      RESULT

```

Rebinning Definition File PROB4.RBN (File 5)

Binning for type 7 question

```

2      RESULT      FIRE-DEPT
5  5  NEIGHBORHOOD  STATION      BUILDING-FIRE  PUMP-FIRE
      NODAMAGE
1  1      1
      1
      NEIGHBORHOOD
1  2      1
      2
      STATION
1  3      1
      3
      BUILDING-FIRE
1  4      1
      4
      PUMP-FIRE
1  5      1
      5
      NODAMAGE

```

```

2 2 EFFECTIVE      INEFFECTIVE
1 1      2
      1
      EFFECTIVE
1 2      2
      2
      INEFFECTIVE
1
2      2      1
      FIRE-DEPT  RESULT
1
5
1 'NEIGHBORHOOD'
2 'STATION'
3 'BUILDING-FIRE'
4 'PUMP-FIRE'
5 'NODAMAGE'
2
1 'EFFECTIVE'
2 'INEFFECTIVE'
$
$ This file was written on 02/08/94 at 15:44:05 by
$ ETPRE Version 3.00, compiled with Microsoft V 5.1.
$ Tree Definition File is PROB4.PRE.
$ Keyword File is PROB4.EK1.
$
@BIN
$$ EVNTRE/PSTEVNT Binning Labels Version 1.00
$$ Title Binning for type 7 question
$$ Dimensions: 2
$$ Dim 1 5 RESULT
$$ 1 A NEIGHBORHOOD
$$ 1 B STATION
$$ 1 C BUILDING-FIRE
$$ 1 D PUMP-FIRE
$$ 1 E NODAMAGE
$$ Dim 2 2 FIRE-DEPT
$$ 2 A EFFECTIVE
$$ 2 B INEFFECTIVE
$$ Done
@REBIN
$$ EVNTRE/PSTEVNT Binning Labels Version 1.00
$$ Title Binning for type 7 question
$$ Dimensions: 2
$$ Dim 1 5 RESULT
$$ 1 A NEIGHBORHOOD
$$ 1 B STATION
$$ 1 C BUILDING-FIRE
$$ 1 D PUMP-FIRE
$$ 1 E NODAMAGE
$$ Dim 2 2 FIRE-DEPT
$$ 2 A EFFECTIVE
$$ 2 B INEFFECTIVE
$$ Done

```

Appendix E- Example Problem 5

ETPRE Execution Sequence

D:\ETPRE\EXAMPLE>ETPRE PROB5.EKI

ETPRE VER. 3.00 02/08/94
AUTHOR: Bob Roginski ORG. 12333

Run date & time = 02/08/94 15:44:19
Compiler Used = Microsoft V 5.1

***** INPUT FILES *****

NOTE: KEYWORD INPUT FILE IS PROB5.EKI
NOTE: TREE DEF. INPUT FILE IS PROB5.PRE

***** OUTPUT FILES *****

NOTE: "EVNTRE" INPUT WRITTEN TO FILE PROB5.TRE
NOTE: BINNING INPUT WRITTEN TO FILE PROB5.BIN
NOTE: PSTEVNT INPUT WRITTEN TO FILE PROB5.RBN

***** RESOURCE USAGE *****

128 TOTAL INPUT LINES READ
9 OF 200 MAX QUESTIONS USED
1 OF 32 MAX SEQUENCES USED
27 OF 600 MAX TOTAL BRANCHES USED
55 OF 2400 MAX TOTAL SPLIT-FRACTIONS USED
18 OF 600 MAX TOTAL EXPRESSIONS USED
18 OF 1800 MAX TOTAL SYMBOLS USED
7 OF 200 MAX TOTAL REF. PARAMS. USED
64 OF 2400 MAX TOTAL PARAM. VALUES USED
0 OF 200 MAX TOTAL POINTERS USED
40 OF 1800 MAX TOTAL REFERENCE TYPES USED
10 OF 1200 MAX TOTAL S.F. COMPARISON VALUES
23 OF 600 MAX TOTAL COMPARISON PARAMETERS
4 OF 600 MAX TOTAL PARAM. COMPARISON TYPES
4 OF 600 MAX TOTAL INTERNAL FUNCTION REFS.
0 OF 256 MAX TOTAL SAMPLE VECTORS
0 OF 256 MAX TOTAL VECTOR LABELS
4 OF 32 MAX TOTAL BINNING CHARACTERISTICS
10 OF 128 MAX TOTAL BINNING OUTCOMES
16 OF 384 MAX TOTAL BINNING SYMBOLS
3 OF 16 MAX TOTAL SORTS
6 OF 128 MAX TOTAL SORT KEYS
4 OF 32 MAX TOTAL RE-BIN. CHARACTERISTICS
10 OF 128 MAX TOTAL RE-BIN. OUTCOMES
10 OF 384 MAX TOTAL RE-BIN. SYMBOLS
0 OF 256 MAX TOTAL LHS SYMBOLIC NAMES
0 OF 16 MAX TOTAL LHS FILE PATHS
0 OF 512 MAX TOTAL LHS SAMPLED SYMBOLS

*** 1 MINOR ERRORS ***
(SEE FILE ETPRE.WNG)

Stop - Program terminated.

Keyword Input File PROB5.EKI (File 1)

```

PRETRIN PROB5.PRE
MODE 1
TREEIN PROB5.TRE
BININ PROB5.BIN
STATS
PRTINP
BIN
WRTBIN
SAVEBIN
INPOUT PROB5-IN.OUT
STATOUT PROB5-STA.OUT
SAMROUT PROB5-PST.OUT
BINOUT PROB5-BIN.OUT
KEEPCUT 5.0E-11
PRTCUT 5.0E-9
RUN
ENDKEY

```

ETPRE Tree Definition File PROB5.PRE (File 2)

Model of an independent contractor's average finances and loan possibilities
Only-Eval 1.00

What month is it?

High-Pay	Nominal-Pay	Low-Pay
0.25	0.42	0.33

How much work is done for client #1 this month?

	A1	A2	A3
High-Pay	0.10	0.40	0.50
Nominal-Pay	0.20	0.40	0.40
Otherwise	0.30	0.40	0.30

Variable: Client1

High-Pay	300	500	700
Nominal-Pay	300	500	700
Otherwise	300	500	700

How much work is done for client #2 this month?

	B1	B2	B3	B4
High-Pay	0.00	0.20	0.60	0.20
Nominal-Pay	0.10	0.20	0.60	0.10
Otherwise	0.20	0.20	0.60	0.00

Variable: Client2

High-Pay	400	500	800	1100
Nominal-Pay	400	500	800	1100
Otherwise	400	500	800	1100

How much work is done for client #3 this month?

	C1	C2	C3	C4	C5
High-Pay	0.10	0.20	0.20	0.20	0.30
Nominal-Pay	0.20	0.20	0.20	0.20	0.20
Otherwise	0.30	0.20	0.20	0.20	0.10

Variable: Client3

High-Pay	200	350	500	650	800
Nominal-Pay	200	350	500	650	800
Otherwise	200	350	500	650	800

	D1	D2
High-Pay	0.40	0.60
Nominal - Pay	0.50	0.50
Otherwise	0.60	0.40

High-Pay	500	700
Nominal-Pay	500	700
Otherwise	500	700

	E1	E2	E3
High-Pay	0.20	0.30	0.50
Nominal-Pay	0.30	0.30	0.40
Otherwise	0.40	0.30	0.30

High-Pay	600	800	1100
Nominal-Pay	600	800	1100
Otherwise	600	800	1100

F1
1.00

700

\$ This is now a type 8 question because a parameter is being defined as a part
\$ of what formerly was a type 6 question.

Does he make his budget?

LT-Save Budget-OK Budget-NotOK LT-Borrow

