DESIGN AND ADAPTATION OF A GENERAL PURPOSE, USER FRIENDLY
STATISTICAL SOFTWARE PACKAGE
FOR THE IBM PERSONAL COMPUTER AND IBM PC COMPATIBLES
(PC VSTAT)

A Thesis Presented to
The Faculty of the College of Engineering and Technology
Ohio University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

by
Deborah G. Morley
November 26, 1986

OHIO UNIVERSITY
LIBRARY
Acknowledgements

I wish to express my deep appreciation for the help and constructive criticisms of my advisor, Dr. Helmut T. Zwahlen. Also, I would like to thank my friends Rezvan Rashidianfar and Mary Hayes for their help and support.
Abstract

The purpose of this thesis is to describe the design and adaptation of a general purpose, user friendly statistical package for the IBM Personal Computer and IBM PC compatibles. This thesis is based on Thomas R. Balent's thesis, (in which the VSTAT computer programs were originally written in the FORTRAN IV language for the HP-1000 minicomputer), as well as Jitti Kongsupapsiri's adaption of VSTAT for the Apple computer and Kijja Laoboonchai's adaption for the Macintosh computer. Both APPLE VSTAT and MAC VSTAT were written in BASIC.

In this thesis, new features were added to the original package and the VSTAT programs were written in Pascal, resulting in a new software package called PC VSTAT. Programs are provided for both the source and compiled versions for the IBM Personal Computer and IBM PC compatibles.
# TABLE OF CONTENTS

## I. Problem Definition
A) Introduction ........................................... 1
B) Objectives ............................................. 2

## II. Design Definition
A) Background ............................................. 4
B) General Requirements ................................. 5
C) Program Requirements ................................. 7
D) Alternatives ........................................... 14
   1. Operating Systems .................................. 14
   2. Programming Languages ............................. 15

## III. System Design
A) PC Disk Operating System ............................. 20
B) Turbo Pascal ........................................... 20
C) Software Architecture .................................. 22
   1. Data Storage ........................................ 22
   2. System Integration .................................. 22
D) Features - Detailed Design ............................ 25
   1. Data Manipulation .................................. 26
   2. Descriptive Statistics ............................... 34
   3. Hypothesis Testing .................................. 39

## IV. Validation ........................................... 50

## V. System Evaluation
A) Testing ................................................. 75
B) Conclusion ............................................. 75
C) Future Recommendations ............................... 76

References .................................................. 78
Bibliography ............................................... 79

Appendices
A) Equations Used ....................................... 81
B) User’s Manual ......................................... 90
C) Program Flow Charts ................................. 101
D) Program Listings ..................................... 126
CHAPTER 1
PROBLEM DEFINITION

A) Introduction

The purpose of my thesis project is to adapt and modify a statistical software package (VSTAT) for the IBM Personal Computer and IBM PC compatibles, and add some new features to the package.

In August 1982, Thomas R. Balent developed a general purpose user friendly statistical package that was installed and implemented on the HP-1000. The system, called VSTAT (Vector STATistics), has been used by both students and faculty of the College for tasks ranging from the solution of homework problems to assisting with research projects. However, the need has arisen within the College for a general purpose user friendly statistical software package that would operate on personal computers.

The VSTAT software that had originally been written in FORTRAN IV for the HP-1000 minicomputer was converted to the Applesoft BASIC language by Jitti Kongsupapsiri in October 1984 to run on every model of the Apple II microcomputer. This package was called APPLE VSTAT. Another adaption of VSTAT was done for the Macintosh microcomputer. This software package (MAC VSTAT) was written in the Microsoft BASIC language by Kijja Laoboonchai in October 1985.
This thesis project combines knowledge in the area of statistical computation with the features of the IBM Personal Computer and the Turbo Pascal language, to produce a general purpose, user friendly statistical software package.

Following a brief introduction, the system requirements, alternatives and design approach are discussed. Validation of the package and an overall system evaluation are included. Information concerning the equations used, program listings and basic program flowcharts can be found in the appendices. A user's manual is also provided in the appendix section.

B) Objectives

This thesis adapts VSTAT for the IBM Personal Computer and provides the following new features:

1. The Kolmogorov-Smirnov two-sample test has been expanded to handle 150 data points per vector.

2. The Chi-Square goodness-of-fit test has been extended to test if the data of interest fits a negative exponential probability density function in addition to the normal probability density function.

3. The Kolmogorov-Smirnov one-sample goodness-of-fit test has been extended to test if the data
of interest fits a negative exponential probability density function in addition to the normal probability density function.

4. The section on Correlation Coefficient has been expanded to test if rho, the correlation coefficient, is significantly different from a value specified by the user, in addition to testing against zero.

5. An option has been provided that allows the user to select between storage of the data vectors on a floppy diskette (specified as disk drive B:), or on hard disk (specified as disk drive C:).

The design and adaptation of a general purpose, user friendly statistical package for the IBM personal computer and IBM PC compatibles (PC VSTAT) provides a useful and convenient tool for performing statistical analyses in research labs, statistics classes, offices or homes.
CHAPTER II

DESIGN DEFINITION

This section provides a discussion of the demand for PC VSTAT, the general and specific requirements for the system, and an evaluation of some alternatives.

A) Background

While the Engineering College has had access to a number of computerized statistical packages via the University's IBM mainframe (i.e., S.P.S.S. and S.A.S.), these packages are much too extensive and learning how to operate them can be cumbersome and time consuming for the casual user. This led to the development of VSTAT for the HP-1000. With the increasing availability of low-cost microcomputers, the need arose to perform statistical analyses without the use of mainframe or minicomputers. Since the IBM Personal Computer has become one of the more popular and most used microcomputers, it would be convenient to have a statistical package similar to the HP-1000's VSTAT that would run on this type of system.

A number of general and specific requirements have been defined for the proposed package. The general criteria dictate how the overall system should be designed while the specific criteria define what functions the package should offer.
B) General Requirements

The general requirements upon which the system is designed are:

1) The system must operate on the IBM Personal Computer and IBM PC compatible computers.

2) The complete computer package must, when loaded into the computer's operating memory, fit into the 256K of memory available, the required configuration of the system.

3) Two floppy disk drive units or one floppy disk drive and one hard disk are required so that the user's data can be stored on media independent of the PC VSTAT diskette.

4) It is necessary that there be a provision to get a hard copy of the graphs, tables and results that are displayed on the terminal screen. The printer should be able to yield an equally detailed output to that displayed on the screen.

5) The package must be stored in one diskette under the name VSTAT, and a formatted diskette must be provided to store data if the system supports a second floppy disk drive. If the second drive is a hard disk, the data must be stored there. (Data
could then be copied onto a floppy diskette if desired).

6) A menu format design is necessary for the package, as it is assumed that the user has had no previous computer programming experience. (It is assumed that the user is computer literate; he/she knows how to turn on the PC and knows what a disk drive and diskette are.)

7) External user instructions for the operation of the system should be kept to a minimum.

8) The data should be stored in data vectors that when necessary, will allow for a maximum of 1000 data points per vector.

9) Use of external tables for hypothesis testing should be eliminated by having programs perform these tasks.

10) The program listings should be well documented.

11) The package should supply the user with descriptive help both in error handling and system operation via a description section provided by PC VSTAT.

12) The package should utilize an operating system and be written in a language that provides for fast
execution time and computation speed.

C) Program Requirements

The user should be able to perform the following tasks:

1) Name and create vectors into which data will be stored for use by the other programs in the package. The user should be able to enter raw or grouped data, depending on the source of the data, and signal the end of data entry by typing END. Raw data would be considered the data collected from experimentation. Grouped data may be considered data that has been previously grouped according to some attribute.

2) Edit (add, remove or change) the data within the vector.

3) List the contents of each data vector.

4) Sort a data vector in ascending or descending order and store the sorted data in a new vector.

5) Perform mathematics with two data vectors \((+, -, \times, /)\) and store the result in a third vector. In addition, the user should be able to perform mathematics on a vector of data with a constant value (scalar) or determine the square root, natural
logarithm or logarithm base 10, of the data in the vector and store the result in a second vector.

6) Perform statistical measures on a data vector. This includes calculating the mean, variance, standard deviation, absolute mean deviation, as well as the range, minimum and maximum value of the data vector.

7) Calculate the correlation coefficient for two data vectors.

8) Produce a simple linear regression, quadratic regression and cubic regression for two data vectors.

9) Produce a scatter plot of the data contained in two data vectors.

10) Produce a frequency table and histogram of the data stored in a vector.

11) Produce the area to the right of the statistic values: Z, F, t and Chi-Square.

12) Perform random sampling and generate random numbers and store the output in data vectors.

13) Purge data vectors from the system when they are no longer needed.

14) Perform the following statistical tests:
a) A t-test in the case where the sample size, the sample means and the sample variances of the two independent samples are available to test whether or not the population means can be considered equal.

b) A t-test on two vectors of data to determine whether or not the population means can be considered to be equal.

c) A paired t-test on two vectors of data to test whether or not the difference in the population means is equal to zero.

d) An F-test in the case where the number of data points in the vector \((n)\) and the sample variances of two samples are available to test whether or not the population variances can be considered to be equal.

e) An F-test on two vectors of data to determine whether or not the population variances can be considered equal.

f) A Chi-square goodness-of-fit test for determining how well a vector fits the normal or negative exponential distribution.

g) A Kolmogorov-Smirnov one-sample test to determine how well a vector of data fits the normal or negative exponential distribution.

h) A Kolmogorov-Smirnov two-sample test to
determine whether or not the distribution of two data vectors can be considered to be equal. The two vectors may have a maximum of 150 data points per vector.

Figures 2.1 through 2.3 on the following pages display in menu format, the options made available by PC VSTAT.
PC VSTAT
STATISTICAL PACKAGE
COPYRIGHT OHIO UNIVERSITY 1986
OPTIONS AVAILABLE
-PAGE 1-

1 DESCRIPTION OF MENU ITEMS
2 CREATE A VECTOR
3 LIST A VECTOR
4 EDIT A VECTOR
5 MATH WITH TWO VECTORS
6 MATH WITH A VECTOR AND A CONSTANT
7 SORT A VECTOR
8 STATISTICAL MEASURES
9 CORRELATION COEFFICIENT FOR TWO VECTORS
10 LINEAR REGRESSION OF TWO VECTORS
11 PAGE -2- OF THIS MENU
12 PAGE -3- OF THIS MENU
13 QUIT (EXIT THIS PACKAGE)

Enter the number of the option you desire and press RETURN:

Figure 2.1 - Main Menu Page-1-
PC VSTAT
STATISTICAL PACKAGE
COPYRIGHT OHIO UNIVERSITY 1986
OPTIONS AVAILABLE
-PAGE 2-

1 NONLINEAR REGRESSION OF TWO VECTORS
2 FREQUENCY TABLE FOR A VECTOR
3 HISTOGRAM FOR A VECTOR
4 SCATTER GRAPH
5 F-TEST (n and sigma-squared known)
6 F-TEST
7 t-TEST (n, mu and sigma-squared known)
8 t-TEST
9 PAIRED t-TEST
10 CHI-SQUARED "goodness-of-fit" TEST
11 PAGE -1- OF THIS MENU
12 PAGE -3- OF THIS MENU
13 QUIT (EXIT THIS PACKAGE)

Enter the number of the option you desire and press RETURN:
OPTIONS AVAILABLE

1. KOLMOGOROV-SMIRNOV ONE-SAMPLE TEST
2. KOLMOGOROV-SMIRNOV TWO-SAMPLE TEST
3. ENTRY OF GROUPED DATA
4. VALUE FOR Z, F, t AND CHI-SQUARE
5. RANDOM NUMBER GENERATOR
6. RANDOM SAMPLING
7. PURGE A VECTOR
8. CHANGE DATA DISK-DRIVE
9. PAGE -1- OF THIS MENU
10. PAGE -2- OF THIS MENU
11. QUIT (EXIT THIS PACKAGE)

Enter the number of the option you want and press RETURN.
D) Alternatives

It is necessary that an appropriate operating system and programming language be selected in order to achieve the performance and provide the functions defined in the system requirements.

1. Operating Systems

Because the operating system contains the information that tells the computer how to handle its basic functions, it is important that alternative operating systems be investigated to provide the desired functionality.

a) PC-DOS/MS-DOS - This disk operating system is referred to as PC-DOS when implemented on the IBM PC. When used on IBM PC compatibles it is called MS-DOS, which was developed by MicroSoft for the 8088-based IBM PC. DOS is fast at loading files and uses little memory, only 12,143 bytes. It is a "user friendly" operating system that provides informative, understandable error messages. DOS operates efficiently and is difficult to "crash".

b) CP/M-86 - This is the 8086-compatible version of the CP/M-80 operating system. While CP/M-86 is considered to be the most universal of nonstandard software, it is hampered by its unclear documentation and is unfriendly in
operation. Manuals quote that an average of 20 to 40 hours of computer use are required to become proficient in using CP/M.

c) P-System - This operating system was developed at the University of California, San Diego (UCSD) campus. Because the P-System is written in UCSD Pascal, it depends on the P-machine emulator to translate pseudo-code into executable CPU instructions. The compilation of Pascal programs results in pseudo-instructions that are translated by an interpreter. The P-System offers the most transportability between completely different computer systems. Although P-System programs are small, their operating speed is slow because of the constant interpreting of the operating system's programs (which allows the programs to run and store in very small amounts of disk space) but degenerates the operating speed.

2) Programming Languages

In addition to selecting an appropriate operating system, it is necessary to choose a programming language that will provide file and data structures to meet the requirements outlined for the IBM PC VSTAT package.
a) BASIC

This is the most commonly used high-level language for microcomputers and the simplest. BASIC is a nonstructured language whose phrasing of commands, or syntax, is less strenuous than that of other languages, which may encourage sloppy programming habits. It is a general-purpose programming language that is used for a variety of needs.

b) Pascal

This is a highly structured language that is organized in blocks of code. Pascal allows the user to define their own data types and even create new types of variables to fit their own needs. It offers a variety of data structures and is rich in control statements.

c) FORTRAN

This language was developed for solving large-scale numerical problems that occur in mathematical, scientific and engineering applications. FORTRAN is a structured language and provides efficient execution. This language supports only three specific data types. There are also restrictions that apply to the naming of variables.
d) "C"

The C programming language is known best for its high transportability. It is a powerful language which allows programmers to construct complicated operations with a minimum of commands. C is a structured language having syntax that encourages a program to consist of stand-alone functions. It is unusual among programming languages in the breadth of its possible applications.

Figures 2.4 and 2.5 summarize the characteristics of the programming languages and operating systems that were investigated.
<table>
<thead>
<tr>
<th>Feature</th>
<th>PC-DOS/MS-DOS</th>
<th>CP/M-86</th>
<th>P-System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating Speed</td>
<td>FAST</td>
<td>AVERAGE</td>
<td>SLOW</td>
</tr>
<tr>
<td>Memory Utilization (under 15,000 bytes)</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>User-friendly</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Documentation</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Available</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
</tbody>
</table>

Figure 2.4 - Operating Systems
<table>
<thead>
<tr>
<th>Feature</th>
<th>BASIC</th>
<th>Pascal</th>
<th>FORTRAN</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Dynamic Data Structures</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>Efficient Execution</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Restricted Data types</td>
<td>YES</td>
<td>NO</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Available</td>
<td>YES</td>
<td>YES</td>
<td>YES</td>
<td>NO</td>
</tr>
</tbody>
</table>

Figure 2.5 - Programming Languages
CHAPTER III
DESIGN OF THE SYSTEM

This section defines the selected operating system and programming language for PC VSTAT. The type of data storage techniques used and the system integration is also discussed. Finally, a detailed design of each of the features of the package is provided.

A) The Operating System

PC-DOS was chosen as the preferred operating system because of its operation speed, small memory requirement, ease of use and availability.

B) Turbo Pascal

The attraction to the Pascal programming language is do to its ability to overcome some of the limitations of the traditional programming languages and that it offers dynamic variable storage structures that permit the size of a data vector to be changed while the program is running.

In response to the growing popularity of personal computers, Borland International has developed Turbo Pascal which provides a friendly interactive environment and extremely fast compilation and execution times. (See Chapter 5, "Conclusions", for user experiences and timings).
The figure below defines the hardware and software tools used in the design of PC VSTAT.

Figure 3.1 - The Computer System
C) Software Architecture

1. Data Storage

The most critical feature of a statistical package is how the data vectors used by the package are to be stored. The design and size of the package's programs are influenced by the type of data structure employed.

A very practical data structure is the variable length file. This type of structure allows for a storage area only as large as required. This is possible because the file is increased in size as data is added to it. The only limitation to the size of the structure is the available space on the data storage media, either floppy diskette or hard disk. Another important feature of the file structure is that it does not affect the size of the programs that utilize it. The files are external to the programs and, therefore, the program does not have to make allowances for the files. As a result, large data files can be used and program size can remain relatively small.

2) System Integration

PC VSTAT consists of twenty independent programs that provide the calculations for each of the statistical items that the package offers. Because of the memory limitation of the diskette,
and the redundancy of equations, some of the programs represent more than one item. A thorough explanation of these sections is provided in the next section, "Features - Detailed Design".

A major concern in the design of this package is the ease of operation, particularly for those who have little or no computer knowledge. A user-friendly environment has been established with the use of menus and prompts to aid the user in selecting and performing the desired statistic. All of the information the user enters is checked for its validity to avoid run-time errors. This is achieved by reading a user input as a string of characters and then comparing the elements of the string to a set of valid characters. If the entry is valid, the input can be converted to its appropriate type (i.e., real or integer) using the data conversion functions made available by Turbo Pascal.

Checks are also done on the user’s inputs to be sure that errors in calculations are avoided, such as attempting to divide by zero.

When a new data vector file is created, a check is done on the names of all existing files to be sure that the new name is unique so that an existing file is not written over. If VEC2 is the name of an
existing file, and the user enters this as the name for a new vector, the following error message is displayed:

WARNING - VECTOR VEC2 ALREADY EXISTS!

Please enter another name:

An error message is also displayed if the user attempts to load a file that does not exist.

When the user attempts to edit a data vector, the observation that he/she wants to remove, add or change is displayed with the corresponding value and the user is asked to verify that this is the correct data point to modify. Verification is also required when the user selects the PURGE option to remove a vector from the disk. Upon entering the name of the vector to be erased, the user is asked if he/she is certain they want to purge the named vector. This provides an opportunity for the user to rethink the results of their action and perhaps avoid losing needed data.

These features are the result of incorporating the human factors of the system into the package design to provide a friendly, interactive environment for the user.
PC VSTAT consists of a group of independent Pascal programs in which VSTAT, the main program, internally locates and passes control to the program containing the section the user selected from the main menu, provided by VSTAT (See figures 2.1-2.3).

The statistics programs are "chained" from VSTAT and then executed. Upon completion of the desired section the user is provided with the opportunity of running the section again. If the user is through with this particular section, a response of 'N' will cause VSTAT to be executed from the present program and the user is returned to the main menu. Each of the three pages of the main menu provide an option for the user to EXIT the package. If this option is selected, the following message is displayed and the user is returned to the disk operating system (DOS):

THANKS FOR USING "PC VSTAT"!

You may now remove the diskette.

A>

D) Features - Detailed Design

The features that PC VSTAT offers can be classified into three sections; data manipulation, descriptive statistics and hypothesis testing. As mentioned previously, files were chosen as the storage structure
for the data vectors. However, arrays are used to aid in calculations. Upon selection of a menu item, the user enters the name(s) of the desired vector(s) upon which some statistical analysis is desired. The file containing the named vector is located and opened. The contents of this file are then read to an array and the desired statistic is performed. The following programs are used to calculate and display the statistical options offered in the PC VSTAT package. Following is a discussion of the design of the programs providing these functions and operations.

1. Data Manipulation

a) VSTAT, as mentioned above, contains the main menu from which the desired statistical section is selected. This program also contains a procedure, DESCRIBE, that provides on-line documentation for the user to describe each of the menu options. The user may view several descriptions of menu options before returning to the main menu. This section is included in the package to give the user an idea of what data or information will be required for he/she to input, and what output the particular test or statistic will return. This on-line instruction allows the user to avoid consulting a user's manual during his/her session with the package.

b) CREATE is a procedure that allows the user to enter
a name for the vector (1-6 characters) to be created and to enter up to 1000 raw data points (numbers). The user signals the end of data entry by typing "END", the only permitted non-number entry. A vector may also be created by entering grouped data. The user enters the value of the data points and the number of times this value is observed in the vector. Again, the restriction on the length of the vector is 1000. The user may also use CREATE to add grouped data to an existing vector.

Entering "END" to signal the termination of data entry invokes two things. First, it creates and opens the file having the name entered by the user, and secondly, a linked-list of records is constructed with the use of pointers. The ability to deal with records and linked structures is a feature of Pascal not found in some other languages. The record is a data type that may contain data elements of various types, including another record. This linked structure is a dynamic structure, as memory is allocated on an "as-needed" basis. The advantage of using this over an array is that the size of the linked structure can be changed as the program is running, which means that the data points of a vector can be altered by inserting or deleting an element. This is done by the use of pointers and
data storage locations. The pointer not only directs the user to a desired position, but is attached to another data element in order to preserve some sort of logical relation among the data. To create a linked-list, pointers will always point to data elements of the type record. The pointer variable type name, and the record to which it points are declared at the same time (Figure 3.2).
(1) TYPE VECTORPOINTER = "VECTORRECORD; 

(2) VECTORRECORD = RECORD
    DATA:REAL;
    NEXT:VECTORPOINTER;
END;

(1) This declaration defines a pointer type with type name of VECTORPOINTER. The " preceding VECTORRECORD indicates the declaration of a pointer type. This also names the type variable that is the target for the pointers. In other words, VECTORPOINTER is a new type of variable whose elements will be pointers to VECTORRECORD.

(2) This is a typical record declaration in which VECTORRECORD is a record containing two fields, one of which is a pointer (NEXT) to other records of type VECTORRECORD, the other field is a real value, DATA, a single element of a data vector.

Figure 3.2 - Data Structure Declaration
The target of pointer NEXT must be another record of the same type, VECTORRECORD. Pascal provides a special pointer called NIL, that allows the user to terminate the chain of records (see Figures 3.3 and 3.4).

Vector DATA1:

<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6.2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3.7</td>
</tr>
<tr>
<td>4</td>
<td>-8.5</td>
</tr>
</tbody>
</table>

Figure 3.4 - Structure of vector DATA1
vector stored on the data-disk. An observation may be deleted or inserted from the beginning, end or within the data vector. The user may also change the value of an element. Editing is necessary as a way to correct mistakes in data entry. Upon entering the name of the vector to be edited, a menu is displayed showing the user the editing options available. An option is selected by the user and then a prompt asking which observation is to be deleted or changed, or which observation is to precede the new entry being inserted. The observation the user enters is displayed along with the corresponding value and upon the user's approval, the vector is edited as instructed. The contents of the file containing the original vector is written over with the contents of the newly edited vector, and the user is returned to the main menu.

d) LISTV is used to display the contents of any vector. The ability to list a vector insures that all data has been entered correctly. The vector may be listed on the printer or displayed on the terminal screen. The vector is listed one data point per line (see Figure 3.5).
VECTOR IBM005:

<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.3000</td>
</tr>
<tr>
<td>2</td>
<td>2.5000</td>
</tr>
<tr>
<td>3</td>
<td>3.7000</td>
</tr>
<tr>
<td>4</td>
<td>5.3000</td>
</tr>
<tr>
<td>5</td>
<td>6.4000</td>
</tr>
<tr>
<td>6</td>
<td>7.2000</td>
</tr>
</tbody>
</table>

Figure 3.5 - List a Data Vector

e) PURGEV is a program that allows the user to erase a file. Since the amount of physical space on a floppy disk has limited storage, it is important that the user be able to purge unwanted vector files. After the name of the vector to be erased is entered, the user is asked if they are certain they want to purge this vector. This provides the user with the opportunity to be sure he/she is purging the correct data.

f) MATHVEC is a program that allows the user to perform math with two named vectors of equal length. The first vector named performs a mathematical function (+,-,*,/) on the second vector, data point by data point. The result of this operation is written and stored in a third vector file, named by the user.
h) MATHSCA - Using only one vector of data the user should be able to perform a mathematical functions (+,−,*,/) on the data with a constant (scalar) or perform a transformation on each data point in the vector. Three transformation functions are available. The user may take the square-root of each data point in a vector. He/she may calculate the common logarithm (log10) or the natural logarithm (ln) of each data point. To perform a mathematical function the user must supply a constant value, then choose the function to be performed. The function is performed on each data point and the result is stored in a second vector file named by the user.

i) SORTV is a program that provides the user with the ability to sort a vector in ascending or descending order and store the sorted data in a new vector. The type of sort desired and the name of the vector to be sorted are entered by the user.

j) DISKDR is a program that provides the option of changing the second disk-drive assignment. If the second device is a floppy disk-drive, it is necessary that the "B:" drive be assigned. If hard disk is being used, then the desired drive assignment is "C:". This section may be used to display the current assigned disk-drive, or to change the current assignment.
2. Descriptive Statistics

The ease at which descriptive statistics can be produced for given data is an important feature of the IBM PC VSTAT package. The following section provides a discussion of the programs that provide these descriptive statistics:

a) NUMANAL is a program that performs statistical measures on a named data vector. Users of IBM PC VSTAT are able to quickly calculate the mean, standard deviation, variance, absolute mean deviation, maximum value, minimum value, range of values and the number of observations in the vector. The actual equations used in these, and subsequent calculations are listed in Appendix A.

b) CCLIN is a program that calculates either a "Correlation Coefficient" or performs a "Linear Regression" for two selected vectors, (see Figure 3.6).

![Diagram](image)

Figure 3.6 - Program CCLIN
The statistic performed is determined when the user selects an option from the main menu. The variable STAT is assigned a value to allow the appropriate parts of the program to be executed. If STAT='1' then rho, the correlation coefficient is calculated and tested to determine if it is statistically significant. The hypothesis tested is whether or not rho is equal to zero, or some other value specified by the user. The statistic used to test the hypothesis depends upon the number of data points in the vector. If the sample contains less than 30 data points, the t-statistic is used. The z-statistic is used for a sample of more than 30 data points. The equations for the test statistics are listed in Appendix A.

The user is prompted to enter the significance (alpha) level desired. The program calculates the test statistic and the actual alpha level. If the user's desired alpha level is less than or equal to the calculated alpha, the hypothesis is accepted and rho is assumed "not significant". If the user's alpha level is greater than the calculated alpha, the hypothesis is rejected and rho can be considered statistically significant.

If the user selects the Linear Regression option on the main menu, program CCLIN is executed with the
variable STAT assigned the value of 2 in order to perform a simple linear regression on two equal length vectors of data. The program calculates the estimates for the parameters of the slope and intercept. After displaying the regression equation, significance tests are performed on the parameter estimates. The user enters the significance (alpha) level desired and the program performs a test of the hypothesis that the given parameter estimate is equal to zero, for each of the two parameters, slope and intercept. The program calculates the variance of the regression line, the standard error, and then, using the appropriate equation (see Appendix A), the standard deviations for the slope and intercept are calculated. Each parameter estimate is tested by performing a two-tailed t-test.

c) NONLIN is a program that allows the user to perform either a quadratic or cubic regression analysis for two equal length vectors. The names of the vectors and the type of regression desired are entered by the user and the quadratic or cubic regression equation is calculated. The program then performs a two-tailed t-test to determine whether or not the quadratic coefficient (or cubic coefficient) can be considered equal to zero.
d) FREQHIST is another program that is used to perform two operations of PC VSTAT, (see Figure 3.7). If the user chooses the Frequency Table option from the main menu, the variable STAT is given the value one and the user inputs the name of the vector to be displayed in a frequency table.

![Diagram of FREQHIST]

Figure 3.7 - Program FREQHIST

An important consideration in the design of this program was the number of intervals to be used in the table. The program is designed to give the user the choice of entering the number of intervals or having the program calculate the number. The number of intervals is calculated by taking the square root of the total number of observations in the vector. This method of calculation has been suggested by Hines and Montgomery (1980) as a rule of thumb. The maximum number of cells allowed is thirty.
After the number of intervals have been determined, the user specifies the desired width of each interval. This width should be greater than the calculated width in order to protect from missing data points. (The calculated width multiplied by the number of intervals is equal to the range of the maximum value of the vector, minus the minimum value of the vector).

The program calculates the relative, cumulative and the cumulative-relative frequencies for each interval.

If the user is interested in displaying a histogram of a data vector, the variable STAT is assigned the value of two upon the selection of this menu option. The user enters the name of the vector of interest, and as in the Frequency Table section is given the option of entering the number of intervals or having the program calculate this, based on the square root of the total number of observations. The maximum number of intervals is again thirty.

Next, the user selects the type of histogram to be displayed (frequency, relative frequency, cumulative frequency and relative-cumulative frequency). The user specifies the width and the program displays the desired histogram on the
terminal screen. The horizontal axis displays the midpoint values for each interval.