```
GEThresh Add( Client1, Client2, Client3, Client4, Client5, Client6 )
High-Pay          4300    4000          3700
```

GEThresh Add(Client1, Client2, Client3, Client4, Client5, Client6)					
Nominal-Pay	3900	3600		3300	

```
GEThresh Add( Client1, Client2, Client3, Client4, Client5, Client6 )
Otherwise           3300      3000           2700
```

Variable: Interest-Rate

High-Pay	8.0	8.5	9.5	11.0
Nominal-Pay	8.5	9.0	10.0	12.0
Otherwise	9.5	10.0	11.0	13.0

\$ Now add a Type 5 question to make use of the defined parameter. If the contractor can
\$ get an interest rate lower than 9.25%, he can take out a loan to expand his business.

Can he take out a loan to improve and expand his business?

Loan-OK	No-Loan
---------	---------

Thresh Add(Interest-Rate)

9.25

Binning: Binning for a type 8 question example

Month

1	High	High-Pay
2	Nominal	Nominal-Pay
3	Low	Low-Pay

Budget

1	OK	LT-Save + Budget-OK
2	Short	Budget-NotOK + LT-Borrow
Savings		
1	Save	LT-Save
2	NoChange	Budget-OK + Budget-NotOK
3	Borrow	LT-Borrow
Loan-Status		
1	Loan	Loan-OK
2	NoLoan	No-Loan

Sorting:

Budget	Month
Savings	Month
Loan-Status	Month

EVNTRE Tree Definition File PROB5.TRE (File 3)

Model of an independent contractor's average finances and loan possibilities

```

9
NQ
1 1.00000E+00
  ONLY-EVAL
1 What month is it?
  3 HIGH-PAY NOMINAL-PAY LOW-PAY
  1
    2.50000E-01 4.20000E-01 3.30000E-01
2 How much work is done for client #1 this month?
  3 A1 A2 A3
  4
  3
  1
    1
    HIGH-PAY
    1.00000E-01 4.00000E-01 5.00000E-01
  1
  1 300.000 500.000 700.000
  1
    1
    2
    NOMINAL-PAY
    2.00000E-01 4.00000E-01 4.00000E-01
  1
  1 300.000 500.000 700.000
  OTHERWISE
    3.00000E-01 4.00000E-01 3.00000E-01
  1
  1 300.000 500.000 700.000
3 How much work is done for client #2 this month?
  4 B1 B2 B3 B4
  4
  3
  1
    1
    HIGH-PAY
    0.00000E+00 2.00000E-01 6.00000E-01 2.00000E-01
  1
  2 400.000 500.000 800.000 1100.00
  1
    1
    2
    NOMINAL-PAY
    1.00000E-01 2.00000E-01 6.00000E-01 1.00000E-01
  1
  2 400.000 500.000 800.000 1100.00
  OTHERWISE

```

	1	2.00000E-01	2.00000E-01	6.00000E-01	0.00000E+00
	2	400.000	500.000	800.000	1100.00
4	How much work is done for client #3 this month?				
	5	C1	C2	C3	C4
	4				
	3				
	1		1		
			1		
		HIGH-PAY			
		1.00000E-01	2.00000E-01	2.00000E-01	2.00000E-01
	1				
	3	200.000	350.000	500.000	650.000
	1		1		
			2		
		NOMINAL-PAY			
		2.00000E-01	2.00000E-01	2.00000E-01	2.00000E-01
	1				
	3	200.000	350.000	500.000	650.000
		OTHERWISE			
		3.00000E-01	2.00000E-01	2.00000E-01	2.00000E-01
	1				
	3	200.000	350.000	500.000	650.000
5	How much work is done for client #4 this month?				
	2	D1	D2		
	4				
	3				
	1		1		
			1		
		HIGH-PAY			
		4.00000E-01	6.00000E-01		
	1				
	4	500.000	700.000		
	1		1		
			2		
		NOMINAL-PAY			
		5.00000E-01	5.00000E-01		
	1				
	4	500.000	700.000		
		OTHERWISE			
		6.00000E-01	4.00000E-01		
	1				
	4	500.000	700.000		
6	How much work is done for client #5 this month?				
	3	E1	E2	E3	
	4				
	3				
	1		1		
			1		
		HIGH-PAY			
		2.00000E-01	3.00000E-01	5.00000E-01	
	1				
	5	500.000	800.000	1100.00	
	1		1		
			2		
		NOMINAL-PAY			
		3.00000E-01	3.00000E-01	4.00000E-01	
	1				
	5	600.000	800.000	1100.00	
		OTHERWISE			
		4.00000E-01	3.00000E-01	3.00000E-01	
	1				
	5	600.000	800.000	1100.00	
7	How much work is done for client #6 this month?				
	1	F1			
	3				
		1.00000E+00			
	1				
	6	700.000			
8	Does he make his budget?				

```

4 LT-SAVE BUDGET-OK BUDGET-NOTOK LT-BORROW
8
3
1 1
1 1
HIGH-PAY
6 1 2 3 4
5 6
CLIENT1 CLIENT2 CLIENT3 CLIENT4
CLIENT5 CLIENT6
$
ADD
GETHRESH 3 4300.00 4000.00 3700.00
REQUIRED COMMENT FOR QUESTION 8 CASE 1
1
7 8.00000 8.50000 9.50000 11.0000
1 1
2
NOMINAL-PAY
6 1 2 3 4
5 6
CLIENT1 CLIENT2 CLIENT3 CLIENT4
CLIENT5 CLIENT6
$
ADD
GETHRESH 3 3900.00 3600.00 3300.00
REQUIRED COMMENT FOR QUESTION 8 CASE 2
1
7 8.50000 9.00000 10.0000 12.0000
OTHERWISE
6 1 2 3 4
5 6
CLIENT1 CLIENT2 CLIENT3 CLIENT4
CLIENT5 CLIENT6
$
ADD
GETHRESH 3 3300.00 3000.00 2700.00
REQUIRED COMMENT FOR QUESTION 8 CASE 3
1
7 9.50000 10.0000 11.0000 13.0000
9 Can he take out a loan to improve and expand his business?
2 LOAN-OK NO-LOAN
5
1 7
INTEREST-RATE
ADD
THRESH 1 9.25000
REQUIRED COMMENT FOR QUESTION 9

```

Binning and Sorting Definition File PROB5.BIN (File 4)

Binning for a type 8 question example

```

4 MONTH BUDGET SAVINGS LOAN-STATUS
3 3 HIGH NOMINAL LOW
1 1 1
1 1
HIGH-PAY
1 2 1
2
NOMINAL-PAY
1 3 1
3
LOW-PAY
2 2 OK SHORT
2 1 8 8
1 + 2
LT-SAVE BUDGET-OK
2 2 8 8

```


		3	+	4	
		BUDGET-NOTOK		LT-BORROW	
3	3	SAVE		NOCHANGE	BORROW
1	1	8			
		1			
		LT-SAVE			
2	2	8		8	
		2		+	3
		BUDGET-OK		BUDGET-NOTOK	
1	3	8			
		4			
		LT-BORROW			
2	2	LOAN		NOLOAN	
1	1	9			
		1			
		LOAN-OK			
1	2	9			
		2			
		NO-LOAN			
3					
2		2		1	
		BUDGET		MONTH	
2		3		1	
		SAVINGS		MONTH	
2		4		1	
		LOAN-STATUS		MONTH	

Rebinning Definition File PROB5.RBN (File 5)

Binning for a type 8 question example

4		MONTH		BUDGET		SAVINGS		LOAN-STATUS
3	3	HIGH		NOMINAL		LOW		
1	1	1						
		1						
		HIGH						
1	2	1						
		2						
		NOMINAL						
1	3	1						
		3						
		LOW						
2	2	OK		SHORT				
1	1	2						
		1						
		OK						
1	2	2						
		2						
		SHORT						
3	3	SAVE		NOCHANGE		BORROW		
1	1	3						
		1						
		SAVE						
1	2	3						
		2						
		NOCHANGE						
1	3	3						
		3						
		BORROW						
2	2	LOAN		NOLOAN				
1	1	4						
		1						
		LOAN						
1	2	4						
		2						
		NOLOAN						

```

3
2      2      1
      BUDGET      MONTH
2      3      1
      SAVINGS      MONTH
2      4      1
      LOAN-STATUS      MONTH
1
3
1 'HIGH'
2 'NOMINAL'
3 'LOW'
2
1 'OK'
2 'SHORT'
3
1 'SAVE'
2 'NOCHANGE'
3 'BORROW'
2
1 'LOAN'
2 'NOLOAN'
$
$ This file was written on 02/08/94 at 15:44:19 by
$ ETPRE Version 3.00, compiled with Microsoft V 5.1.
$ Tree Definition File is PROB5.PRE.
$ Keyword File is PROB5.EK1.
$
@BIN
$$ EVNTRE/PSTEVNT Binning Labels Version 1.00
$$ Title Binning for a type 8 question example
$$ Dimensions: 4
$$ Dim 1 3 MONTH
$$ 1 A HIGH
$$ 1 B NOMINAL
$$ 1 C LOW
$$ Dim 2 2 BUDGET
$$ 2 A OK
$$ 2 B SHORT
$$ Dim 3 3 SAVINGS
$$ 3 A SAVE
$$ 3 B NOCHANGE
$$ 3 C BORROW
$$ Dim 4 2 LOAN-STATUS
$$ 4 A LOAN
$$ 4 B NOLOAN
$$ Done
@REBIN
$$ EVNTRE/PSTEVNT Binning Labels Version 1.00
$$ Title Binning for a type 8 question example
$$ Dimensions: 4
$$ Dim 1 3 MONTH
$$ 1 A HIGH
$$ 1 B NOMINAL
$$ 1 C LOW
$$ Dim 2 2 BUDGET
$$ 2 A OK
$$ 2 B SHORT
$$ Dim 3 3 SAVINGS
$$ 3 A SAVE
$$ 3 B NOCHANGE
$$ 3 C BORROW
$$ Dim 4 2 LOAN-STATUS
$$ 4 A LOAN
$$ 4 B NOLOAN
$$ Done

```

Appendix F - Example Problem 6

ETPRE Execution Sequence

D:\ETPRE\EXAMPLE>ETPRE DEMO.EKI

ETPRE VER. 3.00 02/08/94
AUTHOR: Bob Roginski ORG. 12333

Run date & time = 02/10/94 12:34:58
Compiler Used = Microsoft V 5.1

***** INPUT FILES *****

NOTE: KEYWORD INPUT FILE IS DEMO.EKI
NOTE: TREE DEF. INPUT FILE IS D:\ETPRE\EXAMPLE\DEMO.PRE
NOTE: "LHSOUT" INPUT FILE IS D:\ETPRE\EXAMPLE\DEMO.LSP
NOTE: "LHSPOST" INPUT FILE IS D:\ETPRE\EXAMPLE\DEMO.MSP
NOTE: LHS USER INPUT FILE IS D:\ETPRE\EXAMPLE\LHS.LSP

***** OUTPUT FILES *****

NOTE: "EVNTRE" INPUT WRITTEN TO FILE D:\ETPRE\EXAMPLE\DEMO.EPI
NOTE: "EVNTRE" INPUT WRITTEN TO FILE D:\ETPRE\EXAMPLE\DEMO.EVI
NOTE: "EVNTRE" INPUT WRITTEN TO FILE D:\ETPRE\EXAMPLE\DEMO.ETI
NOTE: BINNING INPUT WRITTEN TO FILE D:\ETPRE\EXAMPLE\DEMO.EBI
NOTE: PSTEVNT INPUT WRITTEN TO FILE D:\ETPRE\EXAMPLE\DEMO.DPF

***** RESOURCE USAGE *****

113 TOTAL INPUT LINES READ
8 OF 200 MAX QUESTIONS USED
3 OF 32 MAX SEQUENCES USED
17 OF 600 MAX TOTAL BRANCHES USED
57 OF 2400 MAX TOTAL SPLIT-FRACTIONS USED
5 OF 600 MAX TOTAL EXPRESSIONS USED
5 OF 1800 MAX TOTAL SYMBOLS USED
4 OF 200 MAX TOTAL REF. PARAMS. USED
24 OF 2400 MAX TOTAL PARAM. VALUES USED
0 OF 200 MAX TOTAL POINTERS USED
42 OF 1800 MAX TOTAL REFERENCE TYPES USED
6 OF 1200 MAX TOTAL S.F. COMPARISON VALUES
5 OF 600 MAX TOTAL COMPARISON PARAMETERS
1 OF 600 MAX TOTAL PARAM. COMPARISON TYPES
1 OF 600 MAX TOTAL INTERNAL FUNCTION REFS.
17 OF 256 MAX TOTAL SAMPLE VECTORS
17 OF 256 MAX TOTAL VECTOR LABELS
6 OF 32 MAX TOTAL BINNING CHARACTERISTICS
13 OF 128 MAX TOTAL BINNING OUTCOMES
13 OF 384 MAX TOTAL BINNING SYMBOLS
2 OF 16 MAX TOTAL SORTS
8 OF 128 MAX TOTAL SORT KEYS
2 OF 32 MAX TOTAL RE-BIN. CHARACTERISTICS
5 OF 128 MAX TOTAL RE-BIN. OUTCOMES
5 OF 384 MAX TOTAL RE-BIN. SYMBOLS
13 OF 256 MAX TOTAL LHS SYMBOLIC NAMES
1 OF 16 MAX TOTAL LHS FILE PATHS
12 OF 512 MAX TOTAL LHS SAMPLED SYMBOLS

*** 1 MINOR ERRORS ***
(SEE FILE ETPRE.WNG)

Stop - Program terminated.

Keyword Input File DEMO.EKI (File 1)