When a relative frequency histogram is created, the histogram's vertical axis is scaled to fit within a range from 0% to 100%. When the cumulative-relative histogram is produced, its vertical axis will always be from 0 to 1.0. This feature allows various vectors of data to be compared against one another to see if they follow the same distribution.

e) SCATT is the program designed to produce a scatter plot of two vectors of data on an X-Y graph. The user supplies the names of the vectors to be used as the X and Y vectors and the program calculates a scaling factor for the axes of the graph. This scaling is based on the size of the area the graph occupies and the minimum and maximum value in the vector. The actual plot is achieved by filling the appropriate locations of a matrix with the necessary characters.

3. Hypothesis Testing

One of the most important tools for the engineer is the ability to draw conclusions about the population from which a sample is drawn. Hypothesis testing is used to assist the engineer in making decisions concerning certain populations based on
described as the process of determining if a certain statement (the hypothesis) is either true or false. This section will provide a description of the hypothesis tests provided by IBM PC VSTAT.

a) t-test

This section provides the user with the ability to test whether the population means can be considered equal for:

1) Two vectors of sample data (individual data stored in each vector).

2) Two random samples (data inputs of known number of observations, sample mean and variance for each random sample).

Either of these two tests is performed for the following cases in which the population variances for the samples are unknown, but assumed to be:

1) equal

2) unequal

The results of these tests are used by the program to calculate the significance level involved, so that the user need not perform a table lookup to make a judgment on the test.

The hypothesis that is tested is whether the population mean of one vector or one random sample is equal to the population mean of a second vector or random sample, respectively.
If the t-test is being performed on two vectors of sample data, the user must supply the names of both vectors and the significance (alpha) level desired for the test. The program calculates the number of observations, sample mean, and variance for each vector.

If a t-test is being performed on two random samples, the user is required to enter the number of observations, the sample mean and variance for each random sample.

In either case, the t-statistic and degrees of freedom are calculated by the program (equations in Appendix A). The user's selected alpha level is then compared to the actual calculated alpha level. If the user's alpha is less than or equal to the actual alpha, then the hypothesis can be accepted and the two vectors or two random samples can be assumed to have come from the same population. If the user's alpha is greater than the actual alpha, then the hypothesis is rejected.

The t-test calculates the two-tailed significance level for the t-statistic and degrees of freedom provided.

This program is also used to perform a "paired t-test", which provides the user the ability to test whether the difference of the population means for
two vectors of data can be considered to be equal to zero. Upon entering the name of the vector and desired significance (alpha) level, the program calculates the number of observations, the sample mean, and the variance for each of the two vectors. The sum of data difference, the sum of squared data difference and the standard deviation of data difference are also calculated. Finally, the t-statistic and degrees of freedom are calculated, and the user's alpha level is compared to the actual alpha level. If the user's desired alpha is less than or equal to the calculated alpha, then the hypothesis can be accepted and the difference of the population means for two vectors of data can be considered to be equal to zero. If the user's desired alpha is greater than the calculated alpha, then the hypothesis is rejected.

Figure 3.8 displays the three available t-tests.
b) F-test

The user is able to test whether or not the population variances can be considered equal for:

1) Two vectors of sample data (individual data stored in each vector).

2) Two random samples (data input of known number of observations and variance of each random sample).

If two vectors of sample data are being tested then the program calculates the mean, variance and number of observations for each vector once the vector names are entered by the user.

If two random samples are being tested, then the user must supply the number of observations and the variance of each sample.

The F-statistic and the degrees of freedom are calculated. The user's desired significance (alpha) level is compared with the calculated alpha. If the user's desired alpha is less than or equal to the calculated alpha, the hypothesis can be accepted and the population variances can be assumed to be equal for both vectors or both random samples. If the user's desired alpha is greater than the calculated alpha then the hypothesis is rejected.

Figure 3.9 displays the available F-tests.
c) Chi-Square goodness-of-fit test

The program CHI performs the Chi-Squared goodness-of-fit test on a vector of data. The user enters, or the program calculates, the estimates for the population mean and variance. The user has the choice of entering, or having the program calculate, the number of intervals to be used in the frequency distribution. The program uses the square root of the number of observations in the vector. The maximum number of intervals permitted is thirty.

The observed frequency for each interval is determined. The results from the observed frequency for each interval are used to calculate the expected values of the proposed distribution. PC VSTAT provides the option of comparing data to:

1) The Theoretical Normal Distribution
2) The Negative Exponential Distribution

Once the expected values have been calculated,
the program performs a test to insure that all intervals of the desired distribution have five or more expected observations in them. If there are intervals violating this constraint, the theoretical and actual intervals involved are combined with their adjacent intervals to achieve the required minimum.

The degrees of freedom and the Chi-Squared statistic are then computed by the program.

The user's desired significance (alpha) level is compared with the calculated alpha. If the user's desired alpha is less than or equal to the calculated alpha, the hypothesis can be accepted and the data within the vector can be assumed to be of the selected distribution. If the user's desired alpha is greater than the calculated alpha then the hypothesis is rejected.

d) Kolmogorov-Smirnov Tests

The user is able to perform the Kolmogorov-Smirnov (K-S) one sample test (goodness-of-fit) and two sample test (comparing the cumulative distribution functions (CDF's) of two independent samples).

The program KSI performs a Kolmogorov-Smirnov goodness-of-fit test for a sample to determine how well the data fits a:
1) Normal Distribution
2) Negative Exponential Distribution

The K-S test can be used on very small samples where the Chi-squared test cannot be used because of the constraint that the number of observations must be greater than five for each interval.

The user enters the name of the vector to be tested. Either the user enters, or the program calculates, the population mean and variance estimates. The user supplies the number of intervals desired and the program determines the frequency distribution of the data vector. The desired distribution is then calculated in addition to the Kolmogorov-Smirnov statistic and the maximum absolute difference. A two-sided test is performed to test the hypothesis that the distribution of the random sample is equal to the chosen distribution. The hypothesis is rejected if the greatest observed difference is greater than the calculated critical value (see equations in Appendix A).

The program KS2 performs the two sample Kolmogorov-Smirnov test, which is a way to compare the cumulative distribution functions (CDF's) of two independent samples in order to determine if the data in each vector is of the same distribution.

The user supplies the names of the two vectors of
data to be tested. Because of the algorithm and the arrays used to perform this test, the maximum number of data points permitted is 150 for each vector.

The program sorts the vectors in ascending order and the test statistic is calculated. The output includes the test value, the cumulative percentage for each vector and the percent difference between each vector. The total observations per vector are displayed along with the K-S statistic, which is the absolute maximum percent difference.

The program calculates the exact two-tailed probability that the difference between the two vectors of data is due to chance and the one-tailed probability that one distribution exceeds the other due to chance; specifically, for two independent samples coming from the same continuous population. An explanation of the method of calculating the above probabilities is provided in Taub (see Bibliography) which was also the source of the algorithm used.

e) VALUE is a program that calculates the significance (alpha) level for a given test statistic and its associated degrees of freedom, if applicable. The test statistics calculated are the z-statistic (Normal), the t-statistic (Student t), the F-statistic and the Chi-squared statistic.
statistic and the Chi-squared statistic.

The user first selects the type of test statistic and then enters the value of the test statistic and the degrees of freedom, if applicable. The distributions are shown in Figure 3.10.
Given $r$, the table gives (a) the one-tail $I_0$ value with $a$ of the area above it, that is, $P(t \geq I_0) = a$, or (b) the two-tail $I_0$ and $-I_0$ values with $a/2$ in each tail, that is, $P(t \leq -I_0) + P(t \geq I_0) = a$.

Given $r_1$ and $r_2$, the table gives the $F_0$ value with $a$ of the area above it; that is, $P(F \geq F_0) = a$.

Given $r$, the table gives the $\chi^2$ value with $a$ of the area above it; that is, $P(\chi^2 \geq \chi^2) = a$.

CHAPTER IV
VALIDATION

To insure that PC VSTAT performs the available statistics correctly, the following validation was performed. The output resulting from these tests follows.

1) 'Create a vector', 'Edit a vector', 'Entry of grouped data' and 'List a vector' were used to create the vectors for testing of validation (see Figures 4.1 and 4.2).

2) 'Math with two vectors' is tested by using vectors IBM002 and IBM003. The output is shown in Figure 4.3.
   a) ADD is the result of IBM002 + IBM003.
   b) SUBTR is the result of IBM002 - IBM003.
   c) MULT is the result of IBM002 * IBM003.
   d) DIV is the result of IBM002 / IBM003.

3) 'Math with a vector and a constant' is tested by using IBM004. The output is shown in Figures 4.4 and 4.5.
   a) ADD10 is the result of IBM004 + 10.
   b) SUB10 is the result of IBM004 - 10.
   c) MULT10 is the result of IBM004 * 10.
   d) DIV10 is the result of IBM004 / 10.
   e) SQRT is the result of (IBM004)^(1/2).
   f) LOG10 is the result of Log10(IBM004).
   g) LN is the result of Ln(IBM004).
4) 'Numerical Analysis' is performed on IBM001. The output is shown in Figure 4.6.

5) 'Correlation Coefficient for two vectors' is tested by using the data in vectors IBM002 and IBM003 (Blank, p.519-520). The output is shown in Figure 4.7.

6) 'Linear Regression for two vectors' is tested by using the data in vectors IBM004 and IBM005, from Blank p.493. The output is shown in Figure 4.8.

7) 'Frequency Table for a Vector' and 'Histogram for a vector' are tested using IBM006 (Blank, p.38). The output is shown in Figures 4.9 and 4.10, respectively.

8) 'Scatter graph' is tested using IBM002 and IBM003 (Blank p.519-520). The output is in Figure 4.11.

9) 'F-test' is tested using data from Blank p.397. The output is shown in Figure 4.12.

10) 't-test' is tested using the data from Hines and Montgomery p.288. The output is shown in Figure 4.13.

11) 'Paired t-test' is tested using the data from IBM007 and IBM008. The output is shown in Figure 4.14.

12) 'Nonlinear Regression' is tested by using data from Mendenhall p.127 and 144 for quadratic regression. The output is shown in Figure 4.15. The cubic regression is tested using data from Mendenhall p.132. The output is shown if Figure 4.16.

13) 'Chi-Square test' is tested using the data from Dixon and Massey p.280 (vector XTEST) for the normal
distribution, and data from Dey p.66 (vector EXP) for the Negative Exponential Distribution. The output is shown in Figures 4.17 and 4.18 respectively.

14) ‘Komogorov-Smirnov one-sample test’ is tested using data from Massey p.72 for the normal distribution and data from Dey p.66 for the Negative Exponential Distribution. The output is in Figures 4.19 and 4.20, respectively.

15) ‘Komogorov-Smirnov two-sample test’ is tested by using data from Taub p.140. The output is shown in Figure 4.21.

16) ‘Random number generation’ and ‘Random Sampling’ are tested by generating the random numbers as shown in Figure 4.22.

a) R1 is uniform random numbers.

b) R2 is normal random variates with mean=25, standard deviation = 5.

c) R3 is negative exponential random variates with mean=40.

d) SAMPW is random sampling with replacement and the population size is 8.

e) SAMPWO is random sampling without replacement and the population size of 10.
<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.0000</td>
<td>1</td>
<td>2.0000</td>
</tr>
<tr>
<td>2</td>
<td>8.0000</td>
<td>2</td>
<td>9.0000</td>
</tr>
<tr>
<td>3</td>
<td>13.0000</td>
<td>3</td>
<td>18.0000</td>
</tr>
<tr>
<td>4</td>
<td>18.0000</td>
<td>4</td>
<td>20.0000</td>
</tr>
<tr>
<td>5</td>
<td>19.0000</td>
<td>5</td>
<td>25.0000</td>
</tr>
<tr>
<td>6</td>
<td>21.0000</td>
<td>6</td>
<td>38.0000</td>
</tr>
<tr>
<td>7</td>
<td>25.0000</td>
<td>7</td>
<td>50.0000</td>
</tr>
<tr>
<td>8</td>
<td>28.0000</td>
<td>8</td>
<td>65.0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>69.0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>71.0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td>92.0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td>107.0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13</td>
<td>110.0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14</td>
<td>135.0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>150.0000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0600</td>
</tr>
<tr>
<td>2</td>
<td>0.1000</td>
</tr>
<tr>
<td>3</td>
<td>0.1600</td>
</tr>
<tr>
<td>4</td>
<td>0.1200</td>
</tr>
<tr>
<td>5</td>
<td>0.2400</td>
</tr>
<tr>
<td>6</td>
<td>0.2100</td>
</tr>
<tr>
<td>7</td>
<td>0.2900</td>
</tr>
<tr>
<td>8</td>
<td>0.3100</td>
</tr>
<tr>
<td>9</td>
<td>0.4300</td>
</tr>
<tr>
<td>10</td>
<td>0.4100</td>
</tr>
<tr>
<td>11</td>
<td>0.4300</td>
</tr>
<tr>
<td>12</td>
<td>0.4700</td>
</tr>
<tr>
<td>13</td>
<td>0.6100</td>
</tr>
<tr>
<td>14</td>
<td>0.6300</td>
</tr>
<tr>
<td>15</td>
<td>0.9500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0200</td>
</tr>
<tr>
<td>2</td>
<td>0.4350</td>
</tr>
<tr>
<td>3</td>
<td>0.5200</td>
</tr>
<tr>
<td>4</td>
<td>0.6000</td>
</tr>
<tr>
<td>5</td>
<td>0.9900</td>
</tr>
<tr>
<td>6</td>
<td>1.3500</td>
</tr>
<tr>
<td>7</td>
<td>1.7800</td>
</tr>
<tr>
<td>8</td>
<td>1.8300</td>
</tr>
<tr>
<td>9</td>
<td>2.1600</td>
</tr>
<tr>
<td>10</td>
<td>2.2200</td>
</tr>
<tr>
<td>11</td>
<td>2.3400</td>
</tr>
<tr>
<td>12</td>
<td>2.9400</td>
</tr>
<tr>
<td>13</td>
<td>3.4300</td>
</tr>
<tr>
<td>14</td>
<td>4.8200</td>
</tr>
<tr>
<td>15</td>
<td>5.3200</td>
</tr>
<tr>
<td>16</td>
<td>5.3200</td>
</tr>
<tr>
<td>17</td>
<td>6.4400</td>
</tr>
<tr>
<td>18</td>
<td>7.8000</td>
</tr>
</tbody>
</table>

Figure 4.1 - Sample Vectors
### Figure 4.2 - Sample vectors

<table>
<thead>
<tr>
<th>VECTOR IBM007:</th>
<th>VECTOR IBM010:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observation</td>
<td>Value</td>
</tr>
<tr>
<td>1</td>
<td>4.0000</td>
</tr>
<tr>
<td>2</td>
<td>3.0000</td>
</tr>
<tr>
<td>3</td>
<td>3.0000</td>
</tr>
<tr>
<td>4</td>
<td>4.0000</td>
</tr>
<tr>
<td>5</td>
<td>4.0000</td>
</tr>
<tr>
<td>6</td>
<td>3.0000</td>
</tr>
<tr>
<td>7</td>
<td>2.0000</td>
</tr>
<tr>
<td>8</td>
<td>2.0000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VECTOR IBM008:</th>
<th>VECTOR IBM011:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observation</td>
<td>Value</td>
</tr>
<tr>
<td>1</td>
<td>3.0000</td>
</tr>
<tr>
<td>2</td>
<td>3.0000</td>
</tr>
<tr>
<td>3</td>
<td>5.0000</td>
</tr>
<tr>
<td>4</td>
<td>3.0000</td>
</tr>
<tr>
<td>5</td>
<td>4.0000</td>
</tr>
<tr>
<td>6</td>
<td>2.0000</td>
</tr>
<tr>
<td>7</td>
<td>4.0000</td>
</tr>
<tr>
<td>8</td>
<td>2.0000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VECTOR IBM009:</th>
<th>VECTOR IBM012:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observation</td>
<td>Value</td>
</tr>
<tr>
<td>1</td>
<td>0.0000</td>
</tr>
<tr>
<td>2</td>
<td>0.0000</td>
</tr>
<tr>
<td>3</td>
<td>1.0000</td>
</tr>
<tr>
<td>4</td>
<td>1.0000</td>
</tr>
<tr>
<td>5</td>
<td>3.0000</td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VECTOR K21:</th>
<th>VECTOR K22:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observation</td>
<td>Value</td>
</tr>
<tr>
<td>1</td>
<td>3.0000</td>
</tr>
<tr>
<td>2</td>
<td>6.0000</td>
</tr>
<tr>
<td>3</td>
<td>7.0000</td>
</tr>
<tr>
<td>4</td>
<td>8.0000</td>
</tr>
<tr>
<td>5</td>
<td>9.0000</td>
</tr>
</tbody>
</table>
### VECTOR ADD:

<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0600</td>
</tr>
<tr>
<td>2</td>
<td>9.1000</td>
</tr>
<tr>
<td>3</td>
<td>18.1600</td>
</tr>
<tr>
<td>4</td>
<td>20.1200</td>
</tr>
<tr>
<td>5</td>
<td>25.2400</td>
</tr>
<tr>
<td>6</td>
<td>38.2100</td>
</tr>
<tr>
<td>7</td>
<td>50.2900</td>
</tr>
<tr>
<td>8</td>
<td>65.3100</td>
</tr>
<tr>
<td>9</td>
<td>69.4300</td>
</tr>
<tr>
<td>10</td>
<td>71.4100</td>
</tr>
<tr>
<td>11</td>
<td>92.4300</td>
</tr>
<tr>
<td>12</td>
<td>107.4700</td>
</tr>
<tr>
<td>13</td>
<td>110.6100</td>
</tr>
<tr>
<td>14</td>
<td>135.6300</td>
</tr>
<tr>
<td>15</td>
<td>150.9500</td>
</tr>
</tbody>
</table>

### VECTOR SUBTR:

<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.9400</td>
</tr>
<tr>
<td>2</td>
<td>8.9000</td>
</tr>
<tr>
<td>3</td>
<td>17.8400</td>
</tr>
<tr>
<td>4</td>
<td>19.8800</td>
</tr>
<tr>
<td>5</td>
<td>24.7600</td>
</tr>
<tr>
<td>6</td>
<td>37.7900</td>
</tr>
<tr>
<td>7</td>
<td>49.7100</td>
</tr>
<tr>
<td>8</td>
<td>64.6900</td>
</tr>
<tr>
<td>9</td>
<td>68.5700</td>
</tr>
<tr>
<td>10</td>
<td>70.5900</td>
</tr>
<tr>
<td>11</td>
<td>91.5700</td>
</tr>
<tr>
<td>12</td>
<td>106.5300</td>
</tr>
<tr>
<td>13</td>
<td>109.3900</td>
</tr>
<tr>
<td>14</td>
<td>134.3700</td>
</tr>
<tr>
<td>15</td>
<td>149.0500</td>
</tr>
</tbody>
</table>

### VECTOR MULT:

<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1200</td>
</tr>
<tr>
<td>2</td>
<td>0.9000</td>
</tr>
<tr>
<td>3</td>
<td>2.8800</td>
</tr>
<tr>
<td>4</td>
<td>2.4000</td>
</tr>
<tr>
<td>5</td>
<td>6.0000</td>
</tr>
<tr>
<td>6</td>
<td>7.9800</td>
</tr>
<tr>
<td>7</td>
<td>14.5000</td>
</tr>
<tr>
<td>8</td>
<td>20.1500</td>
</tr>
<tr>
<td>9</td>
<td>29.6700</td>
</tr>
<tr>
<td>10</td>
<td>29.1100</td>
</tr>
<tr>
<td>11</td>
<td>39.5600</td>
</tr>
<tr>
<td>12</td>
<td>50.2900</td>
</tr>
<tr>
<td>13</td>
<td>67.1000</td>
</tr>
<tr>
<td>14</td>
<td>85.0500</td>
</tr>
<tr>
<td>15</td>
<td>142.5000</td>
</tr>
</tbody>
</table>

### VECTOR DIV:

<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>33.3333</td>
</tr>
<tr>
<td>2</td>
<td>90.0000</td>
</tr>
<tr>
<td>3</td>
<td>112.5000</td>
</tr>
<tr>
<td>4</td>
<td>166.6667</td>
</tr>
<tr>
<td>5</td>
<td>104.1667</td>
</tr>
<tr>
<td>6</td>
<td>180.9524</td>
</tr>
<tr>
<td>7</td>
<td>172.4138</td>
</tr>
<tr>
<td>8</td>
<td>209.6774</td>
</tr>
<tr>
<td>9</td>
<td>160.4611</td>
</tr>
<tr>
<td>10</td>
<td>173.1707</td>
</tr>
<tr>
<td>11</td>
<td>213.9535</td>
</tr>
<tr>
<td>12</td>
<td>227.6596</td>
</tr>
<tr>
<td>13</td>
<td>180.3279</td>
</tr>
<tr>
<td>14</td>
<td>214.2857</td>
</tr>
<tr>
<td>15</td>
<td>157.8947</td>
</tr>
</tbody>
</table>

Figure 4.3 - Math with two Vectors
## Figure 4.4 - Math with a Vector and a Constant

<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11.0000</td>
</tr>
<tr>
<td>2</td>
<td>12.0000</td>
</tr>
<tr>
<td>3</td>
<td>13.0000</td>
</tr>
<tr>
<td>4</td>
<td>14.0000</td>
</tr>
<tr>
<td>5</td>
<td>15.0000</td>
</tr>
<tr>
<td>6</td>
<td>16.0000</td>
</tr>
</tbody>
</table>

### Vector Addition

<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.0000</td>
</tr>
<tr>
<td>2</td>
<td>20.0000</td>
</tr>
<tr>
<td>3</td>
<td>30.0000</td>
</tr>
<tr>
<td>4</td>
<td>40.0000</td>
</tr>
<tr>
<td>5</td>
<td>50.0000</td>
</tr>
<tr>
<td>6</td>
<td>60.0000</td>
</tr>
</tbody>
</table>

### Vector Multiplication

<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-9.0000</td>
</tr>
<tr>
<td>2</td>
<td>-8.0000</td>
</tr>
<tr>
<td>3</td>
<td>-7.0000</td>
</tr>
<tr>
<td>4</td>
<td>-6.0000</td>
</tr>
<tr>
<td>5</td>
<td>-5.0000</td>
</tr>
<tr>
<td>6</td>
<td>-4.0000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1000</td>
</tr>
<tr>
<td>2</td>
<td>0.2000</td>
</tr>
<tr>
<td>3</td>
<td>0.3000</td>
</tr>
<tr>
<td>4</td>
<td>0.4000</td>
</tr>
<tr>
<td>5</td>
<td>0.5000</td>
</tr>
<tr>
<td>6</td>
<td>0.6000</td>
</tr>
</tbody>
</table>
### VECTOR SQRT:

<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0000</td>
</tr>
<tr>
<td>2</td>
<td>1.4142</td>
</tr>
<tr>
<td>3</td>
<td>1.7321</td>
</tr>
<tr>
<td>4</td>
<td>2.0000</td>
</tr>
<tr>
<td>5</td>
<td>2.2361</td>
</tr>
<tr>
<td>6</td>
<td>2.4495</td>
</tr>
</tbody>
</table>

### VECTOR LOG10:

<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0000</td>
</tr>
<tr>
<td>2</td>
<td>0.3010</td>
</tr>
<tr>
<td>3</td>
<td>0.4771</td>
</tr>
<tr>
<td>4</td>
<td>0.6021</td>
</tr>
<tr>
<td>5</td>
<td>0.6990</td>
</tr>
<tr>
<td>6</td>
<td>0.7782</td>
</tr>
</tbody>
</table>

### VECTOR LN:

<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0000</td>
</tr>
<tr>
<td>2</td>
<td>0.6931</td>
</tr>
<tr>
<td>3</td>
<td>1.0986</td>
</tr>
<tr>
<td>4</td>
<td>1.3863</td>
</tr>
<tr>
<td>5</td>
<td>1.6094</td>
</tr>
<tr>
<td>6</td>
<td>1.7918</td>
</tr>
</tbody>
</table>

Figure 4.5 - Vector Math
STATISTICAL MEASURES

VECTOR C:IBM001

<table>
<thead>
<tr>
<th>Measure</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Observations</td>
<td>8</td>
</tr>
<tr>
<td>Minimum Value</td>
<td>4.0000</td>
</tr>
<tr>
<td>Maximum Value</td>
<td>28.0000</td>
</tr>
<tr>
<td>Range</td>
<td>24.0000</td>
</tr>
<tr>
<td>Mean</td>
<td>17.0000</td>
</tr>
<tr>
<td>Variance</td>
<td>67.4286</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>8.2115</td>
</tr>
<tr>
<td>Mean Deviation</td>
<td>6.5000</td>
</tr>
</tbody>
</table>

Figure 4.6 - Statistical Measures
CALCULATION OF CORRELATION COEFFICIENT

VECTOR C:IBM002

Number of Observations = 15
Average = 64.0667
Variance = 2183.9238
Standard Deviation = 46.7325

VECTOR C:IBM003

Number of Observations = 15
Average = 0.3613
Variance = 0.0577
Standard Deviation = 0.2401

CORRELATION COEFFICIENT = 0.9609

STATISTICAL HYPOTHESIS TEST

Ho: rho EQUAL 0 (independence)
H1: rho NOT EQUAL 0

Selected Alpha = 0.05
Calculated t-Value = 12.5099
Degrees of Freedom = 13
Probability of both tails = 0.0000

REJECT Ho, thus rho is significantly different than 0.0000

Figure 4.7 - Correlation Coefficient
CALCULATION OF LINEAR REGRESSION

Vector IBM004  (Containing the independent variable)
Vector IBM005  (Containing the dependent variable)

REgression equation:
\[ Y = 0.1200 + 1.2229X \]

Standard deviation of "a" (intercept) = 0.2034
Standard deviation of "b" (slope) = 0.0522

STATISTICAL HYPOTHESIS TEST for Intercept

Ho: "a" (intercept) = 0
Hi: "a" not equal to 0

Selected Alpha = 0.05
Calculated t-value = 0.5901
Degrees of freedom = 4
Probability of both tails = 0.5868

Accept Ho; thus "a" is NOT significant

STATISTICAL HYPOTHESIS TEST for slope

Ho: "b" (slope) = 0 (Two vectors are independent)
Hi: "b" not equal to 0

Selected Alpha = 0.05
Calculated t-value = 23.4191
Degrees of freedom = 4
Probability of two tails = 0.0000

Reject Ho; thus "b" is significant

Figure 4.8 - Linear Regression
### FREQUENCY TABLE FOR VECTOR IBM006

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30.0000 - 40.0000</td>
<td>8</td>
<td>8</td>
<td>0.0800</td>
<td>0.0800</td>
</tr>
<tr>
<td>2</td>
<td>40.0000 - 50.0000</td>
<td>17</td>
<td>25</td>
<td>0.1700</td>
<td>0.2500</td>
</tr>
<tr>
<td>3</td>
<td>50.0000 - 60.0000</td>
<td>42</td>
<td>67</td>
<td>0.4200</td>
<td>0.6700</td>
</tr>
<tr>
<td>4</td>
<td>60.0000 - 70.0000</td>
<td>21</td>
<td>88</td>
<td>0.2100</td>
<td>0.8800</td>
</tr>
<tr>
<td>5</td>
<td>70.0000 - 80.0000</td>
<td>11</td>
<td>99</td>
<td>0.1100</td>
<td>0.9900</td>
</tr>
<tr>
<td>6</td>
<td>80.0000 - 90.0000</td>
<td>1</td>
<td>100</td>
<td>0.0100</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

Number of Observations = 100
Mean = 56.3100
Std. Dev. = 12.1295
Minimum Value = 33.0000
Maximum Value = 89.0000

Figure 4.9 - Frequency Table
Figure 4.11 - Scatter Graph
F-TEST

Sample 1
Number of Observations = 15
Variance = 0.2000
Standard Deviation = 0.4472

Sample 2
Number of Observations = 20
Variance = 0.1500
Standard Deviation = 0.3873

STATISTICAL HYPOTHESIS TEST

Ho: Variance of first population = Variance of second population
H1: Variance of first population is NOT EQUAL TO Variance of second population

Selected ALPHA = 0.1000
Calculated F-Value (Fo) = 1.3333
D.F. in numerator (Sample 1) = 14
D.F. in denominator (Sample 2) = 19
Probability of both tails = 0.2751

Thus: ACCEPT Ho

Figure 4.12 - F-Test
SAMPLE 1

<table>
<thead>
<tr>
<th>No. of Observations</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>91.73</td>
</tr>
<tr>
<td>Variance</td>
<td>3.89</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>1.97</td>
</tr>
</tbody>
</table>

SAMPLE 2

<table>
<thead>
<tr>
<th>No. of Observations</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>93.75</td>
</tr>
<tr>
<td>Variance</td>
<td>4.02</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>2.01</td>
</tr>
</tbody>
</table>

HYPOTHESIS TEST FOR EQUAL VARIANCES

Ho: \( \mu_1 = \mu_2 \)
H1: \( \mu_1 \) is not equal to \( \mu_2 \)

Selected Alpha = 0.0500
Calculated t-value = -2.0315
Degrees of Freedom = 14
Probability of both tails = 0.0616

THUS: ACCEPT Ho

HYPOTHESIS TEST FOR UNEQUAL VARIANCES:

Ho: \( \mu_1 = \mu_2 \)
H1: \( \mu_1 \) is not equal to \( \mu_2 \)

Selected Alpha = 0.0500
Calculated t-value = -2.0315
Degrees of Freedom = 16
Probability of both tails = 0.0592

THUS: ACCEPT Ho
PAIRED t-TEST

VECTOR IBM007

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Observations</td>
<td>8</td>
</tr>
<tr>
<td>Average</td>
<td>3.1250</td>
</tr>
<tr>
<td>Variance</td>
<td>0.6964</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>0.8345</td>
</tr>
</tbody>
</table>

VECTOR IBM008

<table>
<thead>
<tr>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Observations</td>
<td>8</td>
</tr>
<tr>
<td>Average</td>
<td>3.2500</td>
</tr>
<tr>
<td>Variance</td>
<td>1.0714</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>1.0351</td>
</tr>
</tbody>
</table>

Sum of difference            = -1.0000
Sum of difference squared    = 11.0000
Average difference           = -0.1250
Std. Dev. of difference      = 1.2464

HYPOTHESIS TEST FOR PAIRED t-TEST

Ho: \( \mu \) of Vector IBM007 = \( \mu \) of Vector IBM008
Hi: \( \mu \) of Vector IBM007 is not equal to \( \mu \) of Vector IBM008

Selected Alpha               = 0.0500
Calculated t-value           = -0.2837
Degrees of Freedom           = 7
Probability of both tails    = 0.7849

THUS: ACCEPT Ho

Figure 4.14 - Paired t-test
Calculation of Quadratic regression

VECTOR IBM009 (Containing the independent variable)
VECTOR IBM010 (Containing the dependent variable)

Regression Equation:
\[ Y = -1.5000 + 2.4167X - 0.4167X^2 \]

Sum of Square Error = 1.0000

STATISTICAL HYPOTHESIS TEST

Ho: Quadratic constant (c) = 0
H1: Quadratic constant (c) is not equal to 0

Selected alpha = 0.0500
Calculated t-value = 1.2910
Degrees of freedom = 2
Probability of both tails = 0.3258

Accept Ho: The Quadratic constant is NOT significantly different than 0

Figure 4.15 - Quadratic Equation
Calculation of Cubic regression

VECTOR IBM011 (Containing the independent variable)
VECTOR IBM012 (Containing the dependent variable)

Regression Equation:

\[ Y = + 0.9524 + 1.0833X + 0.1190X^2 - 0.0833X^3 \]

Sum of Square Error = 0.0238

STATISTICAL HYPOTHESIS TEST

Ho: Cubic constant (d) = 0
H1: Cubic constant (d) is not equal to 0

Selected alpha = 0.0500
Calculated t-value = 13.7477
Degrees of freedom = 3
Probability of both tails = 0.0008

Reject Ho: The Cubic constant is significantly different than 0

Figure 4.16 - Cubic Regression
### Chi-Square goodness-of-fit Test on VECTOR C:XTEST

<table>
<thead>
<tr>
<th>CELL</th>
<th>CELL BOUNDRIES</th>
<th>OBSERVED</th>
<th>EXPECTED ((\text{Ob.-Ex.})^2/\text{Ex.})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-999999.0000  - 90.0000</td>
<td>8</td>
<td>8.0717 0.0006</td>
</tr>
<tr>
<td>2</td>
<td>90.0000 - 100.0000</td>
<td>15</td>
<td>13.1344 0.2650</td>
</tr>
<tr>
<td>3</td>
<td>100.0000 - 110.0000</td>
<td>21</td>
<td>20.9337 0.0002</td>
</tr>
<tr>
<td>4</td>
<td>110.0000 - 120.0000</td>
<td>23</td>
<td>23.5001 0.0106</td>
</tr>
<tr>
<td>5</td>
<td>120.0000 - 130.0000</td>
<td>16</td>
<td>18.5822 0.3588</td>
</tr>
<tr>
<td>6</td>
<td>130.0000 - 140.0000</td>
<td>9</td>
<td>10.3489 0.1758</td>
</tr>
<tr>
<td>7</td>
<td>140.0000 - 999999.0000</td>
<td>8</td>
<td>5.4291 1.2174</td>
</tr>
</tbody>
</table>

**STATISTICAL HYPOTHESIS TEST**

- **Ho**: Data in vector C:XTEST is Normally Distributed.
- **H1**: Data in vector C:XTEST is NOT Normally Distributed.

- Number of Observations \(= 100\)
- Selected Alpha \(= 0.0500\)
- Calculated Chi-Square value \(= 2.0285\)
- Degrees of Freedom \(= 4\)
- Prob.( chi-sq. \(\geq\) 2.0285 , 4 ) \(= 0.7357\)

Thus; ACCEPT Ho, Vector C:XTEST is Normally Distributed.

**Figure 4.17** — Chi-Squared goodness-of-fit test (Normal Distribution)
Chi-Square goodness-of-fit Test on VECTOR C:EXP

<table>
<thead>
<tr>
<th>CELL</th>
<th>CELL BOUNDARIES</th>
<th>OBSERVED</th>
<th>EXPECTED</th>
<th>(O-E)^2/E</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-999999.0000 - 2.0000</td>
<td>8</td>
<td>9.1970</td>
<td>0.1558</td>
</tr>
<tr>
<td>2</td>
<td>2.0000 - 999999.0000</td>
<td>10</td>
<td>7.7733</td>
<td>0.6378</td>
</tr>
</tbody>
</table>

STATISTICAL HYPOTHESIS TEST

H₀: Data in vector C:EXP is Neg. Exponentially Distributed.
H₁: Data in vector C:EXP is NOT Neg. Exponentially Distributed.

Number of Observations = 18
Selected Alpha = 0.0500
Calculated Chi-Square value = 0.7936
Degrees of Freedom = 1
Prob.(chi-sq. > 0.7936 / 1) = 0.7460

Thus; ACCEPT H₀, Vector C:EXP is Neg. Exponentially Distributed.

Figure 4.18 - Chi-Square goodness-of-fit test
(Negative Exponential Distribution)
STATISTICAL HYPOTHESIS TEST

Ho: Data in vector C:IBM016 is Normally Distributed.
H1: Data in vector C:IBM016 is NOT Normally Distributed.

Number of Observations = 511
Selected Alpha = 0.0500
Maximum Absolute Difference = 0.0166
Degrees of Freedom = 511
K-S Prob. two-tailed value = 0.0601

Thus; ACCEPT Ho, Vector C:IBM016 is Normally Distributed

Figure 4.19 - Kolmogorov-Smirnov One-Sample test
(Normal Distribution)
KOLMOGORV-SMIRNOV ONE SAMPLE TEST ON VECTOR C:EXP

<table>
<thead>
<tr>
<th>CELL Range</th>
<th>FREQ. OBS.</th>
<th>CUM. OBS.</th>
<th>CUM. EXP.</th>
<th>KOL-SMI STAT.</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td>To</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 -999999.0000 -</td>
<td>0.0200</td>
<td>1</td>
<td>0.0526</td>
<td>0.0071</td>
</tr>
<tr>
<td>2 0.0200 -</td>
<td>0.5022</td>
<td>1</td>
<td>0.1053</td>
<td>0.1644</td>
</tr>
<tr>
<td>3 0.5022 -</td>
<td>0.9844</td>
<td>2</td>
<td>0.1579</td>
<td>0.2968</td>
</tr>
<tr>
<td>4 0.9844 -</td>
<td>1.4667</td>
<td>2</td>
<td>0.2105</td>
<td>0.4082</td>
</tr>
<tr>
<td>5 1.4667 -</td>
<td>1.9489</td>
<td>2</td>
<td>0.2632</td>
<td>0.5019</td>
</tr>
<tr>
<td>6 1.9489 -</td>
<td>2.4311</td>
<td>3</td>
<td>0.3158</td>
<td>0.5808</td>
</tr>
<tr>
<td>7 2.4311 -</td>
<td>2.9133</td>
<td>0</td>
<td>0.3684</td>
<td>0.6472</td>
</tr>
<tr>
<td>8 2.9133 -</td>
<td>3.3956</td>
<td>1</td>
<td>0.4211</td>
<td>0.7031</td>
</tr>
<tr>
<td>9 3.3956 -</td>
<td>3.8776</td>
<td>1</td>
<td>0.4737</td>
<td>0.7501</td>
</tr>
<tr>
<td>10 3.8776 -</td>
<td>4.3600</td>
<td>0</td>
<td>0.5263</td>
<td>0.7897</td>
</tr>
<tr>
<td>11 4.3600 -</td>
<td>4.8422</td>
<td>1</td>
<td>0.5789</td>
<td>0.8230</td>
</tr>
<tr>
<td>12 4.8422 -</td>
<td>5.3244</td>
<td>2</td>
<td>0.6316</td>
<td>0.8511</td>
</tr>
<tr>
<td>13 5.3244 -</td>
<td>5.8067</td>
<td>0</td>
<td>0.6842</td>
<td>0.8747</td>
</tr>
<tr>
<td>14 5.8067 -</td>
<td>6.2889</td>
<td>0</td>
<td>0.7368</td>
<td>0.8945</td>
</tr>
<tr>
<td>15 6.2889 -</td>
<td>6.7711</td>
<td>1</td>
<td>0.7895</td>
<td>0.9112</td>
</tr>
<tr>
<td>16 6.7711 -</td>
<td>7.2533</td>
<td>0</td>
<td>0.8421</td>
<td>0.9253</td>
</tr>
<tr>
<td>17 7.2533 -</td>
<td>7.7356</td>
<td>0</td>
<td>0.8947</td>
<td>0.9371</td>
</tr>
<tr>
<td>18 7.7356 - 999999.0000</td>
<td>1</td>
<td>0.9474</td>
<td>0.9471</td>
<td>0.0003</td>
</tr>
</tbody>
</table>

STATISTICAL HYPOTHESIS TEST

Ho: Data in vector C:EXP is Neg. Exponentially Distributed.
Hi: Data in vector C:EXP is NOT Neg. Exponentially Distributed.

Number of Observations = 18
Selected Alpha = 0.0500
Maximum Absolute Difference = 0.2821
Degrees of Freedom = 18
K-S Prob. two-tailed value = 0.3201

Thus; ACCEPT Ho, Vector C:EXP is Neg. Exponentially Distributed

Figure 4.20 - Kolmogorov-Smirnov One-sample test (Negative Exponential Distribution)
### Kolmogorov-Smirnov Two-Sample Test

<table>
<thead>
<tr>
<th>TEST VALUES</th>
<th>FREQUENCY</th>
<th>% CUMULATIVE</th>
<th>% DIFF.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C:K21</td>
<td>C:K22</td>
<td></td>
</tr>
<tr>
<td>1.0000</td>
<td>0</td>
<td>1</td>
<td>0.0000</td>
</tr>
<tr>
<td>2.0000</td>
<td>0</td>
<td>1</td>
<td>0.0000</td>
</tr>
<tr>
<td>3.0000</td>
<td>1</td>
<td>0</td>
<td>20.0000</td>
</tr>
<tr>
<td>4.0000</td>
<td>0</td>
<td>1</td>
<td>20.0000</td>
</tr>
<tr>
<td>5.0000</td>
<td>0</td>
<td>1</td>
<td>20.0000</td>
</tr>
<tr>
<td>6.0000</td>
<td>1</td>
<td>0</td>
<td>40.0000</td>
</tr>
<tr>
<td>7.0000</td>
<td>1</td>
<td>0</td>
<td>60.0000</td>
</tr>
<tr>
<td>8.0000</td>
<td>1</td>
<td>0</td>
<td>80.0000</td>
</tr>
<tr>
<td>9.0000</td>
<td>1</td>
<td>0</td>
<td>100.0000</td>
</tr>
<tr>
<td>Total</td>
<td>5</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Maximum Cumulative Difference = 80.0000 %

The exact 1-tailed probability is 0.0397 that any apparent exceeding of C:K22 values by C:K21 values is due to chance. (Max. ordered cum. dist. function difference = -80.0000 %)

The exact 1-tailed probability is 1.0000 that any apparent exceeding of C:K21 values by C:K22 values is due to chance. (Max. cum. dist. function difference = 0.0000 %)

The exact 2-tailed probability is 0.0794 that a cum. dist. function difference of 80.0000 % or more is random for samples of sizes 5 and 4.

---

Figure 4.21 - Kolmogorov-Smirnov Two-Sample Test
### Figure 4.22 - Random Number Generation and Random Sampling

**VECTOR C:R1:**

<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6.0000</td>
</tr>
<tr>
<td>2</td>
<td>6.0000</td>
</tr>
<tr>
<td>3</td>
<td>6.0000</td>
</tr>
<tr>
<td>4</td>
<td>5.0000</td>
</tr>
<tr>
<td>5</td>
<td>5.0000</td>
</tr>
<tr>
<td>6</td>
<td>6.0000</td>
</tr>
<tr>
<td>7</td>
<td>5.0000</td>
</tr>
<tr>
<td>8</td>
<td>6.0000</td>
</tr>
<tr>
<td>9</td>
<td>5.0000</td>
</tr>
<tr>
<td>10</td>
<td>6.0000</td>
</tr>
</tbody>
</table>

**VECTOR C:SAMPW:**

<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.0000</td>
</tr>
<tr>
<td>2</td>
<td>6.0000</td>
</tr>
<tr>
<td>3</td>
<td>5.0000</td>
</tr>
<tr>
<td>4</td>
<td>4.0000</td>
</tr>
<tr>
<td>5</td>
<td>3.0000</td>
</tr>
<tr>
<td>6</td>
<td>2.0000</td>
</tr>
<tr>
<td>7</td>
<td>2.0000</td>
</tr>
<tr>
<td>8</td>
<td>7.0000</td>
</tr>
<tr>
<td>9</td>
<td>3.0000</td>
</tr>
<tr>
<td>10</td>
<td>8.0000</td>
</tr>
</tbody>
</table>

**VECTOR C:SAMPMW:**

<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24.0000</td>
</tr>
<tr>
<td>2</td>
<td>19.0000</td>
</tr>
<tr>
<td>3</td>
<td>22.0000</td>
</tr>
<tr>
<td>4</td>
<td>21.0000</td>
</tr>
<tr>
<td>5</td>
<td>18.0000</td>
</tr>
<tr>
<td>6</td>
<td>31.0000</td>
</tr>
<tr>
<td>7</td>
<td>15.0000</td>
</tr>
<tr>
<td>8</td>
<td>25.0000</td>
</tr>
<tr>
<td>9</td>
<td>18.0000</td>
</tr>
<tr>
<td>10</td>
<td>17.0000</td>
</tr>
</tbody>
</table>

**VECTOR C:R2:**

<table>
<thead>
<tr>
<th>Observation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18.0000</td>
</tr>
<tr>
<td>2</td>
<td>12.0000</td>
</tr>
<tr>
<td>3</td>
<td>23.0000</td>
</tr>
<tr>
<td>4</td>
<td>40.0000</td>
</tr>
<tr>
<td>5</td>
<td>46.0000</td>
</tr>
<tr>
<td>6</td>
<td>80.0000</td>
</tr>
<tr>
<td>7</td>
<td>61.0000</td>
</tr>
<tr>
<td>8</td>
<td>7.0000</td>
</tr>
<tr>
<td>9</td>
<td>46.0000</td>
</tr>
<tr>
<td>10</td>
<td>2.0000</td>
</tr>
</tbody>
</table>
CHAPTER V
SYSTEM EVALUATION

A) Testing

Extensive testing has been performed during the design of the programs to ensure that each program of PC VSTAT worked individually. Various examples were tested to check that each program could catch errors and provide appropriate error messages.

Upon completion of the final integration of the package, tests were performed to ensure that the package operated properly as a single entity. It was necessary to check that control of information could be passed between the various programs of the package.

Also, the programs used to validate APPLE VSTAT and MAC VSTAT were run on PC VSTAT and the results compared to ensure correctness, accuracy and agreement between packages.

The "ease of use" of PC VSTAT was also tested. Five students were given the PC VSTAT diskette and were told to create vectors and perform several of the statistical options provided by the package. All of the students were able to perform this task without consulting the user's manual (Appendix B).

B) Conclusion

One of the most important features of PC VSTAT is the
speed at which the statistics are performed. For example, it took only approximately 2 seconds for a Numerical Analysis (calculation of mean, standard deviation, etc.), to be performed on a vector of 30 data points! A sort of 1000 data points was performed in about 6 1/2 seconds. Many of the statistics seem to be performed instantly when small vectors (less than 50 data points) are used. The quick turnaround time of the calculations of PC VSTAT provide a real advantage over other statistical packages.

The final product of this system design is a general purpose, user friendly statistical package that eliminates hand calculations and dependency on distribution tables. The package provides an easy means of performing descriptive statistics and hypothesis testing. PC VSTAT has met all of the defined design requirements.

C) Future Recommendations

As PC VSTAT was being developed, ideas for future recommendations were generated.

1. Investigate the Kolmogorov-Smirnov two-sample test to see if the program could be redesigned so that it would handle more than the present 150 data points per vector.

2. An extended user's manual could be developed that would provide more than its present time descriptions and in-depth error definitions.

3. More programs could be added to the package to allow
for a wider range of tests.

4. A section could be added that would use the Z, t, Chi-squared and F tests in reverse, (i.e., given the probability, calculate the value for the right hand and/or left hand side of the test statistic.

5. Develop a version of PC VSTAT that uses the color and graphics capabilities that are now very popular additions to the IBM PC and fully compatibles.
References


Bibliography


Puetz, Gilbert H., "The Design of A Probabilistic


MANUALS


APPENDIX A

EQUATIONS USED
Equations Used

Mean

\[ x = \frac{\sum x}{n} \]

Variance

\[ s^2 = \frac{\sum x^2 - (\sum x)^2}{n-1} \]

Standard deviation

\[ s = \sqrt{s^2} \]

Mean deviation

\[ D = \frac{\sum |x - \bar{x}|}{n} \]

Correlation coefficient

\[ r = \frac{\sum xy - n\bar{x}\bar{y}}{\sqrt{\sum (x - \bar{x})^2 \sum (y - \bar{y})^2}} \]

where

\[ s_x = \sqrt{\frac{\sum (x - \bar{x})^2}{n-1}} \]

\[ s_y = \sqrt{\frac{\sum (y - \bar{y})^2}{n-1}} \]

The hypothesis tested is whether or not \( \rho \) is equal to zero or some other value specified by the user.
- A t statistic is used if the sample size is less than or equal to 30.
- A z statistic is used if the sample size is greater than 30.

The calculation for the test statistic in either case is:

\[ t = \frac{r \sqrt{n - 2}}{\sqrt{1 - r^2}} \]
The degrees of freedom are \( v = n - 2 \)

**Simple linear regression**

For the estimate of the slope:
\[
b = \frac{\sum xy - \sum x \sum y}{\sum x^2 - (\sum x)^2}
\]

For the estimate of the \( y \) intercept:
\[
a = \bar{y} - bx
\]

For the residual variance:
\[
S^2 = \frac{(\sum y^2 - a \sum y - b \sum xy)/(n - 2)}{n - 2}
\]

To test the significance of the estimates, a \( t \)-test is performed using the following test statistics:
\[
t_b = \frac{|b|}{S} \sqrt{\frac{(x - \bar{x})^2}{\sum (x^2 - \bar{x}^2)}}
\]
\[
t_a = \frac{|a|}{S} \sqrt{\frac{(1 + \frac{\sum x}{n})}{\sum (x - \bar{x})^2}}^{1/2}
\]

with degrees of freedom = \( n - 2 \) for both
t-test

Case 1: \( \sigma_1^2 = \sigma_2^2 = \sigma^2 \)

To calculate the "pooled" estimate of the variance:

\[
P = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2}
\]

The degrees of freedom = \( n_1 + n_2 - 2 \)

To calculate the test statistic:

\[
t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}} \cdot \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}
\]

Case 2: \( \sigma_1^2 \neq \sigma_2^2 \)

To calculate the test statistic:

\[
t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}} \cdot \sqrt{\frac{1}{n_1} + \frac{1}{n_2}}
\]
To calculate the degrees of freedom:

\[
\begin{pmatrix}
S^2 & S^2 \\
1 & 2 \\
--- & --- \\
n_1 & n
\end{pmatrix}^2
\]

\[
\left( \frac{S^2}{n_1} + \frac{S^2}{n_2} \right) + \frac{S^2}{n_1} + \frac{S^2}{n_2} - 2
\]

\[
\frac{\left( \frac{S^2}{n_1} \right)}{1} + \frac{\left( \frac{S^2}{n_2} \right)}{1}
\]

\[
\frac{n_1 + 1}{1} + \frac{n_2 + 1}{2}
\]
Paired t-test

To calculate the statistic:

\[
t_0 = \frac{\bar{D}}{S / \sqrt{n}}
\]

where

\[
\bar{D} = \frac{\sum D}{n}
\]

and

\[
S^2 = \frac{\sum D^2 - (\sum D)^2 / n}{n - 1}
\]

The degrees of freedom = n - 1
The difference between \(x_1\) and \(x_2\) = D

1 \hspace{1cm} 2
F-test

To calculate the statistic:

\[
F = \frac{\frac{S_1}{1}}{\frac{S_2}{2}}
\]

The degrees of freedom = \(n_1 - 1\) and \(n_2 - 1\)

Chi-squared test
(goodness of fit)

To calculate the statistic:

\[
\chi^2 = \sum_{i=1}^{n} \frac{(\text{observed}_i - \text{expected}_i)^2}{\text{expected}_i}
\]

To calculate the degrees of freedom:

\[= n - p - 1\]

where, \(n\) is the number of observations, and \(p\) is the number of parameters that were estimate.
Kolmogorov-Smirnov one-sample test

\[ D = \max \left| F(r) - S(n) \right| \]

where \( n \) is the number of observations
\( F(r) \) is the theoretical cumulative distribution
\( S(n) \) is the observed cumulative distribution
\( D \) is the maximum deviation.

Kolmogorov-Smirnov two-sample test

for a one-tailed test:

\[ D = \max \left| S(x) - S(z) \right| \]

for a two-tailed test:

\[ D = \max \left| S(x) - S(z) \right| \]

where:
\( S(x) \) is the observed cumulative step function of one of the samples.
\( S(z) \) is the observed cumulative step function of the other sample.
\( D \) is the maximum deviation.

Critical Value

\[ E = \left( -\ln(\beta/2) / 2n \right)^{1/2} \]

where \( E \) is the critical value
\( \beta \) is the significance level
\( n \) is the random sample size.
Random number generator:

$$SEED(i) = (8192 \times SEED(i-1)) \mod 67099547$$

where,

- $SEED(i-1) = $ Seed for random number generator
- $SEED(i) = $ The random number and the new seed.

Uniform random numbers:

$$Y = LL + (UL-LL) \times RNUM + 0.5$$

where,

- $Y = $ Uniform random number
- $LL = $ Lower limit of range
- $UL = $ Upper limit of range
- $RNUM = $ Random number

Negative Exponential random variates:

$$Y = -\ln(RNUM) \times MEAN$$

where,

- $Y = $ Neg. Exponential random variate
- $RNUM = $ Random number
- $MEAN = $ the mean

Normal random variates:

$$Y = MEAN + (SEE-6) \times SDEV$$

where,

- $Y = $ Normal random variate
- $MEAN = $ the mean
- $SEE = $ sum of 12 random numbers
- $SDEV = $ the standard deviation
APPENDIX B

USER'S MANUAL
Introduction

A) Hardware and Software Required
B) Getting Started
C) Changing the DATA Disk Drive
D) Using PC VSTAT
E) Finishing PC VSTAT
F) List the Contents of the Data Diskette
G) Copying Data From Hard Disk to Floppy Diskette
H) Formatting a New Diskette
I) When Something Goes Wrong
J) Miscellaneous
PC VSTAT
User's Manual

PC VSTAT is a general purpose, user friendly statistical package for the IBM Personal Computer and fully compatibles. It is a vector based system in which all data is stored in data vectors. These vectors are in fact files that are operated on by the programs of the package to produce a desired function. Thus, the name VSTAT is derived from Vector STATistical analysis.

PC VSTAT has been especially designed for the user who has little or no knowledge of computers or computer programming. For this reason, the user must interact with the programs, selecting options, answering questions and supplying required information. A menu format was selected to provide the user with a friendly environment.

It is important to note that the user of the package is assumed to have a working knowledge of statistics. Statistical theory is not explained within the package, as it is to be used as a tool, not a teacher.

A) Hardware and Software Required

To operate PC VSTAT the user must have an IBM Personal Computer or fully IBM PC compatible computer, the PC VSTAT diskette, either two floppy disk drives and a formatted data diskette, or one floppy disk drive and a hard disk (for data). It is assumed that if the user
has access to an IBM PC, that he/she will also have access to PC DOS, the operating system that comes with the IBM Personal Computer. A video monitor and a dot matrix or quality printer are also required.

B) Getting Started

To start up the system the user must place the DOS diskette in the A: disk drive and turn on the power switch. Once DOS has been loaded, a prompt will be displayed: A>. This signals that the computer system is ready to run an application program in the A: drive.

If you are using a system that supports a hard disk, DOS will probably be installed on the system and will boot automatically when you turn on the PC. If this is the case, you must transfer control from the C: disk drive to the A: disk drive, as the PC VSTAT diskette will be inserted in the A: drive. Changing the current logged disk drive can be accomplished by simply typing the letter of the desired drive. For example, if your hard disk drive loads DOS automatically, you will see C>, the C: drive prompt displayed on the screen. Simply type A>, and press the return key. You should now see A>, the A: drive prompt.

If you have loaded DOS manually, remove the DOS diskette from the A: disk drive. Insert the VSTAT diskette and type "VSTAT" (without the quotes) and press
the return key. In a few seconds the main menu will be displayed.

Insert the formatted data diskette into the B: disk drive if two floppy drives are being used. If a hard disk is being used as the second disk drive, your data will be stored on the C: drive. You may want to create a special directory on the hard disk for your VSTAT data vectors. You may consult the DOS manual for instructions on doing this.

C) Changing The DATA Disk Drive

It is important that you assign a disk drive for your data to be written to and read from. As described above, you may use a floppy disk drive and a diskette for the data, or a hard disk. If you are using a floppy diskette, the disk drive for the data must be set to B: If a hard disk is being used, set the disk drive to C:.

This option can be found on page 3 of the main menu and should be checked prior to each execution of PC VSTAT.

D) Using PC VSTAT

PC VSTAT provides three pages of menu items. The user must select an item by typing the corresponding number and pressing the return key. The program corresponding to the selected item will be executed and the user will be required to enter some information (i.e., the name of the vector of interest, etc.). The
user must enter this information through the keyboard and press the return key.

To assist the user of PC VSTAT, there is a help section (description section) located on the first page of the main menu. It is suggested that the inexperienced user read through each of the item descriptors prior to running any programs. This section provides a description of each statistic available.

E) Finishing PC VSTAT

To exit from PC VSTAT, the user must select the EXIT option which is located on each of the three pages of the main menu. Upon entering this option, a short message will be displayed prompting the user to remove the diskette. At this time the user will have been returned to DOS. After removing the PC VSTAT diskette and the data diskette, if applicable, the computer may be turned off.

F) Listing The Contents of The Data Diskette

Assuming that DOS has been loaded into your computer’s memory (see section B above), it is possible to view the contents or the names of the files residing on any disk. The dir command may be used to view files on the diskette that is located in the A: disk drive. To view the contents of a diskette in the B: disk drive, simply type "dir B:" (without the quotes). Or
"dir C:" to view the contents of the hard disk. It is important to note that this command is not a part of the PC VSTAT program and cannot be accomplished while PC VSTAT is running. A more detailed explanation of this can be found in the DOS manual.

G) Copying Data From Hard Disk to Floppy Disk

To copy a data file from hard disk to floppy you must have DOS loaded into your computer’s memory and a formatted diskette for the data in the A: disk drive.

COPY C:FILENAME A:

The COPY command above will copy from the C: drive, the file named FILENAME to the A: drive. Details of the COPY command can be found in the DOS manual.

H) Formatting a new diskette

Another DOS command is available for formatting a new data diskette. Assuming that DOS has been loaded into the computer’s memory (see section B above) and that the current logged disk drive is A: (you should have the A> prompt on your screen), the command FORMAT can be entered. DOS will instruct you to place the new diskette in the A: disk drive and press any key. A prompt will be displayed to inform you when the formatting has been completed. It is important to note that formatting is a DOS function, not part of IBM PC VSTAT. A diskette cannot be formatted while IBM PC
VSTAT is running. Consult the DOS manual for further instruction on the FORMAT command.