```
$ This file written on 01-28-1994 at 14:27:24
$ By the SETAC EVNTRE/PSTEVT User Interface
$   Version 1.01, Created January 4, 1994
$
$ Message Output Files for All Programs
$
LHSMMSG D:\ETPRE\EXAMPLE\DEMO.LMO
MESSAGE D:\ETPRE\EXAMPLE\DEMO.EMO
$ Input will be taken from this file:
$
PRETRIN D:\ETPRE\EXAMPLE\DEMO.PRE
POSTFL D:\ETPRE\EXAMPLE\DEMO.DPF
$
$ LHS and LHSPPOST Keywords
$
LHSTITL Run using ETPRE Example Input File
LHSOBS 15
LHSSEED 167438
LHSPVAL 1
LHSRPTS CORR HIST
LHSOUT D:\ETPRE\EXAMPLE\DEMO.LSP
LHSPPOST D:\ETPRE\EXAMPLE\DEMO.MSP
$
$ EVNTRE Processing Keywords
$
MODE 3
SAMMOD 10
SAMALLB
RUN
ADDSEQ
NOSORT
BIN
KEEPCUT .0000001
TREEIN D:\ETPRE\EXAMPLE\DEMO.ETI
BININ D:\ETPRE\EXAMPLE\DEMO.EBI
SAMDIN D:\ETPRE\EXAMPLE\DEMO.EPI
SAM1IN D:\ETPRE\EXAMPLE\DEMO.EVI
$
$ EVNTRE Output Keywords
$
WRTBIN
BINOUT D:\ETPRE\EXAMPLE\DEMO.EBO
STATS
STATOUT D:\ETPRE\EXAMPLE\DEMO.ESO
SAVEBIN
EVNTBIN D:\ETPRE\EXAMPLE\DEMO.EBF
$
ENDKEY
PRTCUT 0
$
```

ETPRE Tree Definition File DEMO.PRE (File 2)

Gas Pump Example Using Parameters

Conditional	1.00
Unconditional	0.04
Fixed-Threshold	1.00

\$ Questions 1-4 all use the same values for each sequence.

\$ Some variables are described at the bottom of the file.

Is pump #1 Damaged?

Pump1-OK	Pump1-Bad
(A)	(Excess)
Variable: Pump1-Flow	
0.0	(Gas-Flow-1)

Data: Gas-Flow-1 20.0 File D:\ETPRE\EXAMPLE\LHS.LSP
 Data: A1 0.6 LogUniform 0.4 0.8
 Data: Crash-Factor 1.0 Lognormal-B 0.8 1.2
 Data: A 0.6 = A1 * Crash-Factor

Is pump #2 Damaged?

Pump2-OK	Pump2-Bad
(B)	(Excess)
Variable: Pump2-Flow	
0.0	(Gas-Flow-2)

Data: B1 0.6 LogUniform 0.4 0.8
 Data: B 0.6 = B1 * Crash-Factor
 Data: Correlate A1 B1 0.75

Is pump #3 Damaged?

Pump3-OK	Pump3-Bad
(CD)	(Excess)
Variable: Pump3-Flow	
0.0	(Gas-Flow-3)

Is pump #4 Damaged?

Pump4-OK	Pump4-Bad
(CD)	(Excess)
Variable: Pump4-Flow	
0.0	(Gas-Flow-4)

Data: Gas-Flow-4 20.0 File D:\ETPRE\EXAMPLE\LHS.LSP

\$ This question samples the explosion threshold in the first two
 \$ sequences, and uses a constant threshold for the third sequence.

How big is the spill?

	Large	Small	No-Spill
GEThresh	Add(Pump1-Flow, Pump2-Flow, Pump3-Flow, Pump4-Flow)		
	(Exp-Threshold)	0.1	
	(Exp-Threshold)	0.1	
	[Exp-Threshold]	0.1	
Data:	Exp-Threshold	50.0	File D:\ETPRE\EXAMPLE\LHS.LSP

\$ This question determines what happens - fire, explosion, or nothing
 \$ If nothing happens, truncate the path so that only fire and explosion remain.
 \$ Thus, there is no "Nothing" branch on this question - it's all truncated.

Does an explosion or fire occur?

	Prob<1	Explosion	Fire
Large		(Explosion-Prob) (Excess)	
Small		0.00	(Fire-Prob)
Otherwise		0.00	0.00

Data: Explosion-Prob 0.7 Uniform 0.40 0.99
 Data: Fire-Prob 0.5 Uniform 0.2 0.8

\$ Now assign consequences based on the presence of an explosion or fire.
 \$ The "probabilities" for this event are really the \$ cost consequences of each outcome.
 \$ They are multiplied into the path "probability" to obtain a final result that is
 \$ really risk (probability times consequences).

What is the damage that occurs?

Prob>0	Consequences
Explosion	100000.0
Otherwise	12000.0

Binning: Gas station example