I) When Something Goes Wrong

Sometimes the user may need to get out of the system because of errors in information input or other unexpected errors. If for some reason a run-time or input/output error may occur, follow the instructions displayed on the screen by Turbo Pascal. If you are told to press the ESC key, do so. You will most likely be returned to DOS, the A> prompt will be displayed. From here you may enter VSTAT to execute the program again. If you cannot escape to DOS, remove the IBM PC VSTAT diskette, turn the power off and then on. If you were required to reboot the system, you must reload DOS (see section B above).

It may be the case that the previous user of PC VSTAT exited the package in some way other than via the exit option on the menu. If PC VSTAT is not exited properly, some flags may not be set for the next time the package is executed. If you are taken directly to the first page of the menu without the title page being displayed, the previous user did not exit the package properly. To set the flags as they should be, simply select the EXIT option on the menu. By exiting through the menu, all flags will be set. Now simply enter VSTAT as you would normally. Always use the EXIT option to
end your session with PC VSTAT.

J) Miscellaneous Information

1. If you are listing a large vector on the screen or attempting to view output that is too long to be displayed on the screen at one time, you may use the control key (Crtl) and the Num Lock key to temporarily stop the scrolling. Press the Crtl key and hold it down while you press the Num Lock key. To resume scrolling, press any key.

2. The Frequency Table, Histogram, Chi-Squared goodness-of-fit test and the Kolmogorov-Smirnov one sample test all use intervals whose quantity and size may be calculated by the program or entered by the user. A word of caution about this information: It is recommended that the user either allow the program to calculate the number of cells required, or base his/her estimate on the square root of the total number of observations of the vector.

It is also recommended that the user enter a cell with that is close to the value that the program suggests. Also, if the first cell value entered is equal to the minimum value of the vector, then there will be no empty intervals at the beginning of the table, and chances that all of your data will fit in the table are better than if the user guesses at these values. The minimum value will be displayed prior to
you entering the first cell value. It is recommended that you use the minimum value of the data vector as the first (interval) cell value.

3. Since two goodness-of-fit tests are available (Chi-Square and Kolmogorov-Smirnov one-sample), on the following page are tabulated the relative merits of the tests to help the user decide which test to use.
### GOODNESS-OF-FIT TEST

<table>
<thead>
<tr>
<th>Test Type</th>
<th>Kolmogorov-Smirnov</th>
<th>Chi-Squared</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>POWER:</strong></td>
<td>Superior</td>
<td>Good</td>
</tr>
<tr>
<td><strong>SAMPLE SIZE:</strong></td>
<td>Unrestricted</td>
<td>Less effective for small samples</td>
</tr>
<tr>
<td><strong>IMPLEMENTATION:</strong></td>
<td>Direct. May be read approx. from a Weibull plot.</td>
<td>Requires judgement to form grouping intervals.</td>
</tr>
</tbody>
</table>
APPENDIX C
PROGRAM FLOW CHARTS
START

DISPLAY MENU ITEMS

INPUT USER'S MENU SELECTION

YES END

NO END VSTAT?

EXECUTE APPROPRIATE PROGRAM

Program VSTAT
Procedure DESCRIBE
Program CREATE

START

ENTRY OF GROUPED DATA?

YES

IS IT A NEW OR OLD VECTOR?

NEW

OLD

NO

INPUT NAME OF VECTOR

YES

DOS VECTOR ALREADY EXIST?

NO

INPUT VALUES

YES

DOS VECTOR ALREADY EXIST?

NO

INPUT VALUES AND NUMBER OF OCCURRENCES

NO

WRITE VECTOR TO A FILE

RETURN TO MAIN MENU

YES
Program LISTV
Program EDITR
Procedure ADD

START

INPUT VALUE TO BE ADDED TO VECTOR AND OBSERVATION OF NEW VALUE

DISPLAY OBSERVATION AND VALUE TO ADD

VALUE AND OBSERVATION CORRECT?

YES

NO

ADD NEW VALUE TO VECTOR AT GIVEN OBSERVATION

WRITE MODIFIED VECTOR TO FILE

DO YOU WANT TO ADD ANOTHER ELEMENT?

YES

NO

RETURN TO EDIT
Procedure REMOVE

1. START

2. INPUT OBSERVATION OF VALUE TO BE REMOVED

3. DISPLAY OBSERVATION AND VALUE

4. IS THIS THE CORRECT OBSERVATION?
   - NO
   - YES

5. REMOVE VALUE FROM VECTOR

6. WRITE NEWLY EDITED VECTOR TO FILE

7. DO YOU WANT TO REMOVE ANOTHER VALUE?
   - YES
   - NO

8. RETURN TO EDIT
Procedure CHANGE

START

INPUT OBSERVATION TO BE CHANGED

DISPLAY OBSERVATION AND VALUE

IS THIS THE CORRECT OBSERVATION?

YES

INPUT NEW VALUE OF OBSERVATION

WRITE NEWLY ADJUSTED VALUE TO FILE

DO YOU WANT TO CHANGE ANOTHER VALUE?

RETURN TO EDIT

NO
Program MATHVEC
Program MATHSCA
Program SORTV

START

INPUT NAME OF VECTOR

DOES FILE EXIST?

YES

OPEN FILE AND READ VECTOR TO ARRAY

INPUT NAME OF VECTOR FOR RESULT

NO

SORT IN ASCENDING ORDER?

YES

PERFORM ASCENDING SORT

NO

PERFORM DESCENDING SORT

WRITE VECTOR RESULT TO FILE

DOES FILE ALREADY EXIST?

YES

RETURN TO MAIN MENU

NO

PERFORM DESCENDING SORT
Program NUMANAL

START

INPUT NAME OF VECTOR

DOES VECTOR EXIST?

YES

OPEN FILE AND READ VECTOR TO ARRAY V

PERFORM NUMERICAL ANALYSIS

SEND RESULTS TO PRINTER

RETURN TO MAIN MENU

NO

ERROR MESSAGE

DO YOU WANT A HARD COPY OF THE RESULTS?

YES

DISPLAY OUTPUT

NO
Program CCLIN
Program NONLIN

START

SELECT QUADRATIC OR CUBIC REGRESSION

INPUT THE VECTOR NAMES AND ALPHA LEVEL

DO FILES EXIST?

OPEN FILES AND READ VECTORS TO ARRAYS U AND W

CALCULATE REGRESSION EQUATION

PERFORM HYPOTHESIS TEST

SEND RESULTS TO PRINTER

RETURN TO MAIN MENU

DISPLAY OUTPUT

DO YOU WANT A HARD COPY OF THE RESULTS?

ERROR MESSAGE

NO

YES
Program FREQHIST
Program SCAT

START

INPUT NAMES OF TWO VECTORS

ERROR MESSAGE

DO VECTORS EXIST?

NO

DISPLAY SCATTER-GRAPH

YES

OPEN FILES AND READ VECTORS TO ARRAYS V AND W

CALCULATE MIN AND MAX VALUE FOR EACH VECTOR

SET UP SCALE

DO YOU WANT A HARD COPY?

NO

SEND RESULTS TO PRINTER

YES

RETURN TO MAIN MENU
Program FTEST

START

TEST = ?

1. INPUT MEANS AND VARIANCES

2. ENTER ALPHA LEVEL

INPUT NAMES OF TWO VECTORS

DO VECTORS EXIST?

YES

NO

ERROR MESSAGE

PERFORM F-TEST

DISPLAY RESULTS

DO YOU WANT A HARD COPY?

YES

RETURN TO MAIN MENU

SEND RESULTS TO PRINTER

CLOSE FILES AND READ VECTORS TO ARRAYS \( \theta \) AND \( \mu \)

CALCULATE MEANS AND VARIANCES
Program TTEST

START

TEST = ?

INPUT NAMES AND
VARIANCES

ENTER
ALPHA LEVEL

PERFORM T-TEST

DISPLAY RESULTS

DO YOU WANT A
HARD COPY?

SEND RESULTS
TO PRINTER

RETURN TO
MAIN MENU

DO VECTORS
EXIST?

YES

ERROR MESSAGE

NO

OPEN FILES AND
READ VECTORS TO
ARRAYS U AND W

CALCULATE MEANS AND
VARIANCES
Program CHI

START

ENTER NAME OF VECTOR AND TYPE OF DISTRIBUTION TO TEST DATA AGAINST

DOES VECTOR EXIST?

OPEN FILE AND READ VECTOR TO ARRAY

DO YOU KNOW POPULATION MEAN AND VARIANCE?

CALCULATE ESTIMATE OF POPULATION MEAN AND VARIANCE

SEND RESULTS TO PRINTER

RETURN TO MAIN MENU

INPUT ALPHA LEVEL AND INFORMATION FOR FREQUENCY TABLE

PERFORM CHI-SQUARE TEST

DO YOU WANT A HARD COPY?

RETURN TO MAIN MENU
Program KS1

START

INPUT VECTOR NAME

DOES VECTOR EXIST?

NO
ERROR MESSAGE

YES

OPEN FILE AND READ VECTOR TO ARRAY 0

INPUT INFORMATION FOR FREQUENCY TABLE AND ALPHA LEVEL

CALCULATE MAXIMUM DIFFERENCE

PERFORM KOHMOGROV-SMIRNOV ONE SAMPLE TEST

DISPLAY

Do you want a hard copy?

NO

YES

SEND RESULTS TO PRINTER

RETURN TO MAIN MENU
Program KS2
Program VALUE
Program RAND

START

INPUT NAME OF VECTOR FOR RESULT

TEST = ?

SELECT TYPE OF RANDOM NUMBER GENERATION

SELECT TYPE OF RANDOM SAMPLING

GENERATE SELECTED RANDOM NUMBERS

WRITE VECTOR TO FILE

RETURN TO MAIN MENU
Program PURGEV
PROGRAM VSTAT(INPUT,OUTPUT);

MAIN PROGRAM STORAGE SPECIFICATIONS

LABEL 10.15,20,25;

TYPE
STRING2=STRING[2];
STRING4=STRING[4];

VAR
DRIVE:STRING2;
VIEW:INTEGER;
STAT:CHAR;

ANS,RES:CHAR;
VECTORFILE: FILE OF REAL;

CHI:VECTORFILE:

VALUE:VECTORFILE:

FREMHIST:

FTEST:

MATHS:

VECTORFILE:

FTEST:

MATHS:

VECTORFILE:

FTEST:

MATHS:

VECTORFILE:

FTEST:

MATHS:

VECTORFILE:

FTEST:

MATHS:

VECTORFILE:

FTEST:

MATHS:

VECTORFILE:

FTEST:

MATHS:

VECTORFILE:

FTEST:

MATHS:

VECTORFILE:

FTEST:

MATHS:

VECTORFILE:

FTEST:

MATHS:

VECTORFILE:

FTEST:

MATHS:

VECTORFILE:

FTEST:

MATHS:

VECTORFILE:

FTEST:

MATHS:

CHAPTER 4

MAIN PROGRAM - VARIABLE DEFINITIONS

STAT - Used to signal which statistic to run.

DRIVE - Current assigned disk-drive for data.

ANS,RES - User's response to yes/no questions.

VECTORFILE - File name variable.

CHI - Program file to perform chi-squared test.

VALUE - Program file to evaluate statistic value.

FREMHIST - Program file to display frequency table and histogram.

FTEST - Program file to perform either F-test.

FTEST - Program file to perform either t-test or paired t-test.

MATHS - Program file to perform quadratic and cubic regression.

SCAT - Program file to perform correlation coefficient and linear regression.

NUMANAL - Program file to perform statistical measures.

SORT - Program file to perform data sort.

MATHS - Program file to perform vector/scalar math.

MATHS - Program file to perform vector math.

VECTORFILE:

KSI - Program file to perform Kolmogorov-Smirnov two sample test.

KSI - Program file to perform Kolmogorov-Smirnov one sample test.

SCAT - Program file to display scatter graph.

RAND - Program file to calculate random variables or perform random sampling.

CREATE - Program file to create vector or enter grouped data.

LISTV - Program file to list a vector.

EDITOR - Program file to edit a vector.

PURGEV - Program file to purge a vector.

VIEW - Signals to display title page only for first execution of VSTAT.

OPTION - User's menu selection.

ST - User's menu selection read as string.

ENTRY - Signals entry of invalid data by user.

OK - Signals initial run of VSTAT to display title page.

BEGIN
PROCEDURE CHECKES(VAR ST:STRING2; VAR ENTRY:STRING4);

PROCEDURE CHECK STORAGE SPECIFICATIONS

PROCEDURE CHECKES - VARIABLE DEFINITIONS

VALID - Set of valid data.

COUNT - Counter for number of valid digits in entry.

POS - Position in a string.

POSITION - An element in a string.

BEGIN

{ INITIALIZE COUNTER }

COUNT:=0;

{ DEFINE VALID DATA SET FOR FIRST DIGIT }
VALID:=['0','1'];
{ EXPAND ONE DIGIT ENTRY TO TWO DIGITS }
IF LENGTH(ST)=1 THEN
 INSERT('0',ST,1);
{ CHECK FIRST POSITION }
POSITION:=COPY(ST,1,1);
{ INCREMENT COUNTER IF FIRST DIGIT IS VALID }
IF POSITION IN VALID THEN
 COUNT:=COUNT+1;
{ REDEFINE VALID DATA SET FOR SECOND DIGIT }
IF POSITION='0' THEN
 VALID:=['1','9']
 ELSE
 VALID:=['0','5'];
{ CHECK SECOND POSITION }
POSITION:=COPY(ST,2,1);
{ INCREMENT COUNTER IF SECOND DIGIT IS VALID }
IF POSITION IN VALID THEN
 COUNT:=COUNT+1;
{ CHECK # OF VALID DIGITS AGAINST LENGTH OF ENTRY }
IF COUNT<LENGTH(ST) THEN
 ENTRY:="BAD"
 ELSE
 ENTRY:="GOOD"
 END(CHECKDES);
{ DESCRIPTIONS OF THE MENU ITEMS }
PROCEDURE DESCRIBE:
{ PROCEDURE DESCRIBE STORAGE SPECIFICATIONS }
VAR
 ENTRY:STRING4;
 PAGE:STRING3;
 ST:STRING2;
 CODE,DES:INTEGER;
 ANS:CHAR;
{ PROCEDURE DESCRIBE - VARIABLE DEFINITIONS }
 ENTRY - Signals invalid entry by user.
 PAGE - Page of main menu.
 ST - User's menu selection read as string.
 DES - User's menu selection as integer.
 CODE - Signals error in data type conversion.
 ANS - User's response to yes/no questions.
BEGIN
 ANS:='Y';
 WHILE ANS='Y' DO  { LOOP TO REPEAT PROCEDURE }
 BEGIN
 { LOOP FOR PAGE ONE OF MENU }
 PAGE:='ONE';
 WHILE PAGE='ONE' DO
 BEGIN
 { LOOP FOR BAD MENU ENTRY }
 PAGE:='TWO';
 ENTRY:='BAD';
 WHILE ENTRY='BAD' DO
 BEGIN
 { DISPLAY PAGE ONE OF DESCRIPTIONS MENU }
 CLS;Writeln;Writeln('DESCRIPTION OF MENU ITEMS');
 Writeln('PAGE 1:');
 Writeln('CREATE A VECTOR');
 Writeln('2 LIST A VECTOR');
 Writeln('3 EDIT A VECTOR');
 Writeln('4 MATH WITH TWO VECTORS');
 Writeln('5 MATH WITH A VECTOR AND A CONSTANT');
 Writeln('6 SORT A VECTOR');
 Writeln('7 STATISTICAL MEASURES');
 Writeln('8 CORRELATION COEFFICIENT OF TWO VECTORS');
 Writeln('9 LINEAR REGRESSION OF TWO VECTORS');
 Writeln('10 NONLINEAR REGRESSION OF TWO VECTORS');
 Writeln('11 FREQUENCY TABLE FOR A VECTOR');
WRITELn('12 HISTOGRAM FOR A VECTOR');
WRITELn('13 SCATTER GRAPH');
WRITELn('14 PAGE -2- OF THIS MENU');
WRITELn('15 EXIT (THIS MENU)');
{ CHECK FOR RETURN W/O ENTRY }
ST := '';
WHILE LENGTH(ST) ≤ 1 DO
  BEGIN
    WRITELN; WRITELN('Enter the number of the section you would like described and press RETURN');
    READLN(ST);
    { CALL PROCEDURE TO CHECK FOR INVALID ENTRY }
    CHECKDES(ST, ENTRY);
    END;
    { CONVERT ENTRY FROM STRING TO INTEGER }
    VAL(ST, DES, CODE);
    CLRSCR; WRITELN;
    { PRINT APPROPRIATE DESCRIPTION }
    CASE DES OF
    1: BEGIN
        WRITELN('CREATE A VECTOR'); WRITELN;
        WRITELN('This section allows you to name and create a vector and enter data into it. All vector names can be a maximum of 6 characters.'); WRITELN;
        END;
    2: BEGIN
        WRITELN('LIST A VECTOR'); WRITELN;
        WRITELN('This section allows you to list the contents of an existing vector'); WRITELN('on the terminal or the printer.'); WRITELN;
        END;
    3: BEGIN
        WRITELN('EDIT A VECTOR'); WRITELN;
        WRITELN('This section allows you to edit an existing data vector.'); WRITELN('You can add, remove, or change an element at the beginning, end or middle of the vector.'); WRITELN;
        END;
    4: BEGIN
        WRITELN('MATH WITH TWO VECTORS'); WRITELN;
        WRITELN('This section allows you to perform math (+,-,*,/) on a vector'); WRITELN('with a second vector and store the result in a third vector.'); WRITELN('NOTE: The original vectors must be the same length.'); WRITELN;
        END;
    5: BEGIN
        WRITELN('MATH WITH A VECTOR AND A CONSTANT'); WRITELN;
        WRITELN('This section allows you to perform math (+,-,*,/) on a vector with a scalar (constant) and to store the result in a second vector. Or, you can use one vector and take the square-root, common log (base 10), or natural log (base e) of the data stored in it and put the result in a second vector.'); WRITELN;
        END;
    6: BEGIN
        WRITELN('SORT A VECTOR'); WRITELN;
        WRITELN('This section allows you to sort a vector, either in ascending order or descending order, and to store the sorted data in a new vector.'); WRITELN;
        END;
    7: BEGIN
        WRITELN('STATISTICAL MEASURES'); WRITELN;
        WRITELN('This section allows you to calculate the mean, variance, standard deviation and mean deviation for a vector. It also lists the number of observations, the minimum and the maximum values.'); WRITELN;
        END;
    8: BEGIN
        WRITELN('CORRELATION COEFFICIENT OF TWO VECTORS'); WRITELN;
        WRITELN('This section allows you to calculate the correlation coefficient for two vectors. Also listed are the number of observations, mean, variance, and standard deviation of the data in each vector. The program will perform a test (if sample is ≤ 30, 2 if >30) to determine if the correlation coefficient (rho) is significantly different than zero or some other specified value. You must enter the desired significance level.'); WRITELN('0 ≤ ALPHA ≤ 1 and the value to test rho against.'); WRITELN('NOTE: The two vectors must be the same length.'); WRITELN;
        END;
    9: BEGIN
    END;
LINEAR REGRESSION OF TWO VECTORS
This section will perform a simple linear regression for two equal length vectors of data. After the initial regression line is calculated, you may perform a test to evaluate whether the intercept and slope of the equation can be considered equal to zero.

NONLINEAR REGRESSION OF TWO VECTORS
This section performs a curvilinear (quadratic or cubic) regression for two equal length data vectors using the Least-Square method. After the regression equation is calculated you may perform a test to determine whether your data fits a cubic equation, quadratic equation or a linear equation reasonably well.

FREQUENCY TABLE
This section allows you to produce a frequency table of the data in the vector. This table may be printed on the terminal screen or on the printer. The maximum number of intervals allowed in the table is 30, and the minimum is 3. The output includes the lower and upper limit of each interval, the frequency in each interval, the cumulative frequency, the relative frequency and the cumulative relative frequency. The program will calculate the number of cells required, if desired. You may enter the number of cells. You must enter the cell width and the lower limit of the first cell.

HISTOGRAM
allows you to produce a histogram of the data in a vector. It may be printed on the terminal screen or on the printer. The number of cells in the histogram is 30 and the minimum is 3. You may choose one of the following four histograms: Relative Frequency, Cumulative Relative Frequency, or one of the others calculated by the program. You may also enter the number of cells. You must enter the cell width and the lower limit of the first cell.

SCATTER GRAPH
This section produces a scatter graph of two equal length vectors of data. You must enter the names of the two vectors to be displayed.
BEGIN
    WRITELN('Enter the number of the section you want described and press RETURN.');
    READLN(ST);
END;
{ CALL PROCEDURE TO CHECK FOR INVALID ENTRY }
CHECKDES(ST,ENTRY);
END;
{ CONVERT STRING TO INTEGER }
VAL(ST,DES,CODE);
CLRSCR;WRITELN;
{ PRINT APPROPRIATE DESCRIPTION }
CASE DES OF
  1: BEGIN
      WRITELN('F-TEST (n and sigma-squared known)'); WRITELN;
      WRITELN('This section allows you to perform a one-sided F-test in the case where');
      WRITELN('the size and the variance of the two samples are available.'); WRITELN;
      WRITELN('Ho: sigma-squared of sample 1 = sigma-squared of sample 2.'); WRITELN;
      WRITELN('The program will calculate the F-statistic, probability and the degrees');
      WRITELN('of freedom for the test and will advise you whether or not to reject the');
      WRITELN('null hypothesis.');
      WRITELN('The program will calculate the F-statistic, probability and degrees');
      WRITELN('of freedom for the test and will advise you whether or not to reject the');
      WRITELN('null hypothesis.');
END;
  2: BEGIN
      WRITELN('F-TEST'); WRITELN;
      WRITELN('This section allows you to perform a one-sided F-test on two vectors');
      WRITELN('of data to test the hypothesis:'); WRITELN;
      WRITELN('Ho: sigma squared of vector A = sigma squared of vector B.'); WRITELN;
      WRITELN('The program will calculate the F-statistic, probability and the degrees of');
      WRITELN('freedom for the test and will advise you whether or not to reject the');
      WRITELN('null hypothesis.');
      WRITELN('null hypothesis.');
END;
  3: BEGIN
      WRITELN('T-TEST (n, mu and sigma-squared known)'); WRITELN;
      WRITELN('This section allows you to perform a two-tailed t-test in the case where');
      WRITELN('the size, the mean and the variance of the two samples is known.'); WRITELN;
      WRITELN('Ho: population mean of sample 1 = population mean of sample 2.'); WRITELN;
      WRITELN('The program will calculate the t-statistic, probability and degrees');
      WRITELN('of freedom for each of the two assumptions:');
      WRITELN('1. The population variances are unknown but assumed EQUAL.');
      WRITELN('2. The population variances are unknown but assumed UNEQUAL.');
      WRITELN('And will advise you whether or not to reject the null hypothesis.');</
      WRITELN('null hypothesis.');
END;
  4: BEGIN
      WRITELN('T-TEST'); WRITELN;
      WRITELN('This section allows you to perform a two tailed t-test on two');
      WRITELN('vectors of data to test the hypothesis:'); WRITELN;
      WRITELN('Ho: population mean of vector A = population mean of vector B.'); WRITELN;
      WRITELN('The program will calculate the t-statistic, probability and degrees');
      WRITELN('of freedom for each of the two assumptions:');
      WRITELN('1. The population variances are unknown but assumed EQUAL.');
      WRITELN('2. The population variances are unknown but assumed UNEQUAL.');
      WRITELN('And will advise you whether or not to reject the null hypothesis.');</
      WRITELN('null hypothesis.');
END;
  5: BEGIN
      WRITELN('PAIRED T-TEST'); WRITELN;
      WRITELN('This section allows you to perform a paired t-test on two');
      WRITELN('vectors of data to test the hypothesis:'); WRITELN;
      WRITELN('Ho: difference between the pop. mean of vector A and the pop. mean');
      WRITELN('of vector B is equal to 0. '); WRITELN;
      WRITELN('The program will calculate the t-statistic, probability and degrees');
      WRITELN('of freedom for the test and will advise you whether or not to reject');
      WRITELN('the null hypothesis.');</
      WRITELN('null hypothesis.');</
END;
  6: BEGIN
      WRITELN('CHI-SQUARE TEST'); WRITELN;
      WRITELN('This section will perform a Chi-Square goodness-of-fit test on a');
      WRITELN('vector of data to test the hypothesis:'); WRITELN;
      WRITELN('Ho: the distribution of your data : a Normal distribution.');
      WRITELN('OR');
      WRITELN('Ho: the distribution of your data: a Negative Exponential distr.'); WRITELN;
      WRITELN('You must choose which distribution you want to test your data vector');
      WRITELN('against and enter the desired significance level 0 < ALPHA < 1. '); WRITELN;
      WRITELN('The program will calculate the Chi-square value, probability, and the');
degrees of freedom, and will advise you whether or not to reject');
WRITE('the null hypothesis.');}
WRITE('It is recommended that this test be used for a sample size of 50 or more');
WRITE('data points. Use Kolmogorov-Smirnov one-sample test for a smaller sample.');
END:
7: BEGIN
WRITE('KOLMOGOROV-SMIRNOV TEST (One Sample)');
WRITE('This section will perform a Kolmogorov-Smirnov one-sample test');
WRITE('goodness of fit) for a vector of data to test the hypothesis:');
WRITE('Ho: the distribution of your data is a Normal distribution.');
WRITE('OR');
WRITE('Ho: the distribution of your data is a Negative Exponential distr.');
WRITE('The program will calculate the K-S value, probability and the degrees');
WRITE('of freedom for the test and will advise you whether or not to reject');
WRITE('the null hypothesis.');
END:
8: BEGIN
WRITE('KOLMOGOROV-SMIRNOV TEST (Two Sample)');
WRITE('This section will perform a Kolmogorov-Smirnov two-sample test');
WRITE('for two vectors of data to test the hypothesis:');
WRITE('Ho: the distribution of vector A is the distribution of vector B.');
WRITE('The test allows you to determine whether the two independent samples');
WRITE('come from the same population.');
WRITE('NOTE: The maximum number of data points allowed per vector is 150.');
END:
9: BEGIN
WRITE('ENTRY OF GROUPED DATA');
WRITE('This section allows you to create a vector of individual data points from');
WRITE('grouped data or you may add grouped data to an existing vector. You must');
WRITE('enter the mid-interval value and the frequency.');
END:
10: BEGIN
WRITE('VALUE FOR t, F, Z AND CHI');
WRITE('This section allows you to enter the value of a test statistic for');
WRITE('t, F, z, or Chi-square and the degrees of freedom (if applicable).');
WRITE('The program will give you the probability value for the right tail');
WRITE('of the selected statistic.');
END:
11: BEGIN
WRITE('RANDOM NUMBER GENERATOR');
WRITE('This section will generate random numbers (uniform) or random');
WRITE('variates (Normal or Negative Exponential) for you. You select how many');
WRITE('numbers you need generated and the program will store the random numbers');
WRITE('into a specified vector.');
END:
12: BEGIN
WRITE('RANDOM SAMPLING');
WRITE('This section will generate a random sampling plan (with or without');
WRITE('replacement) for you. You must specify the population size and sample size.');
WRITE('The program will put the selected item numbers into a specified vector.');
END:
13: BEGIN
WRITE('PURGE A VECTOR');
WRITE('This section allows you to delete a vector (data file) from the DATA disk.');
END:
14: BEGIN
WRITE('CHANGE DATA DISK DRIVE');
WRITE('This section will inform you of which disk drive is currently assigned');
WRITE('for your data. You may also change the currently assigned disk drive.');
WRITE('If you are using a computer that supports a hard disk drive, you will want');
WRITE('to assign the C: data disk drive. If your second disk drive is for floppy');
WRITE('diskettes, you will want to assign the B: disk drive.');
WRITE('NOTE: B: is the default data disk drive value.');
END:
15: PAGE='ONE';
16: EXIT;
ENDOF(CASE);
END:{14:}
15: EXIT;
ENDOF(CASE);
END{WHILE PAGE='ONE'}
{ REPEAT THIS SECTION ? }
ANS:='M';
WHILE (ANS='Y') AND (ANS='N') DO
BEGIN
  WRITELN('Would you like to see another section described? (Y/N)');
  READLN(ANS);
END;
END{DESCRIBE};

{ CHANGE DATA DISK-DRIVE }
PROCEDURE DISKDRV(VAR DRIVE:STRING2);
{ PROCEDURE DISKDRV STORAGE SPECIFICATION }
VAR
  ANS:CHAR;
BEGIN
  ANS:='Y';
  WHILE ANS='Y' DO
BEGIN
    { DISPLAY CURRENT DATA DISK-DRIVE }
    CLRSCR;WRITELN('CHANGE DATA DISK-DRIVE');WRITELN;
    ANS:='M';
    WHILE (ANS='Y')AND(ANS='N') DO
BEGIN
      WRITELN('The current disk-drive assigned for your data is ',DRIVE);WRITELN;
      WRITELN('Do you want to change the disk-drive? (Y/N)');
      READLN(ANS);
      IF ANS='Y' THEN BEGIN
        { CHANGE DATA DISK-DRIVE }
        IF DRIVE='B.' THEN
          DRIVE:='C:'
        ELSE
          DRIVE:='B:'
        END;
      END;
    END{DISKDRV};