```
'Pump1 = Pump 1 Condition'
1 'Pump1-OK = OK'      Pump1-OK
2 'Pump1-Bad = Hit'    Pump1-Bad
'Pump2 = Pump 2 Condition'
1 'Pump2-OK = OK'      Pump2-OK
2 'Pump2-Bad = Hit'    Pump2-Bad
'Pump3 = Pump 3 Condition'
1 'Pump3-OK = OK'      Pump3-OK
2 'Pump3-Bad = Hit'    Pump3-Bad
'Pump4 = Pump 4 Condition'
1 'Pump4-OK = OK'      Pump4-OK
2 'Pump4-Bad = Hit'    Pump4-Bad
Result
1 'Explosion = Explode' Explosion
2 'Fire = Fire'          Fire
```

Sorting:

Pump1	Pump2	Pump3	Pump4	Result
Result				

Rebinning:

```
Result
1 'Explosion = Explode' Explosion
2 'Fire = Fire'          Fire
```

Dataset:

```
Gas-Flow-2 20.0 File D:\ETPRE\EXAMPLE\LHS.LSP
Gas-Flow-3 20.0 File D:\ETPRE\EXAMPLE\LHS.LSP
CD 0.2 Normal 0.2 0.03
```

\$ End

EVNTRE Tree Definition File DEMO.ETI (File 3)

Gas Pump Example Using Parameters

```
8
NQ
3 1.00000E+00 4.00000E-02 1.00000E+00
  CONDITIONAL UNCONDITIONAL FIXED-THRESH
1 Which sequence?
3 1.00000E+00 0.00000E+00 0.00000E+00
  0.00000E+00 1.00000E+00 0.00000E+00
  0.00000E+00 0.00000E+00 1.00000E+00
2 Is pump #1 Damaged?
2 PUMP1-OK PUMP1-BAD
3
  5.65467E-01 4.34533E-01
```

```

5.65467E-01 4.34533E-01
5.65467E-01 4.34533E-01
1
1 .000000 20.1789
.000000 20.1789
.000000 20.1789
3 Is pump #2 Damaged?
2 PUMP2-OK PUMP2-BAD
3
5.66112E-01 4.33888E-01
5.66112E-01 4.33888E-01
5.66112E-01 4.33888E-01
1
2 .000000 19.9123
.000000 19.9123
.000000 19.9123
4 Is pump #3 Damaged?
2 PUMP3-OK PUMP3-BAD
3
2.00813E-01 7.99187E-01
2.00813E-01 7.99187E-01
2.00813E-01 7.99187E-01
1
3 .000000 20.2479
.000000 20.2479
.000000 20.2479
5 Is pump #4 Damaged?
2 PUMP4-OK PUMP4-BAD
3
2.00813E-01 7.99187E-01
2.00813E-01 7.99187E-01
2.00813E-01 7.99187E-01
1
4 .000000 20.5952
.000000 20.5952
.000000 20.5952
6 How big is the spill?
3 LARGE SMALL NO-SPILL
5
4 1 2 3 4
PUMP1-FLOW PUMP2-FLOW PUMP3-FLOW PUMP4-FLOW
ADD
GETHRESH 2 58.3832 .100000
58.3832 .100000
58.3832 .100000
REQUIRED COMMENT FOR QUESTION 6
7 Does an explosion or fire occur?
2 EXPLOSION FIRE
10
3
1 6
1
LARGE
6.90573E-01 3.09428E-01
6.90573E-01 3.09428E-01
6.90573E-01 3.09428E-01
1 6
2
SMALL
0.00000E+00 4.99817E-01
0.00000E+00 4.99817E-01
0.00000E+00 4.99817E-01
OTHERWISE
0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00
8 What is the damage that occurs?
1 CONSEQUENCES
14
2

```

```

1      7
      1
EXPLOSION
1.00000E+05
1.00000E+05
1.00000E+05
OTHERWISE
1.20000E+04
1.20000E+04
1.20000E+04

```

Binning and Sorting Definition File DEMO.EBI (File 4)

Gas station example	SEQUENCE	PUMP1 RESULT	PUMP2 FIXED-THRESH	PUMP3
6	PUMP4	UNCONDITIONAL		
3	3			
1	1			
	1			
	CONDITIONAL			
1	2			
	1			
	2			
	UNCONDITIONAL			
1	3			
	1			
	3			
	FIXED-THRESH			
2	2	PUMP1-OK		
1	1			
	2			
	1			
	PUMP1-OK			
1	2			
	2			
	PUMP1-BAD			
2	2	PUMP2-OK		
1	1			
	3			
	1			
	PUMP2-OK			
1	2			
	3			
	2			
	PUMP2-BAD			
2	2	PUMP3-OK		
1	1			
	4			
	1			
	PUMP3-OK			
1	2			
	4			
	2			
	PUMP3-BAD			
2	2	PUMP4-OK		
1	1			
	5			
	1			
	PUMP4-OK			
1	2			
	5			
	2			
	PUMP4-BAD			
2	2	EXPLOSION		
1	1			
	7			
	1			
	EXPLOSION			
1	2			
	7			
	2			
	FIRE			
0				

Rebinning Definition File DEMO.DPF (File 5)

```

N/A
2
3 3 SEQUENCE      RESULT      UNCONDITIONAL  FIXED-THRESH
1 1 1
1 1 1
1 2 1
2 2 2
1 3 1
2 2 3
1 1 6
1 2 1
2 2 6
1 2 2
2 2 1
6 1 2
5 6 3 4
SEQUENCE      PUMP1      PUMP2      PUMP3
PUMP4      RESULT
2 1 6
SEQUENCE      RESULT
1
3
1 'CONDITIONAL'
2 'UNCONDITIONAL'
3 'FIXED-THRESH'
2
1 'Explode'
2 'Fire'
$
$ This file was written on 02/10/94 at 12:34:58 by
$ ETPRE Version 3.00, compiled with Microsoft V 5.1.
$ Tree Definition File is D:\ETPRE\EXAMPLE\DEMO.PRE.
$ Keyword File is DEMO.EKI.
$
@BIN
$$ EVNTRE/PSTEVNT Binning Labels Version 1.00
$$ Title Gas station example
$$ Dimensions: 6
$$ Dim 1 3 SEQUENCE
$$ 1 A CONDITIONAL
$$ 1 B UNCONDITIONAL
$$ 1 C FIXED-THRESH
$$ Dim 2 2 Pump 1 Condition
$$ 2 A OK
$$ 2 B Hit
$$ Dim 3 2 Pump 2 Condition
$$ 3 A OK
$$ 3 B Hit
$$ Dim 4 2 Pump 3 Condition
$$ 4 A OK
$$ 4 B Hit
$$ Dim 5 2 Pump 4 Condition
$$ 5 A OK
$$ 5 B Hit
$$ Dim 6 2 RESULT
$$ 6 A Explode
$$ 6 B Fire
$$ Done
@REBIN
$$ EVNTRE/PSTEVNT Binning Labels Version 1.00

```