{ MAIN PROGRAM - MAIN MENU }
BEGIN
  SET FLAG FOR DISPLAYING TITLE PAGE
  IF VIEWO1 THEN BEGIN
    CLRSCR;
    TEXTCOLOR(7);
    WRITELN;WRITELN;
    WRITELN('PC VSTAT')
    WRITELN('Menu Based')
    WRITELN('Data Vector Statistical Package')
    WRITELN('for the')
    WRITELN('IBM Personal Computer')
    WRITELN('Copyright Ohio University 1986')
    WRITELN('Project Director - Dr. Helmut T. Zwahlen')
    WRITELN('Programmer - Debbie Morley')
    WRITELN('Programmed in Turbo Pascal')
    DELAY(5000);
INSTRUCTIONS FOR FORMATTING NEW DISKETTE

BEGIN
    CLRSCR;WRITELN;WRITELN;WRITELN;
    WRITE('Do you need instructions for formatting a new diskette for your data? (Y/N)');
    READLN(ANS);
END;

IF ANS='Y' THEN
    BEGIN
        CLRSCR;WRITELN;WRITELN;WRITELN;
        WRITE('The instructions you are about to receive concerning disk formatting');
        WRITE('are to be executed by the operating system DOS. You cannot format a disk');
        WRITE('while VSTAT is running. Therefore, you may want to take notes on this ');
        WRITE('procedure, or print a hard copy of the instructions. Once the instructions');
        WRITE('have been displayed on the terminal screen, press and hold the SHIFT key');
        WRITE('and press the PrtSc key to obtain a hard copy.');
        WRITE('Press RETURN to continue.');
        READLN(RES);
        CLRSCR;WRITELN;WRITELN;
        WRITE('Formatting a New Diskette');
        WRITE('1. Load DOS into your computer memory.');
        WRITE('2. If the A: disk drive prompt , A) , is not displayed, then');
        WRITE('type A:');
        WRITE('3. Place the diskette to be formatted in the A: disk drive.');
        WRITE('4. Type FORMAT:');
        WRITE('5. Press any key to begin formatting.');
        WRITE('6. A message will be displayed on your terminal screen when format');
        WRITE('has been completed.');
        WRITE('7. Remove diskette from A: disk drive.');
        WRITE('NOTE: The DOS Reference Manual also has formatting instructions.');
        WRITE('Press any key to continue.');
        READLN(RES);
    END;

END;

END;

END;

VIEW:=1;
ANS:=Y;
WHILE ANS='Y' DO
    BEGIN
        ENTRY:='BAD';
        WHILE ENTRY='BAD' DO
            BEGIN
                MAIN MENU SELECTIONS
                10:CLRSCR;WRITELN;
                WRITE('PC VSTAT');
                WRITE('STATISTICAL PACKAGE');
                WRITE('COPYRIGHT OHIO UNIVERSITY 1986');
                WRITE('OPTIONS AVAILABLE');
                WRITE('PAGE 1-1');
                WRITE('1 DESCRIPTION OF MENU ITEMS');
WRITELN('CREATE A VECTOR');
WRITELN('LIST A VECTOR');
WRITELN('EDIT A VECTOR');
WRITELN('MATH WITH TWO VECTORS');
WRITELN('MATH WITH A VECTOR AND A CONSTANT');
WRITELN('SORT A VECTOR');
WRITELN('STATISTICAL MEASURES');
WRITELN('CORRELATION COEFFICIENT FOR TWO VECTORS');
WRITELN('LINEAR REGRESSION OF TWO VECTORS');
WRITELN('PAGE -2- OF THIS MENU');
WRITELN('PAGE -5- OF THIS MENU');
WRITELN('QUIT (EXIT THIS PACKAGE)');
{ CHECK FOR RETURN W/O ENTRY }
ST:='';
WHILE LENGTH(ST)<1 DO BEGIN
  WRITELN;WRITELN;WRITELN('Enter the number of the option you desire and press RETURN:');
  READLN(ST);
END;
{ CHECK FOR INVALID ENTRY }
CHECKDES(ST.ENTRY);
END;
{ CONVERT ENTRY FROM STRING TO INTEGER }
VAL(ST.OPTION.CODE);
{ EXECUTE APPROPRIATE PROCEDURE OR PROGRAM }
CLRSCR;
CASE OPTION OF
 1:DESCRIBE;
 2:BEGIN
    STAT:='I';
    ASSIGN(CREATE,'CREATE.CHN');
    CHAIN(CREATE);
    END;
 3:BEGIN
    ASSIGN(LISTV,'LISTV.CHN');
    CHAIN(LISTV);
    END;
 4:BEGIN
    ASSIGN(EDITR,'EDITR.CHN');
    CHAIN(EDITR);
    END;
 5:BEGIN
    ASSIGN(MATHVEC,'MATHVEC.CHN');
    CHAIN(MATHVEC);
    END;
 6:BEGIN
    ASSIGN(MATHSCA,'MATHSCA.CHN');
    CHAIN(MATHSCA);
    END;
 7:BEGIN
    ASSIGN(SORT,'SORT.CHN');
    CHAIN(SORT);
    END;
 8:BEGIN
    ASSIGN(NUMANAL,'NUMANAL.CHN');
    CHAIN(NUMANAL);
    END;
 9:BEGIN
    STAT:='C';
    ASSIGN(CCLIN,'CCLIN.CHN');
    CHAIN(CCLIN);
    END;
10:BEGIN
    STAT:='L';
    ASSIGN(CCLIN,'CCLIN.CHN');
    CHAIN(CCLIN);
    END;
11:BEGIN
    ENTRY:='BAD';
    WHILE ENTRY='BAD' DO
      { LOOP FOR INVALID ENTRY }
    BEGIN
IS:CLSCHR;WRITELN;
WRITELNM(' Enter the number of the option you desire and press RETURN:');
READ(ST);
END;
{ CHECK FOR INVALID ENTRY }
CHECKDES(ST,ENTRY);
END;
{ CONVERT ENTRY FROM STRING TO INTEGER }
VAL(ST,OVERRIDE,CODE);
{ EXECUTE APPROPRIATE PROGRAM }
CLSCHR: =
CASE OPTION OF
1:BEGIN
  ASSIGN(NONLIN,'NONLIN.CHN');
  CHAIN(NONLIN);
END;
2:BEGIN
  STAT:='';
  ASSIGN(FREQHIST,'FREQHIST.CHN');
  CHAIN(FREQHIST);
END;
3:BEGIN
  STAT:='';
  ASSIGN(FREQHIST,'FREQHIST.CHN');
  CHAIN(FREQHIST);
END;
4:BEGIN
  ASSIGN(SCAT,'SCAT.CHN');
  CHAIN(SCAT);
END;
5:BEGIN
  STAT:='';
  ASSIGN(FTEST,'FTEST.CHN');
  CHAIN(FTEST);
END;
6:BEGIN
  STAT:='';
  ASSIGN(FTEST,'FTEST.CHN');
  CHAIN(FTEST);
END;
7:BEGIN
  STAT:='';
  ASSIGN(TTEST,'TTEST.CHN');
  CHAIN(TTEST);
END;
8:BEGIN
  STAT:='';
  ASSIGN(TTEST,'TTEST.CHN');
  CHAIN(TTEST);
END;
9: BEGIN
  STAT:="C';
  ASSIGN(TTEST,'TTEST.CHN');
  CHAIN(TTEST);
  END;
10: BEGIN
  ASSIGN(CH1,'CH1.CHN');
  CHAIN(CH1);
  END;
11: GOTO 10; {PAGE 1}
12: BEGIN
  ENTRY:='BAD';
  WHILE ENTRY='BAD' DO
    { LOOP FOR INVALID ENTRY }
    BEGIN
      20: CLRSCR; WRITELN;
      WRITELN('PC VSTAT');
      WRITELN('STATISTICAL PACKAGE');
      WRITELN('COPYRIGHT OHIO UNIVERSITY 1986');
      WRITELN('-PAGE 3-');
      WRITELN('OPTIONS AVAILABLE');
      WRITELN('-PAGE 3-');
      WRITELN('1 KOLMOGOROV-SHIRNOV ONE-SAMPLE TEST');
      WRITELN('2 KOLMOGOROV-SHIRNOV TWO-SAMPLE TEST');
      WRITELN('3 ENTRY OF GROUPED DATA');
      WRITELN('4 VALUE FOR Z,F,t AND CHI-SQUARE');
      WRITELN('5 RANDOM NUMBER GENERATOR');
      WRITELN('6 RANDOM SAMPLING');
      WRITELN('7 PURGE A VECTOR');
      WRITELN('8 CHANGE DATA DISK-DRIVE');
      WRITELN('9 PAGE -1- OF THIS MENU');
      WRITELN('10 PAGE -2- OF THIS MENU');
      WRITELN('11 QUIT (EXIT THIS PACKAGE)');
    END;
    ( CHECK FOR RETURN W/O ENTRY )
    ST:='';
    WHILE LENGTH(ST)<1 DO
      BEGIN
        WRITELN('Enter the number of the option you want and press RETURN:');
        READLN(ST);
      END;
      ( CHECK FOR INVALID ENTRY )
      CHECKDES(ST,ENTRY);
    END;
    ( CHANGE ENTRY FROM STRING TO INTEGER )
    VAL(ST,OPTION,CODE);
    CLRSCR;
    ( EXECUTE APPROPRIATE PROGRAM )
    CASE OPTION OF
      1: BEGIN
        ASSIGN(KS1,'KS1.CHN');
        CHAIN(KS1);
      END;
      2: BEGIN
        ASSIGN(KS2,'KS2.CHN');
        CHAIN(KS2);
      END;
      3: BEGIN
        STAT:='2';
        ASSIGN(CREATE,'CREATE.CHN');
        CHAIN(CREATE);
      END;
      4: BEGIN
        ASSIGN(VALUE,'VALUE.CHN');
        CHAIN(VALUE);
      END;
      5: BEGIN
        STAT:='1';
        ASSIGN(RAND,'RAND.CHN');
        CHAIN(RAND);
      END;
      6: BEGIN
        STAT:='2';
        ASSIGN(RAND,'RAND.CHN');
CHAIN(RAND);
END;
7:BEGIN
ASSIGN(PURGEV,'PURGEV.CHN');
CHAIN(PURGEV);
END;
8:DISK(DRIVE);
9:GOTO 10; {PAGE 1}
10:GOTO 15; {PAGE 2}
11:GOTO 25; {EXIT}
END(OF CASE);
END;
12:GOTO 25; {QUIT}
END(OF CASE);
END;
13:GOTO 25; {QUIT}
END(OF CASE);
END;
14:GOTO 10;
ELSE
GOTO 10;
END(OF CASE);
END;
{EXIT VSTAT }
25:CLRSCR;WRITELN;WRITELN(' THANKS FOR USING "PC VSTAT" !');
WRITELN;WRITELN;WRITELN('You may now remove the diskette.');
VIEW:=0;
END.
{DATA ENTRY TO VECTOR}

PROGRAM CREATE;
{_MAIN PROGRAM STORAGE SPECIFICATIONS }

LABEL 10;
LABEL 5;

TYPE
VECTORPOINTER = "VECTORREC@RD:
VECTORRECORD = RECORD
  DATA:REAL;
  NEXT:VECTORPOINTER;
END:
RANGE = ARRAY[1, 1000] OF REAL;
STRING2=STRING[2];
STRING4=STRING[4];
STRING13=STRING[13];

VAR
DRIVE:STRING2;
VIEW:INTEGER;
STAT:CHAR;
VECT:ANS,RES:CHAR;
OK:BOOLEAN;
SUM,OC,CODE,I,N:INTEGER;
DATA:REAL;
ENTRY,STOC:STRING[4];
NAME:STRING[8];
ST:STRING[13];
WHICH:STRING[21];
VECTORFILE:FILE OF REAL;
VSTAT:FILE;
FIRSTDATA,LASTDATA,NEWDATA,NEWPTR:VECTORPOINTER;

MAIN PROGRAM - VARIABLE DEFINITIONS

DRIVE - Currently assigned disk-drive for data.
OK - True if file does not exist.
SUM - Total number of elements entered as grouped data.
ENTRY - Signals validity of user's input.
STOC - Number of occurrences of a value read as a string.
OC - Number of occurrences of a value.
CODE - Signals error in data type conversion.
I - Used as counter for vector array.
N - Number of elements in a vector.
DATA - An element of a vector.
VECT - Signals if entry of grouped data is to new or old vector.
STAT - Signals entry of grouped or single data.
RES,ANS - User's response to yes/no questions.
NAME - Name of vector to be created.
ST - Vector element read as a string.
WHICH - Label for statistic.
FIRSTDATA - Points to first element in linked-list.
LASTDATA - Points to last element in linked-list.
NEWDATA - Points to next empty position at end of linked-list.
NEWPTR - Points to position to insert data in linked-list.
VSTAT - File containing main menu.
VECTORFILE - File name variable.

CHECK VALIDITY OF MENU ENTRY
PROCEDURE CHECKDES(VAR ST:STRING2; VAR ENTRY:STRING4);
{PROCEDURE CHECK STORAGE SPECIFICATIONS }

TYPE
CHARACTERSET:SET OF CHAR;

VAR
VALID:CHARACTERSET;
COUNT,POS:INTEGER;
POSITION:CHAR;

PROCEDURE CHECKDES - VARIABLE DEFINITIONS

VALID - Set of valid data.
COUNT - Counter for number of valid digits in entry.
POS - Position in a string.
POSITION - An element in a string.
BEGIN
{ INITIALIZE COUNTER }
COUNT:=0;
{ DEFINE VALID DATA SET FOR FIRST DIGIT }
VALID::['0', '1'];
{ EXPAND ONE DIGIT ENTRY TO TWO DIGITS }
IF LENGTH(STOC)=1 THEN 
  INSERT('0', STOC, 1);
{ CHECK FIRST POSITION }
POSITION:=COPY(STOC, 1);
{ INCREMENT COUNTER IF FIRST DIGIT IS VALID }
IF POSITION IN VALID THEN
  COUNT:=COUNT+1;
{ REDEFINE VALID DATA SET FOR SECOND DIGIT }
IF POSITION='0' THEN
  VALID::['1'..'9'];
ELSE
  VALID::['0'..'5'];
{ CHECK SECOND POSITION }
POSITION:=COPY(STOC, 2);
{ INCREMENT COUNTER IF SECOND DIGIT IS VALID }
IF POSITION IN VALID THEN
  COUNT:=COUNT+1;
{ CHECK # OF VALID DIGITS AGAINST LENGTH OF ENTRY }
IF COUNT=LENGTH(STOC) THEN
  ENTRY::'GOOD';
ELSE
  ENTRY::'BAD';
END{CHECKDEJS};
{ CHECK # OF OCCURRENCES ENTERED }
PROCEDURE OCCHECK(VAR STOC, ENTRY: STRING4); 
{ PROCEDURE OCCEOX STORAGE SPECIFICATIONS }
TYPE CHARACT = SET OF CHAR;
VAR 
  POS, C: INTEGER;
  VALID: CHARACT;
{ PROCEDURE OCCEOX - VARIABLE DEFINITIONS }
  POSITION1 - First character in a string.
  POSITION - A character in a string.
  POS - Position in a string.
  C - Counter of number of zeros is a string.
  VALID - Set of valid characters.
BEGIN
{ INITIALIZE COUNTER }
C:=0;Z:=0;
{ DEFINE VALID DATA SET }
VALID::['0'..'9'];
{ INSERT ZEROS UNTIL ENTRY 4 DIGITS }
IF LENGTH(STOC)=4 THEN 
  REPEAT
    INSERT('0', STOC, 1);
  UNTIL LENGTH(STOC)=4;
{ CHECK FIRST POSITION OF ENTRY }
POSITION1:=COPY(STOC, 1);
{ INCREMENT VALID DIGIT COUNTER }
IF POSITION1 IN VALID THEN
  COUNT:=COUNT+1;
{ INCREMENT ZERO COUNTER }
C:=C+1;Z:=Z+1;
{ CHECK OTHER POSITIONS }
FOR POS:=2 TO 4 DO 
  BEGIN
    POSITION:=COPY(STOC, POS);
    IF POSITION IN VALID THEN
BEGIN
  { INCREMENT VALID DIGIT COUNTER }
  C:=C+1;
  IF POSITION='0' THEN
    { INCREMENT ZERO COUNTER }
  Z:=Z+1;
  END;
END;
ELSE
IF POSITION='1' THEN
BEGIN
  { INCREMENT VALID DIGIT COUNTER }
  C:=C+1;
  { REDEFINE VALID SET IF FIRST POSITION =1 }
  VALID:=['0'];
  { CHECK OTHER POSITIONS }
  FOR POS:=2 TO 4 DO
  BEGIN
    POSITION:=COPY(STOC,POS,1); 
    IF POSITION IN VALID THEN
      { INCREMENT VALID DIGIT COUNTER }
      C:=C+1;
    END;
  END;
END;
{ COMPARE # VALID DIGITS TO STRING LENGTH }
IF C=LENGTH(STOC) THEN
  ENTRY:="GOOD"
ELSE
  ENTRY:="BAD";
END(CHECK);
{ CHECK FOR INVALID DATA ENTRY }
PROCEDURE CHECK(VAR ST:STRING13; VAR ENTRY:STRING4);
{ PROCEDURE CHECK STORAGE SPECIFICATIONS }
TYPE
  CHARACTERSET=SET OF CHAR;
VAR
  VALID1, VALID2:CHARACTERSET;
  C, POS: INTEGER;
  POSITION: CHAR;

BEGIN
  PROCEDURE CHECK - VARIABLE DEFINITIONS
  VALID1, VALID2 - Sets of valid characters.
  C - Counter for number of valid digits in entry.
  POS - A position in a string.
  POSITION - A character in a string.

  { INITIALIZE COUNTER }
  C:=0;
  { DEFINE VALID DATA SETS. }
  VALID1:=['0'..'9', '.', 'E', 'e', '+', '-'];
  VALID2:=['0'..'9', 'E', 'e', '+', '-'];
  { CHECK FIRST POSITION OF ENTRY }
  POSITION:=COPY(ST,1,1);
  { INCREMENT COUNTER IF DIGIT VALID }
  IF POSITION IN VALID1 THEN
    C:=C+1;
  { CHECK OTHER POSITIONS }
  FOR POS:=2 TO LENGTH(ST) DO
  BEGIN
    POSITION:=COPY(ST,POS,1);
    { INCREMENT COUNTER IF DIGIT VALID }
    IF POSITION IN VALID1 THEN
      C:=C+1;
    { ALLOW ONLY ONE DECIMAL POINT }
    IF POSITION='.' THEN
      { REDEFINE VALID SET ONCE DECIMAL ENTERED }
      }
VALID2:=["0", "9"];  
END;  
{ COMPARISON OF VALID DIGITS TO STRING LENGTH }  
IF C(1)LENGTH(ST) THEN  
ENTRY:="BAD"  
ELSE  
ENTRY:="GOOD";  
END;  
{ MAIN PROGRAM }  
BEGIN  
ANS:="Y";  
WHILE ANS="Y" DO  
BEGIN  
IF STAT='1' THEN WHICH:='CREATE A VECTOR',  
IF STAT='2' THEN WHICH:='ENTRY OF GROUPED DATA';  
{ DISPLAY NAME OF SECTION }  
CLRSCR;WRITELN;WRITELN('"\n" WHICH);  
{ DETERMINE STATUS OF VECTOR FOR GROUPED ENTRY }  
IF STAT='2' THEN  
BEGIN  
VECT:="M";  
WHILE (VECT)="N"AND(VECT)="O") DO  
BEGIN  
WRITELN;WRITELN("Is this a New or Old vector that you want to add grouped data to?");  
WRITELN("(Enter N for New, O for old)");  
READLN(VECT);  
END;  
IF VECT='O' THEN STAT:="3";  
END;  
{ CREATE NEW VECTOR }  
IF (STAT='1')OR(STAT='2') THEN  
BEGIN  
REPEAT  
{ CHECK FOR RETURN W/O ENTRY }  
NAME:="";  
WHILE (LENGTH(NAME)<1)OR(LENGTH(NAME)>6) DO  
BEGIN  
WRITELN;WRITELN("What is the name of the vector you are about to create?");  
WRITELN("(Maximum 6 characters)");WRITELN;  
READLN(NAME);  
END;  
{ ASSIGN DISK-DRIVE TO FILENAME }  
NAME:=DRIVE+NAME;  
{ OPEN FILE }  
ASSIGNVECTORFILE.NAME;  
{ CHECK FOR EXISTING FILE WITH THIS NAME }  
\$I-\$RESET\$VECTORFILE\$I;  
OK:=(\$RESULT=0);  
IF OK THEN  
{ PRINT ERROR MESSAGE }  
BEGIN  
CLRSCR;WRITELN;WRITELN("WARNING - There already exists a file with this name.");  
WRITELN("NOTE: To erase a vector select the PURGE option from the main menu.");  
WRITELN;WRITELN("Press RETURN to continue.");  
READLN(RES);  
ANS:="M";  
WHILE (ANS)="Y"AND(ANS)="N") DO  
BEGIN  
WRITELN;WRITELN("Would you still like to create a vector? (Y/N)");  
READLN(ANS);CLRSCR;  
END;  
END;  
IF ANS="N" THEN GOTO 10;  
UNTIL NOT OK;  
{ OPEN A NEW FILE }  
ASSIGN\$VECTORFILE\$NAME;  
REWRITE\$VECTORFILE\$;  
END;  
{ ADD GROUPED DATA TO OLD VECTOR }  
IF STAT='3' THEN  
BEGIN
REPEAT
{ CHECK FOR RETURN W/O ENTRY }
NAME:="":
WHILE (LENGTH(NAME)<1) OR (LENGTH(NAME)<6) DO
BEGIN
WRITELN(WRITELN("What is the name of the vector you want to add grouped data to?");
READLN(NAME);
END;
{ ASSIGN DRIVE-DRIVE TO FILENAME }
NAME:=DRIVE+NAME;
{ OPEN FILE }
ASSIGN(VECTORFILE,NAME);
{ CHECK IF FILE EXISTS }
$I$RESET(VECTORFILE){$I+$};
OK:=(IOR(RESULT=0));
IF NOT OK THEN
{ PRINT ERROR MESSAGE }
BEGIN
CLRSCR;WRITELN;WRITELN("VECTOR ",NAME," DOES NOT EXIST!");
ANS:='Y';
WHILE (ANS(='Y') AND (ANS(='N')) DO
BEGIN
WRITELN(WRITELN(WRITELN("Would you like to try again to locate a vector? (Y/N)");
READLN(ANS);CLRSCR;
END;
IF ANS='N' THEN GOTO 10;
UNTIL OK;
{ OPEN A NEW FILE }
ASSIGN(VECTORFILE,NAME);
RESET(VECTORFILE);
SEEK(VECTORFILE,FILENAME(VECTORFILE));
END(STAT=3);
{ ENTER DATA INTO VECTOR }
IF STAT='1' THEN
BEGIN
{ CREATE A VECTOR - INPUT PROMPT }
CLRSCR;WRITELN;WRITELN("To create a vector, enter a value and press RETURN. Continue this process");
WRITELN(WRITELN(WRITELN("until all elements of the vector have been entered.");
WRITELN(WRITELN(WRITELN("Type END to signify the end of the vector."));
WRITELN("---------")

RITELN;
{ SET POINTER }
FIRSTDATA:=NIL;
{ ENTER DATA INTO VECTOR }
REPEAT
WRITELN("ENTER DATA:");
READLN(ST);
{ LOOP UNTIL USERS ENTERS "END" }
WHILE ST(='END') DO
BEGIN
{ CHECK FOR INVALID ENTRY }
CHECK(ST,ENTRY);
IF ENTRY='BAD' THEN
{ PRINT ERROR MESSAGE }
WRITELN("INVALID ENTRY - Try Again!");
ELSE
BEGIN
{ CHANGE DATA FROM STRING TO REAL }
VAL(ST,DATA,CODE);
{ WRITE NEW VECTOR TO A FILE }
NEWDATA:=DATA;
WRITE(VECTORFILE,DATA);
{ CREATE A LINKED LIST OF DATA ELEMENTS AND ADJUST POINTERS }
NEWDATA:=DATA;
IF FIRSTDATA=NIL
FIRSTDATA:=NEWDATA
ELSE
LASTDATA^.NEXT:=NEWDATA;
LASTDATA:=NEWDATA;
LASTDATA^.NEXT:=NIL;
END:
WRITELN('ENTER DATA:');
READLN(ST);
END;
UNTIL ST='END';
END1STAT:
l:
{ ENTRY OF GROUPED DATA }
IF (STAT=2) OR (STAT=3) THEN
BEGIN
  { INPUT PROMPT - GROUPED DATA }
  CLRSCR; WRITELN;
  WRITELN('For each entry you will be prompted to enter the value of the data point');
  WRITELN('and the frequency of occurrences of that value. To signal the end of data');
  WRITELN('Enter type END when prompted to ENTER VALUE');
  WRITELN('end of data');
  WRITELN('entry, type END when prompted to ENTER VALUE:');
  WRITELN('Enter VALUE:');
REWLN(ST);
CALL PROCEDURE CHECK FOR INVALID ENTRY }
CHECK(ST,ENTRY);
IF ENTRY='BAD' THEN
  PRINT ERROR MESSAGE
  WRITELN('INVALID ENTRY - Try Again !');
ELSE
BEGIN
  CHANGE ENTRY FROM STRING TO REAL
  VAL(ST,DATA,CODE);
  ENTER # OF OCCURRENCES
  ENTRY:='BAD';
  WHILE ENTRY='BAD' DO
BEGIN
  WRITELN('OCCURRENCES:');
  READLN(STOC);
  WRITELN;
  CALL PROCEDURE TO CHECK FOR INVALID ENTRY
  OCCHECW(STOC,ENTRY);
  IF ENTRY='BAD' THEN
    PRINT ERROR MESSAGE
    WRITELN('INVALID ENTRY - Try Again !');WRITELN;
  END;
  CHANGE FROM STRING TO INTEGER
  VAL(STOC,OC,CODE);
  CALCULATE TOTAL OF VALUES ENTERED
  SUM:=SUM+OC;
  IF MAXIMUM ALLOWED IS 1000 VALUES
  THEN
BEGIN
  PRINT ERROR MESSAGE
  CLRSCR; WRITELN>('The maximum vector size is 1000 data points.');
  WRITELN('Your last entry will not be entered in the vector.');WRITELN;
  REMOVE LAST ENTRY FROM TOTAL
  SUM:=SUM-OC;
  DO NOT ADD THIS VALUE TO VECTOR
  GOTO 5;
END;
{ WRITE NEW DATA TO FILE }
FOR I:=1 TO OC DO
BEGIN
  NEWDATA:=DATA;
  WRITE(VECTORFILE,DATA);
  CREATE LINKED-LIST OF DATA ELEMENTS AND ADJUST POINTERS
  NEWDATA*.DATA:=DATA;
  IF FIRSTDATA=NIL THEN
  FIRSTDATA:=NEWDATA
  ELSE
  LASTDATA*:NEXT:=NEWDATA;
  FIRSTDATA:=LASTDATA;
LASTDATA:=NEWDATA;
LASTDATA^.NEXT:=NIL;
END;
5:END;
{ PROMPT USER TO INPUT NEXT VALUE }
WRITELN('ENTER VALUE:');
READLN(ST);
END;
UNTIL ST='END';
END(STAT-2);
{ CLOSE FILE }
CLOSE(VectorFile);
{ REPEAT THIS SECTION ? }
ANS:='N';
WHILE (ANS('Y')) AND (ANS('N')) DO
BEGIN
CLRSCR;WRITELN;WRITELN;
IF STAT='1' THEN
WRITELN('Would you like to create another vector ? (Y/N)');
IF (STAT='2') OR (STAT='3') THEN
WRITELN('Would you like to enter grouped data into another vector ? (Y/N)');
READLN(ANS);
END;
END;
{ RETURN TO MAIN MENU }
IO:ASSIGN(VSTAT,'VSTAT.COM');
EXECUTE(VSTAT);
END(CREATE).
{ LIST A VECTOR }

PROGRAM LISTV;
{ MAIN PROGRAM STORAGE SPECIFICATIONS }
LABEL 5;
TYPE
  STRING2-STRING[2];
  SB-STRING[8];
VAR
  DRIVE:STRING2;
  VIEW:INTEGER;
  N:INTEGER;
  LISTT:SB;
  OK:BOOLEAN;
  ANS:CHAR;
  VECTORFILE:FILE
  VSTAT:FILE;

{ MAIN PROGRAM - VARIABLE DEFINITIONS }

  DRIVE - Assigned disk-drive for data.
  N - Number of elements in vector.
  LISTT - Name of vector to list.
  OK - True if file exists.
  ANS - User's response to yes/no questions.
  VECTORFILE - File name variable.
  VSTAT - File containing main menu.

{ LIST A VECTOR ON THE TERMINAL SCREEN }
PROCEDURE TERM(VAR N:INTEGER;VAR LISTT:SB);
{ PROCEDURE TERM STORAGE SPECIFICATIONS }
VAR
  I:INTEGER; { COUNTER FOR ARRAY CONTAINING VECTOR }
  DATA:REAL; { AN ELEMENT OF THE VECTOR }
BEGIN
  CLRSCR;WRITELN;WRITELN('VECTOR',LISTT,'.');
  WRITELN('Observation Value');WRITELN;
  { PRINT VECTOR ON VDT }
  FOR I:=1 TO N DO
  BEGIN
    READ(VECTORFILE,DATA);
    WRITELN(I:4,DATA:13:4);
  END;
END;TERM;

{ LIST THE VECTOR ON THE PRINTER }
PROCEDURE PRNTR(VAR N:INTEGER;VAR LISTT:SB);
{ PROCEDURE PRNTR STORAGE SPECIFICATIONS }
VAR
  I:INTEGER; { COUNTER FOR ARRAY CONTAINING VECTOR }
  DATA:REAL; { AN ELEMENT OF THE VECTOR }
BEGIN
  CLRSCR;WRITELN(LISTT,'.');WRITELN(LISTT,'VECTOR',LISTT,'.');
  WRITELN(LISTT,'');WRITELN(LISTT,'Observation Value');WRITELN(LISTT,'.);
  { PRINT VECTOR (HARD COPY) }
  FOR I:=1 TO N DO
  BEGIN
    READ(VECTORFILE,DATA);
    WRITELN('I:4,
    DATA:13:4);
  END;
END;PRNTR;

{ MAIN PROGRAM }
BEGIN
  ANS:='Y';
  WHILE ANS='Y' DO { LOOP TO REPEAT THIS SECTION }
  BEGIN
    CLRSCR;WRITELN('LIST A VECTOR');
    REPEAT { CHECK FOR RETURN W/O ENTRY }
      LISTT:='';
    WHILE (LENGTH(LISTT)(1)OR(LENGTH(LISTT)>6)) DO
      BEGIN
        "";
      END;
  ...
WRITELN(WRITE('What is the name of the vector you want to list ?'));
READLN(LISTT);

END:
{ ASSIGN DISK-DRIVE TO FILENAME }
LISTT:=DRIVE+LISTT;
{ OPEN FILE CONTAINING VECTOR TO BE LISTED }
ASSIGN(VECTORFILE, LISTT);
{ CHECK TO SEE IF VECTOR EXISTS }
{($I-)RESET(VECTORFILE){$I+};
OK:=(0result=0);
CLRSCR;
IF NOT OK THEN
{ PRINT ERROR MESSAGE }
BEGIN
WRITELN(WRITE('VECTOR ', LISTT, ' DOES NOT EXIST !'));
ANS:= 'N';
WHILE (ANS)='Y' AND (ANS)='N' DO
BEGIN
WRITELN(WRITE('Would you like to try again to LIST a vector ? (Y/N)'));
READLN(ANS);
END;
{ RETURN TO MAIN MENU }
IF ANS='N' THEN GOTO 5;
END;
UNTIL OK:
{ DETERMINE VECTOR SIZE }
N:=FILESIZE(VECTORFILE);
{ CHOOSE OUTPUT DEVICE }
ANS:= 'N';
WHILE (ANS)='Y' AND (ANS)='N' DO
BEGIN
CLRSCR; WRITELN(WRITE('Do you want a HARD COPY of the vector ? (Y/N)'));
READLN(ANS);
END;
IF ANS='Y' THEN
{ LIST ON PRINTER }
PRINTER(N, LISTT)
ELSE
{ LIST ON TERMINAL }
TER(M, LISTT);
{ CLOSE FILE }
CLOSE(VECTORFILE);
{ LIST ANOTHER VECTOR ? }
ANS:= 'N';
WHILE (ANS)='Y' AND (ANS)='N' DO
BEGIN
WRITELN(WRITE('Would you like to list another vector ? (Y/N)'));
READLN(ANS);
END;
IF ANS='N' THEN GOTO 5;
END;
{ RETURN TO MAIN MENU }
S:=ASSIGN(VSTAT, 'VSTAT.COM');
EXECUTE(VSTAT);
{ EDIT A VECTOR }

PROGRAM EDITR;
{ MAIN PROGRAM STORAGE SPECIFICATIONS }
LABEL 5;
TYPE
STRING2:STRING[2];
RANGE:ARRAY[1..1000] OF REAL;
STRING4:STRING[4];
SB:STRING[8];
STRINGI3:STRING[13];
VAR
DRIVE:STRING2;
VIEW:INTEGER;
VECTORFILE:FILE OF REAL;
VSTAT:FILE;
V:RANGE;
NAME2:SB;
I,N:INTEGER;
DATA:REAL;
OK:BOOLEAN;
ED,RES,ANS:CHAR;

MAIN PROGRAM - VARIABLE DEFINITIONS
DRIVE - Assigned disk-drive for data.
VECTORFILE - File name variable.
VSTAT - File containing main menu.
V - Array that vector is read into.
NAME2 - Name of vector to edit.
I - Vector array counter.
N - Number of elements in vector.
DATA - An element in a vector.
OK - True if file exists.
ED - Type of editing to perform.
RES,ANS - User's response for yes/no questions.

CHECK FOR INVALID DATA ENTRY
PROCEDURE CHECK VAR ST:STRINGI3; VAR ENTRY:STRING4);
{ CHECK STORAGE SPECIFICATIONS }
TYPE
CHARACTERSET=SET OF CHAR;
VAR
VALID1,VALID2:CHARACTERSET;
C,POS:INTEGER;
POSITION:CHAR;

PROCEDURE CHECK - VARIABLE DEFINITIONS
VALID1,VALID2 - Sets of valid characters.
C - Counter for number of valid digits in entry.
POS - A position in a string.
POSITION - A character in a string.

BEGIN
{ INITIALIZE COUNTER }
C:=0;
{ DEFINE VALID DATA SETS } VALID1:=["0"..'9'.,-."];
VALID2:=["0"..'9'.,.,,];
{ CHECK FIRST POSITION OF ENTRY }
POSITION:=COPY(ST,1,1);
{ INCREMENT COUNTER IF DIGIT VALID }
IF POSITION IN VALID1 THEN
C:=C+1;
{ CHECK OTHER POSITIONS }
FOR POS:=2 TO LENGTH(ST) DO
BEGIN
POSITION:=COPY(ST,POS,1);
{ INCREMENT COUNTER IF DIGIT VALID }
IF POSITION IN VALID2 THEN
C:=C+1;
{ ALLOW ONLY ONE DECIMAL POINT }
IF POSITION='.' THEN
  { REDEFINE VALID SET ONCE DECIMAL ENTERED }
  VALID2=['0'..'9'];
END;
{ COMARE # VALID DIGITS TO STRING LENGTH }
IF C()LENGTH(ST) THEN
  ENTRY='BAD'
ELSE
  ENTRY='GOOD';
END(ENTRY);
{ READ FILE TO AN ARRAY }
PROCEDURE READFI(VAR N:INTEGER; VAR V:RANGE);
{ PROCEDURE READFI STORAGE SPECIFICATIONS }
VAR
  I:INTEGER;  { COUNTER FOR ARRAY CONTAINING VECTOR }
  DATA:REAL;  { AN ELEMENT OF THE VECTOR }
BEGIN
  { DETERMINE VECTOR SIZE }
  N:=FILESIZE(VECTORFILE);
  FOR I:=1 TO N DO
    BEGIN
      READ(VECTORFILE,DATA);
      V[I]:=DATA;
    END;
  CLOSE VECTORFILE;
END(READFI);
{ WRITE AN ARRAY TO A FILE }
PROCEDURE WRITEFI(VAR V:RANGE;VAR N:INTEGER);
{ PROCEDURE WRITEFI STORAGE SPECIFICATIONS }
VAR
  I:INTEGER;  { COUNTER FOR ARRAY CONTAINING VECTOR }
  DATA:REAL;  { AN ELEMENT OF THE VECTOR }
BEGIN
  { WRITE ARRAY V TO A FILE }
  FOR I:=1 TO N DO
    BEGIN
      DATA:=V[I];
      WRITE(VECTORFILE,DATA);
    END;
  CLOSE VECTORFILE;
END(WRITEFI);
{ ADD AN ELEMENT TO A VECTOR }
PROCEDURE ADD(VAR N:INTEGER; VAR NAME2:SB);
{ PROCEDURE ADD STORAGE SPECIFICATIONS }
VAR
  DATA,NEWEL:REAL;
  NN,CODE,I,OBs:INTEGER;
  ENTRY,STOBS:STRING[14];
  ANS,RES:CHAR;
  ST:STRING[13];
{ PROCEDURE ADD - VARIABLE DEFINITIONS }
BEGIN
  NN:=N;
  ANS:='Y';
WHILE ANS='Y' DO    ( LOOP TO REPEAT THIS SECTION )
BEGIN
    { INCREMENT LENGTH OF VECTOR }
    NN:=NN+1;
    { DEFINE ELEMENT TO BE ADDED }
    CLSCHR;WRITELN;WRITELN( 'ADDING AN ELEMENT TO THE VECTOR:' );
    WRITELN;WRITELN;
    WRITELN( 'Type the value of the number you want to add to the vector after' );
    WRITELN( 'the prompt 'ENTER DATA'.' );
    WRITELN;WRITELN( '------------------------------------------' );
    ENTRY:='BAD';
    WHILE ENTRY='BAD' DO
    BEGIN
    { INPUT DATA ELEMENT TO ADD TO VECTOR }
    WRITELN;WRITELN( 'ENTER DATA:' );
    READLN(ST);
    { CALL PROCEDURE TO CHECK FOR INVALID ENTRY }
    CHECK(ST,ENTRY);
    IF ENTRY='BAD' THEN
       { PRINT ERROR MESSAGE }
       WRITELN( 'INVALID ENTRY - Try Again !' )
    END:
    { CHANGE DATA FROM STRING TO REAL }
    VAL(ST,NEWEL,CODE);
    { DETERMINE PLACEMENT OF NEW ELEMENT IN VECTOR }
    RES:='N';
    WHILE RES='N' DO
    BEGIN
    OBS:=0;
    WHILE (OBS(1)OR(OBS)NN) DO
    BEGIN
    CLSCHR;WRITELN;WRITELN( 'What OBSERVATION is this new value ? (observation 1-',NN,')' );
    READLN(STOBS);
    { CHANGE OBSERVATION FROM STRING TO INTEGER }
    VAL(STOBS,NEWEL,CODE);
    END:
    IF OBS=NN THEN
       { ADD ELEMENT TO END OF VECTOR }
    BEGIN
    { OPEN FILE CONTAINING VECTOR }
    RESET(VECTORFILE);
    { LOCATE END OF FILE }
    SEEK(VECTORFILE,FILESIZE(VECTORFILE));
    DATA:=NEWEL;
    { WRITE NEW ELEMENT TO END OF FILE }
    WRITE(VECTORFILE,DATA);
    RES:='Y';
    END
    ELSE
       { ADD ELEMENT OTHER THAN AT END }
    BEGIN
    { OPEN FILE CONTAINING VECTOR }
    RESET(VECTORFILE);
    { SEARCH FOR POSITION IN VECTOR FOR NEW ELEMENT }
    SEEK(VECTORFILE,OBS-1);
    READ(VECTORFILE,DATA);
    { DOUBLE-CHECK PLACEMENT OF NEW ELEMENT IN VECTOR }
    RES:='N';
    WHILE (RES='Y')AND(RES='N') DO
    BEGIN
    CLSCHR;WRITELN;WRITELN( 'Observation ',OBS,' will be given the value ',NEWEL:8:4); 
    WRITELN;WRITELN( 'Is this the correct observation ? (Y/N)' );
    READLN(RES);
    END;
    { OPEN VECTOR FILE }
    RESET(VECTORFILE);
    { READ VECTOR TO ARRAY }
    READFI(N,V);
    { ADJUST LOCATIONS OF VECTOR ELEMENTS }
    I:=NN;
    REPEAT
    V[I]:=V[I-1];
I:=I-1;
UNTIL I=OBS;
{ ADD NEW VECTOR ELEMENT }
V[OBS]:=NEWEL;
{ WRITE THE NEW VECTOR TO A FILE }
WRITE(VECTORFILE);
WRITEFi(V,NN);
END;
{ CLOSE FILE }
CLOSE(VECTORFILE);
{ ADD ANOTHER ELEMENT TO THE VECTOR ? }
ANS:="M';
WHILE (ANS<>"Y")AND(ANS<>"N") DO
BEGIN
CLRSCR;WRITELN;WRITELN('Would you like to add another element to VECTOR ',NAME2,'? (Y/N)');
READLN(ANS);
END:
N:=NN;
END;
ADD11;

REHOVE AN ELEMENT FROM A VECTOR
PROCEDURE REHOVE(VAR N: INTEGER; VAR NAME2: S8);
{ PROCEDURE REMOVE STORAGE SPECIFICATIONS }
VAR
CODE, I, OBS: INTEGER;
DATA: REAL;
STOBS: STRING[4];
ANS, RES: CHAR;
{ PROCEDURE REMOVE - VARIABLE DEFINITIONS }
CODE - Signals error in data type conversion.
I - Vector array counter.
OBS - Observation to be removed.
STOBS - Observation read as string.
DATA - An element of the data vector.
ANS, RES - User's response to yes/no questions.
BEGIN
ANS:="Y';
WHILE ANS='Y' DO  { LOOP TO REPEAT EDIT SECTION }
BEGIN
RES:="N';
WHILE RES='N' DO  { LOOP FOR SELECTION OF INCORRECT OBSERVATION }
BEGIN
OBS: =O;
WHILE (OBS(1)) OR (OBS(N)) DO
BEGIN
{ PROMPT TO INPUT OBSERVATION TO BE REMOVED }
CLRSCR;WRITELN;WRITELN('Which OBSERVATION of the vector would you like removed?');
WRITELN('Observations 1- ',OBS);)
READLN(STOBS);
{ CHANGE OBSERVATION FROM STRING TO INTEGER }
VAL(STOBS,OBS,CODE);
END:
{ OPEN FILE }
RESET(VECTORFILE);
{ DETERMINE VECTOR SIZE }
M:=FILESIZE(VECTORFILE);
{ SEARCH FOR OBSERVATION TO BE REMOVED }
SEEK(VECTORFILE,OBS-1);READ(VECTORFILE,DATA);
{ DOUBLE-CHECK ELEMENT TO BE REMOVED FROM VECTOR }
RES:="M';
WHILE (RES<>"Y") AND (RES<>"N") DO
BEGIN
CLRSCR;WRITELN;WRITELN('Observation Value');
WRITELN('Observation ',OBS,4,' ',DATA:13:4);
WRITELN('Is this the observation you want to REMOVE from the vector? (Y/N)');
READLN(ANS);
END:
{ WRITE THE NEW vector }
READLN(Res);
END;
END;
{OPEN FILE }
RESET(Vectorfile);
{READ VECTOR TO AN ARRAY }
READFI(N,V);
{REMOVE THE ELEMENT }
I:=Obs;
REPEAT
V[I]:=V[I+1];
I:=I+1;
UNTIL I>N;
{DECREMENT VECTOR LENGTH }
N:=N-1;
{WRITE THE NEW VECTOR TO A FILE }
REWRITE(Vectorfile);
WRITEFI(V,N);
{REMOVE ANOTHER ELEMENT FROM THE VECTOR }
ANS="#';
WHILE (ANS='Y')AND(ANS='N') DO
BEGIN
CLRSCR;WRITELN;WRITELN('Would you like to remove another element from VECTOR ',NAME2,' ? (Y/N)');
READLN(ANS);
END;
END REMOVE;)
{CHANGE AN ELEMENT IN A VECTOR }
PROCEDURE CHANGE(VAR N:INTEGER;VAR NAME2:SO);
{PROCEDURE CHANGE STORAGE SPECIFICATIONS }
VAR
ENTRY,STFROM:STRING[4];
STTO:STRING[15];
DATA,OLD,TTO:REAL;
CODE,FROM:INTEGER;
RES,ANS:CHAR;