```

$$ Title N/A
$$ Dimensions: 2
$$ Dim 1 3 SEQUENCE
$$ 1 A CONDITIONAL
$$ 1 B UNCONDITIONAL
$$ 1 C FIXED-THRESH
$$ Dim 2 2 RESULT
$$ 2 A Explode
$$ 2 B Fire
$$ Done

```

Sample Definition File DEMO.EPI (File 6)

FEB 10, 1994 @12:34

15 1

13

```

P2 1 1 2 TTT M2 1 1 2 TTT P3 1 1 2 TTT M3 1 1 2 TTT
P4 1 1 2 TTT M4 1 1 2 TTT P5 1 1 2 TTT M5 1 1 2 TTT
C6 1 1 TFF C6 1 1 TFF P7 1 1 0 TTT P7 1 2 0 TTT
P7 2 2 0 TTT

```

Sample Vector File DEMO.EVI (File 7)

1	13	3.784800E-01	10.040600	3.887540E-01
21.720500	2.332350E-01	20.361900	2.332350E-01	
8.817790	72.956900	72.956900	8.804870E-01	
1.195130E-01	4.091540E-01			
2	13	4.875210E-01	42.806900	4.636910E-01
32.923600	1.856510E-01	15.830600	1.856510E-01	
16.466600	34.946100	34.946100	9.147510E-01	
8.524900E-02	6.098890E-01			
3	13	6.356060E-01	16.082000	5.722520E-01
14.305500	2.449650E-01	12.009800	2.449650E-01	
26.127800	55.351100	55.351100	7.911870E-01	
2.088130E-01	7.328630E-01			
4	13	4.896430E-01	34.159800	5.904340E-01
19.902500	2.032290E-01	42.385600	2.032290E-01	
50.298600	74.397100	74.397100	9.690480E-01	
3.095200E-02	3.878710E-01			
5	13	6.034290E-01	21.038000	7.532000E-01
7.864010	1.413190E-01	29.076400	1.413190E-01	
11.307400	20.511300	20.511300	5.603190E-01	
4.396810E-01	5.002000E-01			
6	13	5.780570E-01	17.854400	4.432720E-01
22.525800	1.953350E-01	21.627800	1.953350E-01	
25.984100	30.720100	30.720100	6.031910E-01	
3.968090E-01	4.735270E-01			
7	13	4.361840E-01	8.015580	4.853430E-01
15.377800	1.874880E-01	14.804400	1.874880E-01	
16.070300	46.326400	46.326400	4.571050E-01	
5.428950E-01	7.054150E-01			
8	13	7.298090E-01	14.078100	5.411220E-01
10.082400	1.729140E-01	13.108300	1.729140E-01	
13.142400	234.315000	234.315000	7.348100E-01	
2.651900E-01	7.770310E-01			
9	13	4.961050E-01	2.659400	6.031240E-01
17.196400	2.017340E-01	31.785000	2.017340E-01	
20.548500	51.118600	51.118600	8.389010E-01	
1.610990E-01	5.638170E-01			
10	13	6.803330E-01	16.613100	7.576270E-01
18.097200	2.568540E-01	24.065800	2.568540E-01	

14.553300	40.542700	40.542700	6.778050E-01
3.221950E-01	5.579320E-01		
11	13	4.830380E-01	12.793300 4.501640E-01
25.391000	1.771940E-01	22.695300	1.771940E-01
32.387700	27.208800	27.208800	4.919360E-01
5.080640E-01	2.091510E-01		
12	13	4.405000E-01	24.067800 4.696260E-01
13.676400	2.239050E-01	10.453500	2.239050E-01
19.179000	48.982000	48.982000	4.064470E-01
5.935530E-01	6.607690E-01		
13	13	6.008650E-01	20.312500 5.497220E-01
40.319900	2.161140E-01	18.789500	2.161140E-01
18.017700	60.149100	60.149100	6.655290E-01
3.344710E-01	2.583870E-01		
14	13	8.386230E-01	27.455900 7.870090E-01
27.039800	2.089160E-01	17.306700	2.089160E-01
23.278800	41.664100	41.664100	5.491960E-01
4.508040E-01	3.042730E-01		
15	13	6.038150E-01	13.706300 6.363400E-01
12.261500	1.633420E-01	9.418760	1.633420E-01
12.747300	36.558000	36.558000	8.178740E-01
1.821260E-01	3.469850E-01		

LHS Output File DEMO.LSP (File 8)

```

$ LHS File Format Version 1.00
$
$ This LHS run was executed on 02/07/94 at 14:17:39.41
$ with LHS Version: 2.00 Release 1, Compiled 01/28/1994
$ The run title was:
$ RUN USING ETPRE EXAMPLE INPUT FILE
$
$ Message output file for this run: C:\EVNTRE\EXAMPLES\DEMO.LMO
$
$ Input file(s) for this run: C:\EVNTRE\EXAMPLES\DEMO.EKI
$ and C:\EVNTRE\EXAMPLES\DEMO.PRE
$
$ An input correlation matrix was specified.
$
$ Point Values for the distributions follow:
$
$ All point values represent mean values that
$ were calculated from the actual LHS sample.
$
  A1          5.7611424E-01
  CRASH-FACTOR 9.8138213E-01
  B1          5.7657743E-01
  EXPLOSION-PROB 6.9057250E-01
  FIRE-PROB 4.9981743E-01
  CD          2.0081297E-01
$
@UNCERTAINTY
@OBSERVATIONS 15
@VARIABLES 6
  A1:
  CRASH-FACTOR:
  B1:
  EXPLOSION-PROB:
  FIRE-PROB:
  CD:
@SAMPLEDATA
  1          6 0.400744 0.944443 0.411622
  0.880487 0.409154 0.233235
  2          6 0.488920 0.997139 0.465021
  0.914751 0.609889 0.185651

```

3	6	0.650541	0.977042	0.585698
0.791187	0.732863	0.244965		
4	6	0.456280	1.07312	0.550203
0.969048	0.387871	0.203229		
5	6	0.593371	1.01695	0.740646
0.560319	0.500200	0.141319		
6	6	0.562192	1.02822	0.431106
0.603191	0.473527	0.195335		
7	6	0.468006	0.932006	0.520751
0.457105	0.705415	0.187488		
8	6	0.757748	0.963129	0.561837
0.734810	0.777031	0.172914		
9	6	0.549547	0.902753	0.668094
0.838901	0.563817	0.201734		
10	6	0.700580	0.971099	0.780175
0.677805	0.557932	0.256854		
11	6	0.527062	0.916472	0.491192
0.491936	0.209151	0.177194		
12	6	0.423330	1.04056	0.451320
0.406447	0.660769	0.223905		
13	6	0.684281	0.878097	0.626038
0.665529	0.258387	0.216114		
14	6	0.766272	1.09442	0.719111
0.549196	0.304273	0.208916		
15	6	0.612837	0.985279	0.645847
0.817874	0.346985	0.163342		

LHS Postprocessor Output File DEMO.MSP (File 9)

```

$ LHS File Format Version 1.00
$
$ This file was created by an LHSPPOST run on 02/07/94 at 14:17:44.19
$ with LHSPPOST Version: Version 0.01, May 18, 1993
$
$ Input file(s) for this run: C:\EVNTRE\EXAMPLES\DEMO.EKI
$ and: C:\EVNTRE\EXAMPLES\DEMO.PRE
$
$ Sampled data was read from the following file(s):
$ C:\EVNTRE\EXAMPLES\DEMO.LSP
$ C:\EVNTRE\EXAMPLES\LHS.LSP
$
$ The point estimates are mean values.
$
      A          0.565467
      B          0.566112
$
@UNCERTAINTY
  @OBSERVATIONS      15
  @VARIABLES         2
    A:
    B:
@SAMPLEDATA
      1          2  0.378480      0.388754
      2          2  0.487521      0.463691
      3          2  0.635606      0.572252
      4          2  0.489643      0.590434
      5          2  0.603429      0.753200
      6          2  0.578057      0.443272
      7          2  0.436184      0.485343
      8          2  0.729809      0.541122
      9          2  0.496105      0.603124
     10          2  0.680333      0.757627
     11          2  0.483038      0.450164
     12          2  0.440500      0.469526
     13          2  0.600865      0.549722
     14          2  0.838623      0.787009
     15          2  0.603815      0.636340

```

LHS User File LHS.LSP (File 10)

```

$ LHS File Format Version 1.00
$
$ This LHS run was executed on 01/28/94 at 13:09:44.88
$ with LHS Version: 2.00 Release 1, Compiled 01/28/1994
$ The run title was:
$ AUX LHS FILE FOR ETPRE EXAMPLE
$
$ Message output file for this run: C:\EVNTRE\EXAMPLES\LHS.LMO
$
$ Input file(s) for this run: C:\EVNTRE\EXAMPLES\LHS.EKI
$ and C:\EVNTRE\EXAMPLES\LHS.INP
$
$ An input correlation matrix was specified.
$
$ Point Values for the distributions follow:
$
$ All point values represent mean values that
$ were calculated from the actual LHS sample.
$
EXP-THRESH          5.8383167E+01
GAS-FLOW-1          2.0178921E+01
GAS-FLOW-2          1.9912302E+01
GAS-FLOW-3          2.0247946E+01
GAS-FLOW-4          2.0595152E+01
$
@UNCERTAINTY
@OBSERVATIONS      15
@VARIABLES         5
EXP-THRESH:
GAS-FLOW-1:
GAS-FLOW-2:
GAS-FLOW-3:
GAS-FLOW-4:
@SAMPLEDATA
1           5      72.9569      10.0406      21.7205
20.3619     8.81779
2           5      34.9461      42.8069      32.9236
15.8306     16.4666
3           5      55.3511      16.0820      14.3055
12.0098     26.1278
4           5      74.3971      34.1598      19.9025
42.3856     50.2986
5           5      20.5113      21.0380      7.86401
29.0764     11.3074
6           5      30.7201      17.8544      22.5258
21.6278     25.9841
7           5      46.3264      8.01558      15.3778
14.8044     16.0703
8           5      234.315      14.0781      10.0824
13.1083     13.1424
9           5      51.1186      23.6594      17.1964
31.7850     20.5485
10          5      40.5427      16.6131      18.0972
24.0658     14.5533
11          5      27.2088      12.7933      25.3910
22.6953     32.3877
12          5      48.9820      24.0678      13.6764
10.4535     19.1790
13          5      60.1491      20.3125      40.3199
18.7895     18.0177
14          5      41.6641      27.4559      27.0398
17.3067     23.2788
15          5      36.5580      13.7063      12.2615
9.41876     12.7473

```

Appendix G - LHS File Format

Format for LHS Output and LHS-Like Files

Specification Version 1.00, June 11, 1993

1. Overview

The file format consists of three blocks: the point estimate data block, the uncertainty header block, and the uncertainty data block. All three blocks must be present in the file in the order listed above. The point estimate data block starts at the beginning of the file and proceeds until the keyword "@UNCERTAINTY" is found as the first item on the line. This marks the beginning of the uncertainty header block. This header block ends with the keyword "@SAMPLEDATA". The uncertainty data block follows this keyword and occupies the rest of the file. A copy of the top portion of an example file is attached.

2. Common Characteristics of the Point Estimate and Uncertainty Header Blocks

- 1) All data fields are space-delimited. All processors should treat commas and tab characters as if they were blank spaces.
- 2) A data record may occupy multiple lines, but each line may have no more than 80 characters. Characters that occur in columns greater than 80 are ignored. If a data record is to be continued on the following line, it must contain a continuation character (either a "#" or a "%") as the last character prior to any trailing blanks or comments (see below). The continuation character must be preceded by one or more blank spaces.
- 3) All text following a "\$" (Dollar Sign) is considered a comment. If a Dollar Sign is the first item on a data record, the entire record is treated as a comment. A Dollar Sign and comment may also follow all data items on a record or the continuation character (trailing comment). In these cases, the Dollar Sign must be preceded by one or more blank spaces. Comments may not come between data items on a single record (except following a continuation character on a continued line).
- 4) Blank lines are considered comment records and are ignored. Any number of blank lines and/or comment lines may be placed between a line and its continuation or between consecutive continuation lines.

- 5) All names used in the file must meet the following criteria: a) each name must contain no more than 16 characters; b) the "\$" (Dollar Sign), "#" (Number Sign), "%" (Percent Sign), "," (Comma), " " (Blank Space), and Tab characters are not valid within names; and c) a name must not form a valid representation of a real number for an ANSI-standard Fortran compiler. All names are to be treated as case-insensitive, so the names "FRED", "Fred", "fred" and "freD" are all equivalent. Names that are longer than 16 characters may be truncated to 16 characters at the discretion of the processor.
- 6) Where numbers are specified, they may be in any format recognized by an ANSI-standard Fortran compiler.
- 7) Each name must be unique within each section. Duplicate definitions in the same section for a single data item are not valid. However, an item may be defined once in each section. If an item is defined once in the point estimate block and once in the uncertainty header block, both definitions are effective. The reading program would have to decide whether its calculations should use the point estimate value, the distribution, or both.
- 8) A line and all of its continuation lines (with all trailing and intervening comments eliminated) must together consist of no more than 32767 characters.

3. Specifications for the Point Estimate Data Block

The point estimate data block must be the first section in the file. It should begin with a single comment record (starting with a Dollar Sign) that documents the version of the LHS file format being used. The record should be exactly as follows:

\$ LHS File Format Version 1.00

The record need not begin in the first column. Any program wishing to read this record (in order to verify input file compatibility) may assume that only one blank character exists between each of the words, but should not make any assumptions regarding which characters will be in upper or lower case.

The version specification comment record should be followed by several additional comment records that document the pedigree of the file, such as the name and version of the program that created it, the date and time that the program was run, and the list of files accessed by the program.

A data record in the point estimate data block consists of anything that is not discarded as a comment according to the format described in Section 2 above. Only one type of data record is permissible in the point estimate data block. A data record consists of a

list of one or more data item names followed by one or two real number values. The first of these values represents the point value for each of the items found in the list of named data items. The second item, which is optional, is the standard deviation associated with the point estimate value. The point estimate value that appears at the end of the line is assigned to all named data items on that line (including all active continuation lines). The list is space-delimited (see Section 2 above).

Any processor reading this data format should look for the following common errors:
a) a record consisting of a list of names that is not followed by a legal value is illegal;
b) a record consisting of a value without a corresponding list of names is illegal; c) names that follow the value(s) on a record are illegal.

Any processor wishing to skip this section can simply read and discard all lines until the keyword "@UNCERTAINTY" is found as the first data item on a non-comment line (@UNCERTAINTY may be preceded by leading blank spaces).

4. Specifications for the Uncertainty Header Block

The uncertainty header block contains two input sections which may occur in either possible order. The first section consists of a single line containing the keyword "@OBSERVATIONS" as its first data item, followed by a positive integer that represents the number of observations that are found in the uncertainty data block. It is critical that the value on this record accurately reflect the true number of observations present in the file. Any processor reading this data format must be sure that there are at least this number of observations present in the file. It may, however, ignore any additional observations that may be present.

The second section describes the number and names of the distributions that are found in the uncertainty data block. The section begins with a single line containing the keyword "@VARIABLES" as its first data item, followed by a positive integer that represents the number of distributions that are found in the uncertainty data block. It is critical that the value on this record accurately reflect the true number of variables present in the file. Any processor reading this data format must be sure that there are at least this number of distributions present in the file. It may, however, ignore any additional distributions that may be present.

The @VARIABLES *n* line must be followed by exactly *n* lines that provide names for the *n* distributions. Each of these lines begins with the primary name for the distribution, followed by a ":" (Colon Character). Following the Colon Character is an optional list of secondary names for the distribution. Thus, each of the *n* distributions is given exactly one primary name and an arbitrary number of secondary names. The list of names is space-delimited and may extend to one or more continuation lines as

described in Section 2. Comments may be present at any point in this block as described in Section 2.

The order in which these records appear is significant. All names on the first record encountered in this section are assumed to correspond to the first distribution in the uncertainty data section of the file. The names on the second record are assumed to correspond to the second distribution, and so forth. For this reason, the number of records found in this section must correspond exactly to the number that follows the `@VARIABLES` keyword, and that value must exactly match the number of distributions found in the uncertainty data block. Anything else is an error condition.

The uncertainty header block is terminated by the `"@SAMPLEDATA"` keyword.

5. Specifications for the Uncertainty Data Block

The uncertainty data block is completely different from either of the preceding blocks. It contains only numeric data (no names). Comments lines are not supported in this block. Continuation lines exist only as described below.

This block contains one data record for each observation. The data record for a file containing n variables consists of two integers followed by n real values. The first integer value contains the observation number. The second integer value contains the number of distribution values that follow (should be n). These integers are followed by one real value that has been selected from each of the n distributions. One such record is present for each observation.

The records in this section are generally written and read using a FORTRAN list-type format in an implied "Do Loop". Thus, one record may consist of several lines without explicit continuation characters. The FORTRAN program will read additional lines as needed. The following hypothetical code example would read one entire uncertainty data block (all variables and all observations):