{PROCEDURE CHANGE - VARIABLE DEFINITIONS }
ENTRY - Signals validity of user's entry.
FROM - Observation to change in vector.
STFROM - Observation to change read as string.
TTO - New value.
STT0 - New value read as string.
DATA - An element of the data vector.
CODE - Signals error in data type conversion.
OLD - Old value of the observation.
RES,ANS - User's response to yes/no questions.
BEGIN
ANS='Y';
WHILE ANS='Y' DO  { LOOP TO REPEAT SECTION }
BEGIN
RES='N';
WHILE RES='N' DO  { LOOP FOR SELECTION OF INCORRECT OBSERVATION }
BEGIN
{DETERMINE ELEMENT OF THE VECTOR TO BE CHANGED }
FROM:=0;
WHILE (FROM<1)OR(FROM>N) DO
BEGIN
{PROMPT FOR OBSERVATION TO BE CHANGED }
CLRSCR;WRITELN;WRITELN('Which OBSERVATION would you like to change ? (Observations 1-',N,')');
READLN(STFROM);
{CHANGE ELEMENT FROM STRING TO INTEGER }
VAL(STFROM,FROM,CODE);
END;
{SEARCH FOR OBSERVATION TO BE CHANGED }
RESET(Vectorfile);
SEEK(Vectorfile,FROM-1);
{READ VECTOR FROM FILE }
READ(VECTORFILE, DATA);
{ DOUBLE-CHECK THE ELEMENT TO BE CHANGED }
RES := 'N';
WHILE (RES = 'Y' AND (RES = 'N')) DO
BEGIN
  CLRSCR; WRITELN; WRITELN('Observation Value'); WRITELN;
  WRITELN('( ', FROM: 4, ', DATA: 13: 4); WRITELN; WRITELN;
  WRITELN('Is this the observation you want to CHANGE in the vector? (Y/N)');
  READLN(RES);
END;
CLRSCR; ENTRY := 'BAD';
WHILE ENTRY = 'BAD' DO
BEGIN
{ DETERMINE NEW VALUE OF ELEMENT TO BE CHANGED }
WRITELN; WRITELN('What would you like to change it to?');
READLN(STTO);
{ CALL PROCEDURE TO CHECK FOR INVALID ENTRY }
CHECK(STTO, ENTRY);
IF ENTRY = 'BAD' THEN
  { PRINT ERROR MESSAGE }
  WRITELN('INVALID ENTRY - Try Again !')
END;
{ CHANGE ELEMENT FROM STRING TO REAL }
VAL(STTO, ITO, CODE);
RESET(VECTORFILE);
{ DETERMINE VECTOR SIZE }
M := FILESIZE(VECTORFILE);
{ READ VECTOR TO A FILE }
READF(M, V);
{ DETERMINE ELEMENT IN VECTOR }
RESET(VECTORFILE);
SEEK(VECTORFILE, FROM: -1);
READ(VECTORFILE, DATA);
OLD := DATA;
V[FROM] := ITO;
{ WRITE EDITED VECTOR TO FILE }
REWRITE(VECTORFILE);
WRITELN(V, M);
{ CHANGE ANOTHER ELEMENT IN THE VECTOR? }
ANS := 'N';
WHILE (ANS = 'Y') AND (ANS = 'N') DO
BEGIN
  CLRSCR; WRITELN; WRITELN('Would you like to change another element in VECTOR ', NAME2, '? (Y/N)');
  READLN(ANS);
END;
END[CHANGE];
{ EDIT A VECTOR }
BEGIN
ANS := 'Y';
WHILE ANS = 'Y' DO
  { LOOP TO REPEAT THIS SECTION }
BEGIN
  CLRSCR; WRITELN; WRITELN('EDIT A VECTOR');
  REPEAT
  { LOCATE VECTOR TO EDIT }
  NAME2 := '';
  WHILE (LENGTH(NAME2) = 0) OR (LENGTH(NAME2) > 6) DO
  BEGIN
    WRITELN; WRITELN('What is the name of the vector you want to edit?');
    READLN(NAME2);
  END;
  { ATTACH DISK-DRIVE TO NAME }
  NAME2 := DRIVE + NAME2;
  { CHECK TO SEE IF VECTOR EXISTS }
  ASSIGN(VECTORFILE, NAME2);
  {$I+} RESET(VECTORFILE);{$I+};
  OK := (RESULT = 0);
  CLRSCR; IF NOT OK THEN
BEGIN
{ PRINT ERROR MESSAGE }
ANS::'N';
WHILE (ANS<> 'Y') AND (ANS<> 'N') DO
BEGIN
  WRITELN('VECTOR ',NAME2,' DOES NOT EXIST !');
  WRITELN('Would you like to try again to locate a vector to edit ? (Y/N)
  )
  READLN(ANS);
END:
IF ANS='N' THEN GOTO S;
END;
UNTIL OK;
{ DETERMINE VECTOR SIZE }
N:=FILESIZE(VECTORFILE);
{ AVAILABLE EDITING PROCEDURES }
ED::'M';
WHILE ('(ED<>'1') AND (ED<>'2') AND (ED<>'3')) DO
BEGIN
  CLRSCR;WRITELN('Would you like to...');
  WRITELN('1 ADD');
  WRITELN('2 REMOVE');
  WRITELN('3 CHANGE');
  WRITELN('...an element of vector ',NAME2, '?' );
  WRITELN(' Enter the number of your selection and press RETURN. ');
  READLN(ED);
END:
{ EXECUTE APPROPRIATE PROCEDURE }
CASE ED OF
'1': ADD(N,NAME2);
'2': REMOVE(N,NAME2);
'3': CHANGE(N,NAME2);
END OF CASE;
{ CLOSE FILE }
CLOSE(VECTORFILE);
{ EDIT ANOTHER VECTOR ? }
ANS::'N';
WHILE (ANS<> 'Y') AND (ANS<> 'N') DO
BEGIN
  CLRSCR;WRITELN('Would you like to edit another vector ? (Y/N)
  ');
  READLN(ANS);
END;
END;
{ RETURN TO MAIN MENU }
S::ASSIGN(VSTAT,"VSTAT.COM");
EXECUTE(VSTAT);
END(EDITR).
```plaintext
Program Mathvec;
{ Storage Specifications }
Label 10;
Type
  Range=Array[1..1000] of Real;
  Ray1=Array[1..2] of String[6];
  Ray2=Array[1..2] of Integer;
  String2=String[2];
Var
  Drive: String2;
  View: Integer;
  Ans, Res, Char;
  II, I, J: Integer;
  Data: Real;
  Ok: Boolean;
  MATH, WHICH: String[8];
  N: Ray2;
  Math: Ray1;
  V, W: Range;
  Vectorfile: File of Real;
  Vstat: File;

VARIABLE DEFINITIONS:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRIVE</td>
<td>Currently assigned disk-drive for data.</td>
</tr>
<tr>
<td>Ans</td>
<td>User's response to yes/no questions.</td>
</tr>
<tr>
<td>Res</td>
<td>User's response to menu selection.</td>
</tr>
<tr>
<td>J</td>
<td>Counter for two arrays containing vectors of interest.</td>
</tr>
<tr>
<td>II, I, J</td>
<td>Counter for number of elements in each array.</td>
</tr>
<tr>
<td>Data</td>
<td>A data element of a vector.</td>
</tr>
<tr>
<td>Ok</td>
<td>True if a file exists.</td>
</tr>
<tr>
<td>Which</td>
<td>String having values &quot;first&quot; or &quot;second&quot;.</td>
</tr>
<tr>
<td>MATH</td>
<td>Name of resulting vector.</td>
</tr>
<tr>
<td>N</td>
<td>Array containing the lengths of the two vectors.</td>
</tr>
<tr>
<td>Math</td>
<td>Names of the two vectors.</td>
</tr>
<tr>
<td>V, W</td>
<td>Arrays containing the data vectors.</td>
</tr>
<tr>
<td>Vectorfile</td>
<td>Name of file (vector) of interest.</td>
</tr>
<tr>
<td>Vstat</td>
<td>File containing the main menu.</td>
</tr>
</tbody>
</table>

++ ttt tt{ttttttt ttttttttttttttttttt ttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt
ANS:="M";
WHILE (ANS()'Y')AND(ANS()'M') DO
BEGIN
  WRITELN;WRITELN;WRITELN('Would you like to try again to locate a vector to do math with ?
  READLN(ANS);
  IF ANS="M" THEN GOTO 10;
END;
END;
UNTIL OK;

{ DETERMINE LENGTH OF VECTOR }
N[1]:=FILESIZE(VECTORFILE);
READ VECTOR TO AN ARRAY
FOR I:=1 TO N[1] DO
BEGIN
  READ(VECTORFILE,DATA);
  IF I=1 THEN
    V[I]:=DATA
  ELSE
    W[I]:=DATA;
END;
CLOSE FILE
WHICH:="second";

{ CHECK THAT VECTOR LENGTHS ARE EQUAL }
IF (N[1]<>N[2]) THEN
  PRINT ERROR MESSAGE
BEGIN
  CLRSCR;WRITELN;WRITELN('These two vectors are NOT the same length !');
  WRITELN;WRITELN('Press RETURN to continue:');
  READLN(ANS);GOTO 10;
END;

{ MATH OPERATIONS AVAILABLE }
RES:="0";
WHILE ((RES()'1')AND(RES()'2')AND(RES()'3')AND(RES()'4')) DO
BEGIN
  LOOP FOR INVALID MENU ENTRY
  CLRSCR;WRITELN;WRITELN('Would you like to:');WRITELN;WRITELN;
  WRITELN' 1 Add vector ','MATH[1],' to vector ','MATH[2],' ?';WRITELN;
  WRITELN' 2 Subtract vector ','MATH[2],' from vector ','MATH[1],' ?';WRITELN;
  WRITELN' 3 Multiply vector ','MATH[1],' by vector ','MATH[2],' ?';WRITELN;
  WRITELN' 4 Divide vector ','MATH[1],' by vector ','MATH[2],' ?';WRITELN;
  WRITELN;WRITELN('Enter the number of your selection and press RETURN. ');
  READLN(RES);
END;

{ PERFORM APPROPRIATE MATH OPERATION }
FOR I:=1 TO N[1] DO
BEGIN
  CASE RES OF
    '1': V[I]:=V[I]+W[I];
    '2': V[I]:=V[I]-W[I];
    '3': V[I]:=V[I]*W[I];
    '4': BEGIN
      CHECK FOR ZERO ELEMENTS BEFORE ATTEMPTING DIVISION
      FOR II:=1 TO N[1] DO
      BEGIN
        IF W[II]=0.0 THEN
        BEGIN
          PRINT ERROR MESSAGE
          CLRSCR;WRITELN;WRITELN('Division by ZERO not allowed ! ');
          GOTO 10;
        END;
        END;
        V[I]:=V[I]/W[I];
      END;
    END;
    END;
END;
{ CREATE FILE FOR NEW VECTOR }
REPEAT
{ CHECK FOR RETURN W/O ENTRY }
  MATH3:="";
  WHILE (LENGTH(MATH3)<1)OR(LENGTH(MATH3)>6) DO
  BEGIN
  END;
CLRSCR;WRITELN('What is the name of the vector that the result is to be stored in?');
WRITELN('Maximum 6 characters');
READLN(MATH3);
END:
{ ATTACH DISK-DRIVE TO NAME }
MATH3:=DRIVE$MATH3;
{ CHECK IF A VECTOR WITH THIS NAME ALREADY EXISTS }
ASSIGN(VECTORFILE,MATH3);
{$I-RESET(VECTORFILE){$I+};
OK:=IO(result=0);
IF OK THEN BEGIN
{ PRINT ERROR MESSAGE }
CLRSCR;WRITELN('WARNING - There already exists a vector with this name!');
WRITELN('NOTE: Select the PURGE option from the main menu to erase a vector.');
END;
UNTIL NOT OK:
{ WRITE RESULT TO NEW FILE }
ASSIGN(VECTORFILE,MATH3);
REW(VECTORFILE);
FOR I:=1 TO N[J] DO BEGIN
DATA:=V[I];
WRITE(VECTORFILE,DATA);
END;
{ CLOSE FILE }
CLOSE(VECTORFILE);
{ SELECT OUTPUT DEVICE }
RESET(VECTORFILE);
ANS:='M';
WHILE (ANS()='Y')AND(ANS()='N') DO BEGIN
CLRSCR;WRITELN('Do you want a HARD COPY of VECTOR ',MATH3,' ? (Y/N)');
READLN(ANS);
CLRSCR;
IF ANS='N' THEN { PRINT RESULT ON VDT }
BEGIN
WRITELN(' Vector ',MATH3,' : ');
WRITELN('Observation Value');
FOR I:=1 TO N[J] DO BEGIN
READ(VECTORFILE,DATA);
WRITELN(LST,'.',I:4,' ',DATA:13:4);
END;
END;
ELSE IF ANS='Y' THEN { SEND RESULTS TO PRINTER }
BEGIN
WRITELN(LST,' ');WRITELN(LST,' Vector ',MATH3,' : ');
WRITELN(LST,' ');WRITELN(LST,'Observation Value');
FOR I:=1 TO N[J] DO BEGIN
READ(VECTORFILE,DATA);
WRITELN(LST,'.',I:4,' ',DATA:13:4);
END;
END;
{ CLOSE FILE }
CLOSE(VECTORFILE);
{ REPEAT THIS STATISTIC? }
ANS:='M';
WHILE (ANS()='Y')AND(ANS()='N') DO BEGIN
WRITELN('Would you like to do more MATH with vectors? (Y/N)');
READLN(ANS);
END;
{ RETURN TO MAIN MENU }
ID:ASSIGN(VSTAT,'VSTAT.COM');
EXECUTE(VSTAT);
END(MATHVEC).
PROGRAM MATHSCE;
{ MAIN PROGRAM STORAGE SPECIFICATIONS }
LABEL 10;
TYPE
  RANGE=ARRAY[1..1000] OF REAL;
  STRING13=STRING[13];
  STRING4=STRING[4];
  STRING2=STRING[2];
VAR
  DRIVE:STRING2;
  VIEW:INTEGER;
  R,ANS:CHAR;
  DATA,SCALAR:REAL;
  CODE,I,N:INTEGER;
  OK:BOOLEAN;
  ENTRY:STRING4;
  SMATH,SMATH2:STRING[8];
  ST:STRING13;
  V,W:RANGE;
  VECTORFILE:FILE OF REAL;
  VSTAT:FILE;

  { MAIN PROGRAM VARIABLE DEFINITIONS: }
  DRIVE - Current disk-drive for data.
  R - User's selection of menu item.
  ANS - User's response to yes/no questions.
  DATA - Data element of a vector.
  SCALAR - Number to perform math with vector.
  I - Counter for array containg vector.
  N - Length of the vector.
  CODE - Signals errors when converting strings.
  OK - True if vector exists on disk.
  ENTRY - String of 'good' or 'bad' (signals invalid entries).
  SMATH - Name of original vector.
  SMATH2 - Name of resulting vector.
  ST - Scalar in string format for validity check.
  V,W - Arrays containing data vectors.
  VECTORFILE - Name of file (vector) of interest.
  VSTAT - File containing main menu.

  PROCEDURE CHECK SCALAR ENTRY }
PROCEDURE CHECK(VAR ST:STRING13; VAR ENTRY:STRING4);
{ PROCEDURE CHECK STORAGE SPECIFICATIONS }
TYPE
  CHARACTERSET=SET OF CHAR;
VAR
  POS,COUNT:INTEGER;
  POSITION:CHAR;
  VALID1,VALID2:CHARACTERSET;
{ PROCEDURE CHECK VARIABLE DEFINITIONS: }
  POS - Position in a string.
  COUNT - Count valid positions in a string.
  POSITION - A character in a string.
  VALID1,VALID2 - Valid data entry sets.
BEGIN
  { DEFINE VALID DATA SETS };
  VALID1:=['0'..'9'];
  VALID2:=['0'..'9','.'];
  { INITIALIZE COUNTER }
  COUNT:=0;
  { CHECK FIRST DIGIT OF SCALAR } :
  POSITION:=COPY(ST,1,1); 
  IF POSITION IN VALID1 THEN
    COUNT:=COUNT+1;
  { REDEFINE VALID DATA SET ONCE DECIMAL ENTERED }
  IF POSITION='.' THEN
    VALID2:=['0'..'9'];
{ CHECK REMAINING DIGITS }
FOR POS:=2 TO LENGTH(ST) DO
BEGIN
  POSITION:=COPY(ST, POS, 1);
  IF POSITION IN VALID2 THEN
    COUNT:=COUNT+1;
  REDEFINE VALID DATA SET ONCE DECIMAL ENTERED
  ELSE
    IF POSITION='.' THEN
      VALID2:=['0'..'9'];
    END;
END;
{ COMPARE # OF VALID DIGITS TO LENGTH OF STRING }
IF COUNT<LENGTH(ST) THEN
  ENTRY:='BAD'
ELSE
  ENTRY:='GOOD';
END.

{ MAIN PROGRAM }
BEGIN
ANS:='Y';
WHILE ANS='Y' DO { LOOP TO REPEAT THIS STATISTIC IF DESIRED }
BEGIN
  CLRSCR; WRITELN; WRITELN('MATH WITH A VECTOR AND A SCALAR');
{ LOCATE VECTOR OF INTEREST }
  REPEAT
    { CHECK FOR RETURN W/O ENTRY }
    SMATH:='';
    WHILE (LENGTH(SMATH))<1 OR (LENGTH(SMATH))>6 DO
      BEGIN
        WRITELN('What is the name of the vector that you want to do math with ?');
        READLN(SMATH);
      END;
{ ATTACH DISK-DRIVE TO NAME }
    SMATH:=DRIVE+SMATH;
    SEARCH FOR FILE
    ASSIGN(VECTORFILE, SMATH);
    {$I-} RESET(VECTORFILE);$I-{RESULT=0};
    CLRSCR;
    IF NOT OK THEN
      { PRINT ERROR MESSAGE }
      BEGIN
        ANS:='M';
        WHILE (ANS='Y') AND (ANS='N') DO
          BEGIN
            WRITELN('VECTOR ' , SMATH, ' DOES NOT EXIST !');
            WRITELN('Would you like to try again to locate a vector ? (Y/N)');
            READLN(ANS);
          END;
        IF ANS='N' THEN GOTO 10;
      END;
    END;
{ DETERMINE SIZE OF VECTOR }
N:=-FILESIZE(VECTORFILE);
{ READ VECTOR TO AN ARRAY }
FOR I:=1 TO N DO
  BEGIN
    READ(VECTORFILE, DATA);
    V[I]:=DATA;
  END;
{ CLOSE FILE }
CLOSE(VECTORFILE);
{ DETERMINE MATH OPERATION DESIRED }
R:='0';
WHILE
  
  BOR
  
  END;
BEGIN
  CLRSCR; WRITELN; WRITELN('Would you like to :'); WRITELN;
  WRITELN('1 Add a scalar to VECTOR ' , SMATH, ' '); WRITELN;
  WRITELN('2 Subtract a scalar from VECTOR ' , SMATH, ' '); WRITELN;
  WRITELN('3 Multiply VECTOR ' , SMATH, ' by a scalar '); WRITELN;
  WRITELN('4 Divide VECTOR ' , SMATH, ' by a scalar '); WRITELN;
  WRITELN('5 Calculate the Square-Root of VECTOR ' , SMATH, ' ');
6 Calculate the Common Log (base 10) of VECTOR ';SMATH,' ?');
6 Calculate the Natural Log (base e) of VECTOR ';SMATH,' ?');WRITELN;WRITELN;
WRITELN;WRITELN('Enter the number of your selection and press RETURN:');
READLN(R);
END;
CLRSCR;
CASE R OF
'1', '2', '3', '4':
BEGIN
ENTRY:='BAD';
WHILE ENTRY='BAD' DO { LOOP FOR INVALID SCALAR ENTRY }
BEGIN
{ CHECK FOR RETURN W/O ENTRY }
ST:='';
WHILE LENGTH(ST)<1 DO
BEGIN
WRITELN;WRITELN('What is the value of the scalar ?');
READLN(ST);
END;
{ CHECK SCALAR ENTRY }
CHECK(ST,ENTRY);
IF ENTRY='GOOD' THEN
{ CHANGE ENTRY FROM STRING TO REAL }
VAL(ST,SCALAR,CODE)
ELSE
BEGIN
WRITELN;WRITELN('INVALID ENTRY !');
WRITELN;WRITELN('Press RETURN to continue:');
READLN(ANS);
END;
END;
IF R='4' THEN BEGIN
{ CHECK FOR ATTEMPTED DIVISION BY ZERO }
IF SCALAR=0.0 THEN
BEGIN
CLRSCR;WRITELN;WRITELN('You cannot divide vector ';SMATH,' by ZERO !');
WRITELN;WRITELN('Press RETURN to continue:');
READLN(ANS);
GOTO 10;
END;
END;
END;
'5', '6', '7':
BEGIN
{ CHECK FOR NEGATIVE OR ZERO ENTRIES FOR LN, LOG AND SQRT }
FOR I:=1 TO N DO
BEGIN
IF (V[I]<0.0) THEN
{ PRINT ERROR MESSAGE }
BEGIN
CLRSCR;WRITELN;WRITELN('This function CANNOT be performed because your vector');
WRITELN('contains either a zero or a negative entry.');
WRITELN;WRITELN('Press RETURN to continue:');
READLN(ANS);
GOTO 10;
END;
END;
END;
END OF CASE;
{ PERFORM APPROPRIATE MATH OPERATION }
FOR I:=1 TO N DO
BEGIN
CASE R OF
'1': V[I]:=V[I]+SCALAR;
'2': V[I]:=V[I]-SCALAR;
'3': V[I]:=V[I]*SCALAR;
'4': V[I]:=V[I]/SCALAR;
'5': V[I]:=SQRT(V[I]);
'6': V[I]:=LN(V[I])/(0.43429448);
'7': V[I]:=LN(V[I]);
END OF CASE;
CREATE NEW VECTOR FOR RESULT

REPEAT

{ CHECK FOR RETURN W/O ENTRY }
SMATH2: ''; 

WHILE (LENGTH(SMATH2)\(1\) OR LENGTH(SMATH2)\(6\)) DO

BEGIN

CLRSCR; WRITELN('What is the name of the vector that the result is to be stored in?');
WRITELN('You may use maximum 6 characters');
READLN(SMATH2);

END;

{ ATTACH DISK-DRIVE TO NAME }
SMATH2:=DRIVE+SMATH2;

{ CHECK IF VECTOR WITH THIS NAME ALREADY EXISTS }
ASSIGN VECTORFILE, SMATH2);
RESET(VECTORFILE);\(1\);OK:=I(result:=0);
IF OK THEN

BEGIN
CLRSCR; WRITELN('WARNING - There already exists a file with this name!');
WRITELN('Press RETURN to continue:');
READLN(ANS);

END;

UNTIL NOT OK;

{ WRITE NEW VECTOR TO FILE }
ASSIGN VECTORFILE, SMATH2;
REWITE(VECTORFILE);
FOR I:=1 TO N DO
BEGIN
DATA:={I};
WRITE(VECTORFILE, DATA);
END;

{ CLOSE FILE }
CLOSE (VECTORFILE);
{ LIST THE NEW VECTOR }
ASSIGN VECTORFILE, SMATH2;
RESET (VECTORFILE);

{ SELECT OUTPUT DEVICE }
ANS:='M';
WHILE (ANS='Y') AND (ANS='N') DO
BEGIN
CLRSCR; WRITELN('Do you want a HARD COPY of VECTOR ', SMATH2, '? (Y/N)');
READLN(ANS);

END;

CLRSCR;
IF ANS='N' THEN

{ PRINT ON VDT }
BEGIN
WRITELN('VECTOR ', SMATH2, ':');
WRITELN('Observation Value');
WRITELN;
FOR I:=1 TO N DO
BEGIN
READ (VECTORFILE, DATA);
WRITELN(' ', I:4, ', DATA:13:4);
END;
END;
IF ANS='Y' THEN

{ SEND OUTPUT TO PRINTER }
BEGIN
WRITELN(LST, ' ');WRITELN(LST, 'VECTOR ', SMATH2, ' '); 
WRITELN(LST, 'Observation Value');WRITELN(LST, ' '); 
FOR I:=1 TO N DO
BEGIN
READ (VECTORFILE, DATA);
WRITELN(LST, ' ',I:4, ', DATA:13:4);
END;
END;

{ CLOSE FILE }
CLOSE(VECTORFILE);

{ REPEAT THIS STATISTIC ? }
ANS:='M';
WHILE (ANS('Y') AND (ANS('N')) DO
  BEGIN
    WRITELN; WRITELN; WRITELN('Would you like to do more VECTOR-SCALAR math? (Y/N)');
    READLN(ANS);
  END;
END;

RETURN TO MAIN MENU
10: ASSIGN(VSTAT,'VSTAT.COM');
EXECUTE(VSTAT);
END(MATHSCA).
{ SORT A VECTOR }
PROGRAM SORT;
{ MAIN PROGRAM STORAGE SPECIFICATIONS }
LABEL 10;
TYPE
  RANGE=ARRAY[1..1000]OF REAL;
STRING2=STRING[2];
VAR
  DRIVE:STRING2;
  VIEW:INTEGER;
  RES,ANS:CHAR;
  I,N:INTEGER;
  DATA:REAL;
  ON:BOOLEAN;
  SORT1,SORT2:STRING[8];
  V:RANGE;
  VECTORFILE:FILE OF REAL;
  VSTAT:FILE;

MAIN PROGRAM VARIABLE DEFINITIONS
DRIVE - Current disk-drive of data.
RES - User's response to menu items.
ANS - User's response to yes/no questions.
I - Counter for array containing vector to sort.
N - Length of vector.
DATA - An element of a vector.
OK - True if file exists.
SORT1 - Name of original vector.
SORT2 - Name of sorted vector.
V - Array containing vector to be sorted.
VECTORFILE - Name of file containing vector to be sorted.
VSTAT - File containing main menu.

SORTING PROCEDURE
PROCEDURE SORTS(VAR V:RANGE; VAR N:INTEGER; VAR RES:CHAR);
{ PROCEDURE SORTS STORAGE SPECIFICATIONS }
VAR
  L,LAST:INTEGER;
  TEMP:REAL;
  SNAP:BOOLEAN;

PROCEDURE SORTS VARIABLE DEFINITIONS
L - Counter for array containing vector.
LAST - Last element in the vector.
TEMP - Data element held here temporarily while swapping.
SNAP - True if vector not completed sorted.

BEGIN
  SNAP:=TRUE;
  LAST:=N-1;
  WHILE (SNAP) DO
    BEGIN
      SNAP:=FALSE;
      FOR I:=1 TO LAST DO
      BEGIN
        IF RES='1' THEN
        BEGIN
          { SORT IN ASCENDING ORDER }
          IF V[I] > V[I+1] THEN
          BEGIN
            TEMP:=V[I];
            V[I]:=V[I+1];
            V[I+1]:=TEMP;
            SNAP:=TRUE
          END;
        END;
        IF RES='2' THEN
        BEGIN
          { SORT IN DESCENDING ORDER }
          IF V[I]<V[I+1] THEN
          BEGIN
            TEMP:=V[I];
            V[I]:=V[I+1];
            V[I+1]:=TEMP;
            SNAP:=TRUE
          END;
        END;
      END;
      IF RES='3' THEN
      BEGIN
        { SORT IN RANDOM ORDER }
        IF RANDOM(I,N) THEN
        BEGIN
          TEMP:=V[I];
          V[I]:=V[I+1];
          V[I+1]:=TEMP;
          SNAP:=TRUE
        END;
      END;
      IF RES='4' THEN
      BEGIN
        { SORT IN CUSTOM ORDER }
        IF (V[I]<V[I+1]) THEN
        BEGIN
          TEMP:=V[I];
          V[I]:=V[I+1];
          V[I+1]:=TEMP;
          SNAP:=TRUE
        END;
      END;
    END;
  END;
END;
BEGIN
TEMP:=V[I];
V[I]:=V[I+1];
V[I+1]:=TEMP;
SWAP:=TRUE;
END;
END;
LAST:=-LAST-1;
END;

{ MAIN PROGRAM - SORT A VECTOR }
BEGIN
ANS:='Y';
WHILE ANS='Y' DO { LOOP FOR REPEATING STATISTIC IF DESIRED }
BEGIN
CLRSCR;WRITELN;WRITELN('SORT A VECTOR');
{ DETERMINE VECTOR TO BE SORTED }
REPEAT
{ CHECK FOR RETURN W/O ENTRY }
SORT1:=''
WHILE (LENGTH(SORT1)<1)OR(LENGTH(SORT1)>6) DO
BEGIN
WRITELN;WRITELN('What is the name of the vector you want to sort ?');
READLN(SORT1);
END;
{ ATTACH DISK-DRIVE TO NAME }
SORT1:=DRIVE+SORT1;
{ CHECK TO SEE IF VECTOR EXISTS }
ASSIGN(VECTORFILE,SORT1);
(*1*)RESET(VVECTORFILE){$1*};
OK:=(IOresult=0);
CLRSCR;
IF NOT OK THEN
{ PRINT ERROR MESSAGE }
BEGIN
WRITELN;WRITELN('VECTOR ',SORT1,' DOES NOT EXIST !');
ANS:='M';
WHILE (ANS(>'Y')AND(ANS()<>'W')) DO
BEGIN
WRITELN;WRITELN('Would you like to try again to locate a vector ? (Y/M)');
READLN(ANS);
END;
IF ANS='N' THEN GOTO 10;
END;
UNTIL OK;
{ DETERMINE SIZE OF VECTOR }
N:=FILESIZE(VECTORFILE);
{ READ VECTOR TO AN ARRAY }
FOR I:=1 TO N DO
BEGIN
READ(VECTORFILE,DATA);
V[I]:=DATA;
END;
{ CLOSE FILE }
CLOSE(VVECTORFILE);
{ DETERMINE ORDER TO SORT }
RES:='O';
WHILE (RES(>'1')AND(RES()<>'2')) DO
BEGIN
CLRSCR;WRITELN;WRITELN('Do you want to sort vector ',SORT1,' in:');
WRITELN;WRITELN(' 1 Ascending order ?');
WRITELN;WRITELN(' 2 Descending order ?');
WRITELN;WRITELN('Enter the number of your selection and press RETURN:');
READLN(RES);
END;
{ CALL SORTING PROCEDURE }
SORTS(V,N,RES);
{ CREATE FILE FOR NEWLY SORTED VECTOR }
REPEAT
{ CHECK FOR RETURN W/O ENTRY }
SORT2:="";
WHILE (LENGTH(SORT2)<1)OR(LENGTH(SORT2)>6) DO
BEGIN
   CLRSCR;WRITELN;WRITELN('What is the name of the vector that the sorted result is to be stored in ?');
   WRITELN('NOTE: Maximum 6 characters. ');
   READLN(SORT2);
END;
{ ATTACH DISK-DRIVE TO NAME }
SORT2:=DRIVE+SORT2;
{ CHECK IF FILE WITH THIS NAME ALREADY EXIST }
ASSIGN(VECTORFILE,SORT2);
{$I-RESET(VECTORFILE){$I+};
OK:=TOresult=0);
IF OK THEN
{ PRINT ERROR MESSAGE }
BEGIN
   CLRSCR;WRITELN;WRITELN('WARNING - There already exists a vector with this name. ');
   WRITELN('NOTE: To erase a vector select the PURGE option from the main menu. ');
   WRITELN;WRITELN;WRITELN('Press RETURN to continue: ');
   READLN(ANS);
END;
UNTIL NOT OK;
{ OPEN NEW FILE }
ASSIGN(VECTORFILE,SORT2);
{ WRITE THE NEW VECTOR NEW FILE }
REWRIITE(VECTORFILE);
FOR I:=1 TO N DO
BEGIN
   DATA:=V[I];
   WRIITE(VECTORFILE,DATA);
END;
{ CLOSE NEW FILE }
CLOSE(VECTORFILE);
{ LIST THE NEW VECTOR }
ASSIGN(VECTORFILE,SORT2);
RESET(VECTORFILE);
N:=FILESIZE(VECTORFILE);
{ DETERMINE OUTPUT DEVICE }
ANS:="M";
WHILE (ANS='Y')AND(ANS='N')DO
BEGIN
   CLRSCR;WRITELN;WRITELN('Do you want a HARD COPY of VECTOR ',SORT2,' ? (Y/N) ');
   READLN(ANS);
END;
CLRSCR;,
IF ANS='N' THEN
{ LIST VECTOR ON VDT }
BEGIN
   WRITELN;WRITELN('VECTOR ',SORT2,':' );
   WRITELN;WRITELN('Observation Value');WRITELN;
   FOR I:=1 TO N DO
BEGIN
  READ(VECTORFILE,DATA);
  WRITELN(' ',I:4,' ',DATA:13:4);
END;
END;
IF ANS='Y' THEN
{ SEND OUTPUT TO PRINTER }
BEGIN
   WRITELN(LST,' ');WRITELN(LST,' VECTOR ',SORT2,':' );
   WRITELN(LST,'Observation Value');WRITELN(LST,' ');
   FOR I:=1 TO N DO
BEGIN
   READ(VECTORFILE,DATA);
   WRITELN(LST,' ',I:4,' ',DATA:13:4);
END;
END;
{ CLOSE FILE }
CLOSE(VECTORFILE);
{ SORT ANOTHER VECTOR ? }
ANS:="M";
WHILE (ANS='Y')AND(ANS='N') DO
BEGIN
  WRITELN;WRITELN;WRITELN('Do you want to sort another vector ? (Y/N)');
  READLN(ANS);
  END;
END;
{RETURN TO MAIN MENU}
10:ASSIGN(VSTAT,'VSTAT.COM');
EXECUTE(VSTAT);
END(SORT).
{STATISTICAL MEASURES}

PROGRAM NUMANAL;
{MAIN PROGRAM STORAGE SPECIFICATIONS}
LABEL 10;
TYPE
RANGEX = ARRAY[1..1000] OF REAL;
STRING2:STRING[2];
VAR
DRIVE:STRING2;
VIEW:INTEGER;
ANS:CHAR;
DATA,MAX,MIN,RANG,TOTAL,AVG,SUMX,SUMXX,VARI,STD,X,MDEV:REAL;
I,N:INTEGER;
OK:BOOLEAN;
V,XX:RANGEX;
NUMAN:STRING[8];
VECTORFILE:FILE OF REAL;
VSTAT:FILE;

{VARIABLE DEFINITIONS}

DRIVE - Current disk-drive for data.
ANS - User's response to yes/no questions.
DATA - Element of the vector.
MAX - Maximum value of vector.
MIN - Minimum value of vector.
RANG - Range of vector elements.
TOTAL,SUMX - Sum of all vector elements.
AVG - Average of vector elements.
SUMXX - Sum of squared elements.
SUMX2 - Sum of all elements squared.
VARI - Variance.
STD - Standard deviation.
X - Sum of each element minus average.
MDEV - Mean deviation.
I - Counter for array containing vector.
N - Length of vector.
OK - True if file exists.
V - Array containing vector elements.
XX - Array of each element minus average (absolute value).
NUMAN - Name of vector of interest.
VECTORFILE - Name of file containing vector.
VSTAT - File containing main menu.

BEGIN
ANS:='Y';
WHILE ANS='Y' DO { LOOP TO REPEAT STATISTIC IF DESIRED }
BEGIN
CLRSCR;WRITELN;WRITELN('STATISTICAL MEASURES');
{DETERMINE VECTOR OF INTEREST}
REPEAT
{CHECK FOR RETURN W/O ENTRY}
NUMAN:='';
WHILE (LENGTH(NUMAN)<1)OR(LENGTH(NUMAN)>6) DO