```
Do i=1, NObs
    Read (1,*) i, NVar, (Values(i,j), j=1, NVar)
End Do
```

The block is typically written to the file using similar code. Note that the use of the FORTRAN `"*"` format for reading and writing does not allow for continuation characters or comment lines. Thus, these are not allowed in this data block.

Note that this block occupies the remainder of the file. Thus, an attempt to read this file following the completion of the above loop should result in an end-of-file error condition.

Distribution:

0405	D. D. Carlson,	6411	0718	T. A. Wheeler,	6641	
0405	A. S. Benjamin,	6411	0736	N. R. Ortiz,	6400	
0405	D. E. Bennett,	6411	0746	J. E. Campbell,	6613	
0405	Ricardo Beraun,	6411	0747	A. L. Camp,	6412	
0405	R. J. Breeding,	6411	0747	S. L. Daniel,	6412	
0405	N. N. Brown,	6411	0747	V. J. Dandini,	6412	
0405	T. R. Jones,	6411	0747	S. E. Dingman,	6412	
0405	K. J. Maloney,	6411	0747	Jacquelin Pfarner,	6412	
0405	K. B. Sobolik,	6411	0747	D. W.	6412	
0425	A. C. Payne,	4115		Whitehead,		
0457	V. J. Johnson,	5600	0747	G. D. Wyss,	6412	(30)
0490	S. D. Spray,	12331	0748	F. T. Harper,	6413	
0490	R. Bennett,	12331	0748	T. D. Brown,	6413	
0491	R. E. Smith,	12333	0748	S. J. Higgins,	6413	
0491	P. N. Demmie,	12333	0748	L. A. Miller,	6413	
0491	M. A. Dvorack,	12333	0763	D. B. Clauss,	9614	
0491	N. R. Grandjean,	12333	0830	D. L. Wright,	12335	
0491	J. V. Hancock,	12333	1141	L. F. Restrepo,	6523	
0491	S. A. Kalemba,	12333	1393	M. K. Fuentes,	5609	
0491	Y. T. Lin,	12333				
0491	R. J. Roginski,	12333				(20)
0491	J. L. Tenney,	12333	9018	Central Technical	8523-2	(3)
0491	R. O.	12333		Files,		
	Winchester,		0899	Technical	7141	(5)
0492	Lutz Dahlke,	12324		Library,		
0492	R. D. Pederson,	12324	0619	Technical	7151	
0492	P. E. D'Antonio,	12324		Publications,		
0492	John Berry,	12324	0100	Document	7613-2	(10)
0492	J. F. Wolcott,	12332		Processing for		
0492	C. G. Shirley,	12332		DOE/OSTI		

DATE

FILMED

7/6/94

END