BEGIN
WRITELN;WRITELN('What is the name of the vector?');
READLN(NUMAN);
END;
{ATTACH DISK-DRIVE TO NAME}
NUMAN:=DRIVE+NUMAN;
{LOCATE DESIRED VECTOR}
ASSIGN(VECTORFILE(NUMAN));
{$I-)RESET(VECTORFILE){$I-});
OK:=(RESULT=0);
CLRSCR;
IF NOT OK THEN
{PRINT ERROR MESSAGE}
BEGIN
WRITELN;WRITELN('VECTOR ',NUMAN,' DOES NOT EXIST !');
ANS:='M';
WHILE (ANS<>'Y')AND(ANS<>'N') DO
BEGIN

WRITELN('Would you like to try again to locate a vector? (Y/N)'); READLN(ANS);
IF ANS='N' THEN GOTO 10;
END;
UNTIL OK;
{Determine vector length}
N:=FILESIZE(VECTORFILE); {Read file to array}
FOR I:=1 TO N DO BEGIN
  READ(VECTORFILE,DATA);
  V[I]:=DATA;
END;
{Close file}
CLOSE(VECTORFILE);
{Find max, min and range}
MAX:=V[I]; MIN:=V[I];
FOR I:=1 TO N DO BEGIN
  IF V[I]<MIN THEN MIN:=V[I];
  IF V[I]>MAX THEN MAX:=V[I];
END;
RANG:=MAX-MIN;
{Calculate mean}
TOTAL:=0.0;
FOR I:=1 TO N DO BEGIN
  TOTAL:=TOTAL+V[I];
END;
AVG:=TOTAL/N;
{Calculate variance}
SUMX:=TOTAL; SUMXX:=0.0;
FOR I:=1 TO N DO
  BEGIN
    SUMXX:=SUMXX+V[I]*V[I];
  END;
VARI:=(SUMXX-SUMX2/N)/(N-1);
{Calculate standard deviation}
STD:=SQRT(VARI);
{Calculate mean deviation}
X:=0.0;
FOR I:=1 TO N DO BEGIN
  XX[I]:=ABS(V[I]-AVG);
  X:=X+XX[I];
END;
MDEV:=X/N;
{Select output device}
ANS:='';
WHILE (ANS='Y') AND (ANS='N') DO
BEGIN
  CLRSCR; WRITELN; WRITELN('Do you want a hard copy of the output? (Y/N)');
  READLN(ANS);
END;
CLRSCR;
IF ANS='N' THEN
{Print output on VDT}
BEGIN
  WRITELN; WRITELN('Statistical Measures');
  WRITELN; WRITELN('Number of Observations = ',N);
  WRITELN; WRITELN('Minimum Value = ',MIN:13:4);
  WRITELN; WRITELN('Maximum Value = ',MAX:13:4);
  WRITELN; WRITELN('Range = ',RANG:13:4);
  WRITELN; WRITELN('Mean = ',AVG:13:4);
  WRITELN; WRITELN('Variance = ',VARI:13:4);
  WRITELN; WRITELN('Standard Deviation = ',STD:13:4);
  WRITELN; WRITELN('Mean Deviation = ',MDEV:13:4);
END
ELSE
{Send output to printer}
BEGIN
  WRITELN(LST); WRITELN(LST,' Statistical Measures');
WRITELN(LST); WRITELN(LST); WRITELN(LST, 'VECTOR ', NUMAN);
WRITELN(LST); WRITELN(LST, 'Number of Observations = ', N);
WRITELN(LST); WRITELN(LST, 'Minimum Value = ', MIN:13:4);
WRITELN(LST); WRITELN(LST, 'Maximum Value = ', MAX:13:4);
WRITELN(LST); WRITELN(LST, 'Range = ', RANG:13:4);
WRITELN(LST); WRITELN(LST, 'Mean = ', AVG:13:4);
WRITELN(LST); WRITELN(LST, 'Variance = ', VARI:13:4);
WRITELN(LST); WRITELN(LST, 'Standard Deviation = ', STD:13:4);
WRITELN(LST); WRITELN(LST, 'Mean Deviation = ', MDEV:13:4);
END;
{ CLOSE FILE }
CLOSE(VECTORFILE);  
{ REPEAT THIS STATISTIC ? }
ANS:= 'N';
WHILE (ANS('Y') AND (ANS('N')) DO
BEGIN
  WRITELN; WRITELN('Would you like to repeat this section ? (Y/N)');
  READLN(ANS);
END;
{ RETURN TO MAIN MENU }
I0: ASSIGN(VSTAT, 'VSTAT.COM');
EXECUTE(VSTAT); 
END(NUMANA).
{ CORRELATION COEFFICIENT & LINEAR REGRESSION }

PROGRAM CLIN;
{ MAIN PROGRAM STORAGE SPECIFICATIONS }
LABEL I0;

VARIABLE DEFINITIONS:

DRIVE - Current disk-drive of data.
STAT - ('C' or 'L') to determine which statistic to perform.
CODE - Error signal for converting types.
COUNT,COUNT1,COUNT2 - Counter for number of valid positions in an entry.
I,J - Counter for array containing vector of interest.
ADF - Degrees of freedom.
AVGV,AVGH - Average of first and second vectors respectively.
AVGV,AVGH - Variance of first and second vectors respectively.
STDV,STDH - Std. Dev. of first and second vectors respectively.
P - Probability of both tails.
SX - Sum of all data elements in first vector.
SY - Sum of all data elements in second vector.
SX2,SXH2 - Sum of all data elements squared from first vector.
SY2,SYH2 - Sum of all data elements squared from second vector.
SX2SY2 - Sum of each element from first vector times each element from second vector.
SC,SSX,SSY,SR2 - Calculations made towards determining correlation coefficient.
ALPHA - Critical region for significance testing.
RHO - Value (other than zero) to test correlation coefficient against.
ST - Calculated t- or z-value (for correlation coefficient testing).
DATA - An element of a vector.
CC - Correlation Coefficient.
A,B,SE - Calculations made towards determining t- and z-values for Lin. Reg. testing.
SA - Standard Deviation of 'b' (slope).
TA - Standard Deviation of 'a' (intercept).
TB - Calculated t-value for slope hypothesis test.
OK - True if file exists.
RESP - What value to test rho against.
OUTPUT - 'Y' for hard copy of results, 'N' to display results on VDT.
SIGN2 - Sign (+ or -) for each variable in the regression equation.
RES - User's response to menu items.
ANS - User's response to yes/no questions.
WHICH - String containing name of statistic.
ENTRY - Value 'bad' signals invalid entry.
C - Array of vector names.
N - Array of vector lengths.
V,W - Arrays containing first and second vectors, respectively.
VECTORFILE - Name of file containing vector of interest.
VSTAT - File containing main menu.

CALCULATE PROBABILITY (t-DISTRIBUTION)
PROCEDURE TVALUE(VAR ADF:INTEGER; VAR P,ST:REAL);
{ PROCEDURE TVALUE STORAGE SPECIFICATIONS }
LABEL 10;
VAR
R,RC,R2,RS,X,Y:REAL;
Z:INTEGER;

PROCEDURE TVALUE VARIABLE DEFINITIONS:
R,RC,R2,RS,X,Y - Calculations made towards determining probability.
Z - Counter for decrementing degrees of freedom in calculations.
ADF - Degrees of freedom.
ST - T-value.
P - Probability of both tails.
BEGIN
ST:=ABS(ST);
RC:=ARCTAN(ST/SQRT(ADF));
RC:=COS(R);R2:=RC*RC;RS:=SIN(R);
X:=1.0;
IF (ADF=1) THEN
Y:=R
ELSE
BEGIN IF (ODD(ADF)=FALSE) THEN BEGIN
Y:=1.0;
Z:=2;
REPEAT
X:=X*R2*(Z-1)/Z;
Y:=Y*X;
Z:=Z+2;
UNTIL Z > (ADF-2);
P:=1-Y*RS;
GOTO 10;
END:
Z:=3;
REPEAT
X:=X*R2*(Z-1)/Z;
Y:=Y*X*RC;
Z:=Z+2;
UNTIL Z > (ADF-2);
Y:=RS*Y;
END;
P:=1-Y*0.6366197723657157;
10:
IF (P < 0.000001) THEN P:=0.0;
END(TVALUE);

CALCULATE PROBABILITY (NORMAL DISTRIBUTION)
PROCEDURE ZVALUE(VAR P,ST:REAL);
{ PROCEDURE ZVALUE STORAGE SPECIFICATIONS }
VAR
R,W,W2,PT:REAL;

PROCEDURE ZVALUE VARIABLE DEFINITIONS:
R,W,W2,PT - Calculations made towards determining probability.
ST - Z-value.
P - Probability of both tails.
BEGIN
IF ABS(ST)>6 THEN BEGIN
BEGIN
P:=0.0;
END;
EXIT;
END;
R:=-1/(SQRT(2*PI)*2.83185307));
W:=-1/(ABS(ST)*0.2316419)+1;
PT:=(W*0.31938153)-(W*0.356563782)+(W*1.781477937)-(W*1.821255978)+(W*1.330274429);
P:=2*PT;
IF (P<0.000001) THEN P:=0.0;
END(ZVALUE);

PROCEDURE CHECK VAR STALPHA:STRING; VAR ENTRY:STRING;
VAR
COUNT:INTEGER;
POSITION:CHAR;
VALID:CHARSET;
BEGIN
{ INITIALIZE COUNTER }
COUNT:=0;
{ DEFINE VALID ENTRY SET }
VALID:='0','9';
{ CHECK FIRST POSITION }
POSITION:=COPY(STALPHA,1,1);
IF POSITION='.' THEN
ENTRY:='BAD';
ELSE
BEGIN
{ CHECK SECOND POSITION }
POSITION:=COPY(STALPHA,2,1);
IF POSITION IN VALID THEN
COUNT:=COUNT+1;
IF POSITION='0' THEN
{ REDEFINE VALID SET }
VALID:='1','9';
{ CHECK THIRD POSITION }
POSITION:=COPY(STALPHA,3,1);
IF POSITION IN VALID THEN
COUNT:=COUNT+1;
IF COUNT>2 THEN
ENTRY:='BAD';
ELSE
ENTRY:='GOOD';
END;
END(CHECK);

{ MAIN PROGRAM - CORRELATION COEFFICIENT & LINEAR REGRESSION }
BEGIN
IF STAT='C' THEN WHICH:='CORRELATION COEFFICIENT';
IF STAT='L' THEN WHICH:='LINEAR REGRESSION';
ANS:='Y';
WHILE ANS='Y' DO { LOOP TO REPEAT STATISTIC IF DESIRED }
BEGIN
VEC:='first';
FOR J:=1 TO 3 DO
BEGIN
CLRSRC:WRITELN:WRITELN('WHAT IS THE NAME OF THE ',VEC,' VECTOR ?');
IF STAT='L' THEN
  BEGIN
    IF J=1 THEN
      WRITELN('Containing the independent variable')
    ELSE
      WRITELN('Containing the dependent variable');
    END;
    READLN(C[J]);
  END;
{ ATTACH DISK-DRIVE TO NAME }
C[J]:=DRIVE+C[J];
{ LOCATE VECTOR }
ASSIGN VECTORFILE,C[J];
{ IF-J-RESET VECTORFILE}{$I+};
OK:=($result=0);
CLRSCR;
IF NOT OK THEN
  { PRINT ERROR MESSAGE }
BEGIN
  WRITELN('VECTOR ',C[J],' DOES NOT EXIST !');
  ANS:='Y';
  WHILE (ANS()='Y')AND(ANS()='N') DO
    BEGIN
      WRITELN('Would you like to try again to locate a vector ? (Y/N)');
      READLN(ANS);
      END;
      IF ANS='N' THEN GOTO 10;
  END;
UNTIL OK;
N[J]:=FILESIZE VECTORFILE;
{ CHECK LENGTH OF VECTOR }
IF N[J]<2 THEN
  BEGIN
    CLRSCR;WRITELN('The vector must contain at least 3 elements in order to calculate');
    WRITELN('the ',WHICH,',');
    WRITELN('Press RETURN to continue:');
    READLN(RES);
    GOTO 10;
  END;
{ READ VECTOR TO ARRAY }
FOR I:=1 TO N[J] DO
  BEGIN
    READ VECTORFILE,DATA);
    IF J=1 THEN
      V[I]:=DATA
    ELSE
      W[I]:=DATA;
  END;
{ CLOSE FILE }
CLOSE VECTORFILE;
VEC:="second";
END;
{ CHECK FOR EQUAL LENGTH VECTORS }
IF (N[I])<N(J) THEN
  { PRINT ERROR MESSAGE }
BEGIN
  CLRSCR;WRITELN('These two vectors are NOT the same length !');
  WRITELN('Press RETURN to continue:');
  READLN(RES);GOTO 10;
END;
{ CHECK FOR PERFECT LINEAR CORRELATION }
COUNT:=0;
FOR I:=1 TO N[J] DO
  IF V[I]=W[I] THEN COUNT:=COUNT+1;
  IF COUNT=N[J] THEN
    BEGIN
      CLRSCR;WRITELN('The two vectors are the same.');
      WRITELN('therefore, there exists a PERFECT linear correlation !');
      WRITELN('Press RETURN to continue:');
      READLN(RES);GOTO 10;
END;
{ CHECK FIRST VECTOR FOR STD. DEV. OF ZERO }
COUNT1:=0;
FOR I:=1 TO N[1]-1 DO
BEGIN
  IF V[I]=V[I+1] THEN
    COUNT1:=COUNT1+1;
END;
{ CHECK SECOND VECTOR FOR STD. DEV. OF ZERO }
COUNT2:=0;
FOR I:=1 TO N[2]-1 DO
BEGIN
  IF W[I]=W[I+1] THEN
    COUNT2:=COUNT2+1;
END;
IF (COUNT1=N[1]-1)OR(COUNT2=N[2]-1) THEN
{ PRINT ERROR MESSAGE }
BEGIN
  CLRSCR;WRITELN;WRITELN('One of your vectors has identical entries, (Standard Deviation=0)');
  WRITELN('Therefore, the ','Which,' cannot be calculated.');
  WRITELN;WRITELN('Press RETURN to continue:');
  READLN(RES);GOTO 10;
END;
{ CALCULATE SUMATIONS }
SX:=0;SY:=0;SX2:=0;SY2:=0;SXY:=0;
FOR I:=1 TO N[J] DO
BEGIN
  SXY:=SXY + V[I] * W[I];
  SX:=SX + V[I];
  SY:=SY + W[I];
  SX2:=SX2 + V[I] * V[I];
  SY2:=SY2 + W[I] * W[I];
END;
IF STAT='C' THEN
BEGIN
  { CORRELATION COEFFICIENT }
  SC:=SXY-SX*SY/N[J];
  SXX:=SX2-SX*SX/N[J];
  SYY:=SY2-SY*SY/N[J];
  SR2:=SC*SC/(SXX*SYY);
  CC:=SQRT(SR2);
  { CALCULATE T-VALUE }
  IF SR2=1 THEN SR2:=0.9;
  ST:=SQRT(SR2*(N[J]-2)/(1-SR2));
  { CALCULATE AVERAGES }
  AVGX:=SX/N[J];
  AVGY:=SY/N[J];
  { CALCULATE VARIANCES \& STD.DEVIATIONS }
  VARY:=(SX2-(SX*SX)/N[J])/(N[J]-1);
  VARY2:=(SY2-(SY*SY)/N[J])/(N[J]-1);
  STDX:=SQRT(VARY);
  STDY:=SQRT(VARY2);
  OUTPUT:='M'.
  WHILE (OUTPUT='Y')AND(OUTPUT='M') DO
BEGIN
  CLRSCR;WRITELN;WRITELN('Do you want a HARD COPY of the results ? (Y/M)');
  READLN(OUTPUT);
END;
IF OUTPUT='Y' THEN
BEGIN
  CLRSCR;WRITELN;WRITELN('Once the CALCULATION OF CORRELATION COEFFICIENT has been displayed o
n');
  WRITELN('the terminal screen, press the SHIFT KEY and the PRTSC KEY to get a hard copy');
  WRITELN('of what is on the screen.');
  WRITELN;WRITELN('Press RETURN to continue:');
  READLN(RES);
END;
{ PRINT OUTPUT ON VDT }
CLRSCR;WRITELN;WRITELN('Calculation of Correlation Coefficient');
WRITELN;WRITELN('Vector C[1]');
WRITELN;WRITELN('Number of Observations = ',N[J]);
WRITELN('Average = ',AVGX:13:4);
Writeln('Variance : ',VAR:13:4);
Writeln('Standard Deviation: ',STDV:13:4);
Writeln('Vector',C[2]);
Writeln('Number of Observations: ',N[2]);
Writeln('Average : ',AVG:13:4);
Writeln('Variance : ',VAR:13:4);
Writeln('Standard Deviation: ',STD:13:4);
Writeln('Correlation Coefficient: ',CC:13:4);

if output='Y' then readln(res);
Writeln(writeln('Press RETURN to continue with significance testing.'));
readln(res);
{Determine Degrees of Freedom }

if N[1]<2 then
{ Enter Alpha Value }

begin

var alpha:
entry := 'BAD';
while entry='BAD' do
{ Loop for Invalid Alpha Entry }

begin

check for return W/O entry

alpha := '':
while length(stalpha)<3 do

begin
clscr,writeln,writeln('Enter the desired ALPHA (0 .. ALPHA < 1) in decimal form for test.');
writeln(writeln('Examples: .5 is entered as .05');
writeln('10% is entered as .10'));
readln(stalpha);
end;
{ Check Validity of Entry }
check(stalpha.entry);
end;
{ Change Alpha From String To Real }
val(stalpha, alpha, code):
{ Choose Hypothesis Test }
resp := 'O':
while (resp='1') and (resp='2') do

begin
clscr,writeln,writeln('Would you like to see if the CORRELATION COEFFICIENT is :');
writeln(writeln('1 significantly different than 0 ?');
writeln('2 significantly different than some certain value?');
writeln(writeln('Enter the number of your selection and press RETURN.'));
readln(resp);
end;
{ Calculate Appropriate Output }
clscr,writeln;

case resp of
'1':begin
if output='Y' then
begin
clscr,writeln,writeln('Once the STATISTICAL HYPOTHESIS TEST has been displayed on the
terminal');

writeln('screen, press the SHIFT KEY and the PRTSC KEY to get a hard copy of what');
writeln('is on the screen.');
writeln('Press RETURN to continue.');
readln(res);
end;
clscr,writeln,writeln('STATISTICAL HYPOTHESIS TEST');
writeln(writeln('Ho: rho EQUAL 0 (independence)');
writeln('Hi: rho NOT EQUAL 0');
rho := 0.0;
if N[1]<=30 then
{ Use T-VALUE For SAMPLE (30 }
TVALUE(ADF,P,ST)
else
{ Use Z-VALUE For SAMPLE =>30 }
ZVALUE(P,ST);
end;
'2':begin
rho := -2;
while (rho<1) or (rho>1) do
{ Loop for Invalid Rho Entry }
begin
writeln('What value would you like to test the CORRELATION COEFFICIENT against ?');
writeln('Enter [-1] or [1] ');
readln(strho);
{ Change Rho From String To Real }
Once the STATISTICAL HYPOTHESIS TEST has been displayed on the screen, press the SHIFT KEY and the PRTSC KEY to get a hard copy of what is on the screen.

Press RETURN to continue:

READLN(RES);

END

END

END
BEGIN
  CLRSCR;WRITELN;WRITELN('Enter the desired ALPHA (0 < ALPHA < 1) in decimal form for testing:')
  WRITELN('Examples: 5% is entered as .05');
  WRITELN('10% is entered as .10');
  READLN(S1ALPHA);
END;

{ CHECK VALIDITY OF ENTRY }
CHECK(STALPHA,ENTRY);

END-

CHECK ALPHA FROM STRING TO REAL
   VAL(STALPHA,ALPHA,CODE);
   { DETERMINE DEGREES OF FREEDOM }
   ADF:=M[1]-2;
   OUTPUT:="M'';
   WHILE (OUTPUT=')"Y')AND(OUTPUT="N") DO BEGIN
   CLRSCR;WRITELN;WRITELN('Would you like a HARD COPY of the output ? (Y/N)');
   READLN(OUTPUT);
   END.
IF OUTPUT:='Y' THEN BEGIN
   CLRSCR;WRITELN;WRITELN('Once the CALCULATION OF LINEAR REGRESSION has been displayed on the' Terminal screen, press the SHIFT KEY and the PRTSC KEY to get a hard copy of what is on the screen.);
   WRITELN(RES);
END;

IF OUTPUT:='Y' THEN BEGIN
   CLRSCR;WRITELN;WRITELN('Once the STATISTICAL HYPOTHESIS TEST has been displayed on the' Terminal screen, press the SHIFT KEY and the PRTSC KEY to get a hard copy of what is on the screen.);
   WRITELN(RES);
END.

{ PRINT INTERCEPT TEST RESULTS }
CLRSCR;WRITELN;WRITELN('STATISTICAL HYPOTHESIS TEST for Intercept');
WRITELN('Ho: "a" (intercept) = 0');
WRITELN('H1: "a" not equal to 0');
WRITELN('Selected Alpha = ',ALPHA:1:2);
WRITELN('Calculated t-value = ',TA:3:4);
WRITELN('Degrees of freedom = ',ADF);
TVALUE(ADF,P,TA);
WRITELN('Probability of both tails = ',P:1:4);
WRITELN(RES);
IF P<ALPHA THEN
  WRITELN('Accept Ho; thus "a" is NOT significant')
ELSE
  WRITELN('Reject Ho; thus "a" is significant')
{ PRINT SLOPE TEST RESULTS }
CLRSCR;WRITELN;WRITELN('STATISTICAL HYPOTHESIS TEST for slope');
WRITELN('Ho: "b" (slope) = 0 (Two vectors are independent)');
WRITELN('H1: "b" not equal to 0');
WRITELN('Selected Alpha = ',ALPHA:1:2);
WRITELN('Calculated t-value = ',TB:3:4);
WRITELN('Degrees of freedom  = ', ADF);
TVVALUE(ADF,P,T8);
WRITELN('Probability of two tails  = ',P:1:4);WRITELN;
IF P<ALPHA THEN
  WRITELN('Accept Ho; thus "b" is NOT significant')
ELSE
  WRITELN('Reject Ho; thus "b" is significant');
READLN(RES);
END;
{ REPEAT THIS STATISTIC ? }
ANS:='Y';
WHILE (ANS('Y')AND(ANS('N'))) DO
BEGIN
  CLRSER;WRITELN;WRITELN('Would you like to calculate another ',' WHICH, '? (Y/N)');
  READLN(ANS);
END;
{RETURN TO MAIN MENU}
ID:ASSIGN(VSTAT,'VSTAT.COM');
EXECUTE(VSTAT);
END(CCLIN).
{ NONLINEAR REGRESSION }

PROGRAM NONLIN;
{ MAIN PROGRAM STORAGE SPECIFICATIONS }
LABEL 10,280,320,470,790;

TYPE
STRING2=STRING[2];
STRING3=STRING[3];
STRING4=STRING[4];
STRING13=STRING[13];
STRING60=STRING[60];
IRAY2=ARRAY[1..2] OF INTEGER;
RAY4=ARRAY[1..4] OF CHAR;
RAY7=ARRAY[1..7] OF REAL;
RAY5=ARRAY[1..5] OF REAL;
RAY2=ARRAY[1..2] OF STRING[8];
RAY45=ARRAY[1..4] OF ARRAY[1..5] OF REAL;
RANGE=ARRAY[1..1000] OF REAL;

VAR
DRIVE:STRING2;
VIEW:INTEGER;
V.W.RANGE;
VECTORFILE:FILE OF REAL;
VSTAT:FILE;
DF,NUM,JVEC,DF2,X,J,L,I,D,CODE:INTEGER;
LET,OUTPUT,ANS,RES:CHAR;
VALUE,ALPHA,S,I,D,T,INN,VA,I,J,DATA,KK,JJ,PW,ZZ,P,Q:REAL;
STALPHA:STRING3;
ENTRY:STRING4;
WHICH:STRING[10];
WHICHV:STRING[11];
OK:BOOLEAN;
N:IRAY2;
SIGN:RAY4;
A:RAY7;
AN,T:RAY5;
B,R:RAY45;
NON:RAY2;

MAIN PROGRAM - VARIABLE DEFINITIONS

DRIVE - Current data disk-drive.
V.W - Arrays containing the first and second vectors, respectively.
VECTORFILE - File containing vector of interest.
VSTAT - File containing the main menu.
DF,DF2,DF - Degrees of Freedom.
NUM - Number of elements in the vector.
JVEC,X,J,L - Array element subscripts.
D - Degree of the regression.
CODE - Conversion error detection variable.
LET - Regression constant.
OUTPUT - 'Y' for hard copy.
RES - User's response to regression type menu.
ANS - User's response to Y/N questions.
I,D,P - Used in calculation of sum of squares error.
T,INN,VA,I,D - Used in calculation of t-value.
Z - Probability of both tails.
DATA - Vector elements.
KK,JJ,PW - Used in raising a value to a power.
S,VALUE,ZZ - Used in calculation of regression equation.
A,B,T,AN - Arrays used in calculation of regression equation.
ALPHA - Alpha value.
STALPHA - String containing alpha value (to check validity).
ENTRY - Used to signal an invalid user entry.
WHICH - String containing 'Quadratic' or 'Cubic'.
WHICHV - String containing 'first' or 'second'.
OK - True if file exists.
N - Array containing the size of each vector.
SIGN - Characters '+' or '-' for regression equation.
NON - Array of two strings containing the vector names.

RAISE TO A POWER
PROCEDURE POWER(VAR X,Y,P:REAL);
{ PROCEDURE POWER STORAGE SPECIFICATIONS }
LABEL 5;
VAR
  NUM,XX,P:REAL;
  YY:INTEGER;
{ PROCEDURE POWER - VARIABLE DEFINITIONS }
P - Number raised to a power.
NUM,XX,YY - Used in calculation of raising to a power.
BEGIN
{ EXPONENT OF ZERO }
IF (Y=0) THEN
BEGIN
  POW:=1.0;
  GOTO 5;
END
{ BASE OF ZERO }
IF (X=0) THEN
BEGIN
  POW:=0.0;
  GOTO 5;
END ELSE
{ NEGATIVE BASE }
IF (X<0) THEN
  XX:=X;
{ POSITIVE BASE }
ELSE
  XX:=XX;
NUM:=Y*LN(XX);
P:=EXP(NUM);
{ CHECK SIGN }
IF (X<0) THEN
BEGIN
  YY:=TRUNC(Y);
  IF (ODD(YY)=TRUE) THEN
    POW:=0.0-P
  ELSE
    POW:=P;
END ELSE
  POW:=P;
S:
END(FUNC POWER);
{ CHECK FOR INVALID DATA ENTRIES }
PROCEDURE CHECK(VAR ST:STRING; VAR ERR:STRING);{ PROCEDURE CHECK STORAGE SPECIFICATIONS }
TYPE
  CHARACTERS=SET OF CHAR;
VAR
  VALID:CHARACTERS;
  POS,COUNT:INTEGER;
  POSITION:CHAR;
{ PROCEDURE CHECK - VARIABLE DEFINITIONS }
VALID - Set of valid characters.
POS - Position in a string.
COUNT - Counter for invalid elements.
POSITION - A single character of a string.
BEGIN
{ DEFINE VALID CHARACTERS }
VALID:={0',9',+-'.'}
{ INITIALIZE COUNTER }
COUNT:=0;
{ CHECK EACH POSITION IN THE STRING }
FOR POS:=1 TO LENGTH(ST) DO
BEGIN
POSITION:=COPY(ST,POS,1);
IF POSITION IN VALID THEN
COUNT:=COUNT+1;
END;
{ COMPARE # OF VALID POSITIONS TO LENGTH OF STRING }
IF COUNT(LNGTH(ST) THEN
{ PRINT ERROR MESSAGE }
BEGIN
ERR:="INVALID ENTRY! - You must enter a number for this value."
EXIT
END(VALID);
ELSE
ERR:="NONE"
END(CHECK);
{ T DISTRIBUTION }
PROCEDURE TDISR(VAR DF:INTEGER; VAR TV,T:REAL);
{ PROCEDURE TDISTR STORAGE SPECIFICATIONS }
VAR
T2,TV,A,YX,YZ,B,TV:REAL;
DF:INTEGER;
{ PROCEDURE TDISTR - VARIABLE DEFINITIONS }
{ DF: Degrees of freedom. }
{ T2,D,TV,A,YX,YZ,B,TV: Used in the calculation of the probability. }
BEGIN
TV:=T*T;
DF::DF;
YX::TV/DF;
B::1+YX;
{ FOR SMALL SAMPLE (LESS THAN 20) }
IF (DF<20) THEN BEGIN
BEGIN
YX::SORT(YX);
A::YX;
IF (DF=1) THEN A:=0;
DF::DF-2;
WHILE (DF<1) DO BEGIN
A::((DF-1)/(B*DF)*A+YX);
DF::DF-2;
END;
END;
IF (DF=0) THEN BEGIN
A::((ARCTAN(YX)+A/B)*0.63661977236)
ELSE A::A/SQRT(B);
IF (A<1) THEN
Z::0
ELSE Z::1-A;
END;
{ FOR LARGE SAMPLE (GREATER THAN OR = 20) }
IF DF=20 THEN BEGIN
BEGIN
{ ASYMPTOTIC SERIES FOR LARGE SAMPLE }
IF (YX)<(0.1E-5)) THEN
YX::LN(B);
A::DF-0.5;
B::48*48;
YX::A*YX;
YX::(((0.4*YX-3.3)*YX-244*YX-85.5)/(0.8*(YX*YX)+100+B*YX+3)/B+1)*SORT(YX);
YX::YX;
IF (YX<-4.17) THEN
VN::0
ELSE IF (YX>4.17) THEN
VN::1
ELSE BEGIN
Z::YX;

IF (YX(0) THEN Y2 := (-YX);
T2 := T2/1.4142142;
D := (((((0.430638E-4*T2+0.2765672E-3)*T2+0.1520143E-3)*T2+0.92705272E-2)*T2+0.42282012E-1)*T2+0.70
523078E-1)*T2+1.0);
D := D*D; D := D*D; D := D*D;
V2 := 0.5 - 0.5*D;
IF (YX(0) THEN V2 := 0.5 - V2;
IF (YX(0) THEN V2 := 0.5;
IF (YX(0) THEN V2 := 0.5 + V2;
Z := Z + V2;
IF (Z(0) THEN Z := 0;
END;
END( TodIST);
{ CHECK A L P H A
PROCEDURE ALPHACK(VAR STALPHA: STRING; VAR ENTRY: STRING);
PROCEDURE ALPHACK - STORAGE SPECIFICATIONS
TYPE
CHARACTERSET = SET OF CHAR;
VAR
COUNT: INTEGER;
POSITION: CHAR;
VALID: CHARACTERSET;
PROCEDURE ALPHACK - VARIABLE DEFINITIONS
COUNT - Counter for number of valid positions in entry.
POSITION - A single character of a string.
VALID - Valid set of characters.
BEGIN
{ INITIALIZE COUNTER }
COUNT := 0;
{ DEFINE VALID SET OF CHARs }
VALID := ['0' .. '9'];
{ CHECK FIRST POSITION OF ENTRY }
POSITION := COPY(STALPHA, 1, 1);
IF POSITION = ' ' THEN
ENTRY := 'BAD';
ELSE
BEGIN
{ CHECK REMAINING POSITIONS }
POSITION := COPY(STALPHA, 2, 1);
IF POSITION IN VALID THEN
COUNT := COUNT + 1;
IF POSITION = '0' THEN
VALID := ['1' .. '9'];
POSITION := COPY(STALPHA, 3, 1);
IF POSITION IN VALID THEN
COUNT := COUNT + 1;
IF COUNT < 2 THEN
ENTRY := 'BAD';
ELSE
ENTRY := 'GOOD';
END;
END(ALPHACK);
{ MAIN PROGRAM - NONLINEAR REGRESSION }
BEGIN
ANS := 'Y';
WHILE ANS = 'Y' DO { LOOP TO REPEAT STATISTIC IF DESIRED }
BEGIN
{ INITIALIZE ARRAYS }
FOR I := 1 TO 4 DO BEGIN
FOR J := 1 TO 5 DO BEGIN
R[I, J] := 0;
END;
END;
CLRSCR; WRITELN; WRITELN('NONLINEAR REGRESSION');
{ LOCATE THE VECTOR OF INTEREST }
WHICHV := 'INDEPENDENT';
FOR JVEC := 1 TO 2 DO BEGIN
  REPEAT
    NON[JVEC] := '
    WHILE (LENGTH(NON[JVEC])<1 OR LENGTH(NON[JVEC])>6) DO BEGIN
      WRITELN(WRITELN('What is the name of the vector containing the ', WHICHV, ' variable?'));
      IF JVEC = 2 THEN WRITELN('NOTE: This vector must be the SAME LENGTH as the first vector. ');
      READLN(NON[JVEC]);
    END;
    { ATTACH DISK-DRIVE TO NAME }
    NON[JVEC] := DRIVE+NON[JVEC];
    { DETERMINE IF THE VECTOR EXISTS }
    ASSIGN VECTORFILE,NON[JVEC];
    {$I-RESET VECTORFILE {$I+} $);
    OK := {$result := 0};
    CLRSCR;
    IF NOT OK THEN BEGIN
      WRITELN('VECTOR ', NON[JVEC], ' DOES NOT EXIST! ');
      ANS := 'M';
      WHILE (ANS('Y') AND (ANS('M')) DO BEGIN
        WRITELN(WRITELN('Would you like to try again to locate a vector? (Y/N) ');
        READLN(ANS);
      END;
      IF ANS='M' THEN GOTO 10;
    END;
    UNIL OK:
    { DETERMINE SIZE OF VECTOR }
    N[JVEC] := FILESIZE VECTORFILE;
    { READ VECTOR TO ARRAY }
    FOR I := 1 TO N[JVEC] DO BEGIN
      READ VECTORFILE DATA;
      IF JVEC = 1 THEN V[I] := DATA
      ELSE W[I] := DATA;
    END;
    { CLOSE FILE }
    CLOSE VECTORFILE;
    WHICHV := 'DEPENDENT';
  END;
  { CHECK FOR == LENGTH VECTORS }
  IF (N[1]=N[2]) THEN BEGIN
    CLRSCR;
    WRITELN(WRITELN('These two vectors are NOT the same length! ');
    WRITELN(WRITELN(WRITELN('Enter RETURN to continue: ');
    READLN(RES); EXIT;
  END;
  NUM := N[2];
  { SELECT TYPE OF REGRESSION }
  RES := '0';
  WHILE (RES('1') AND (RES('2')) DO BEGIN
    CLRSCR;
    WRITELN(WRITELN('Would you like to perform a, ');
    WRITELN(WRITELN(' 1 Quadratic Regression ( Y = a + bx + cx^2, ');
    WRITELN(WRITELN(' 2 Cubic Regression ( Y = a + bx + cx^2 + dx^3, ');
    WRITELN(WRITELN('Enter the number of your choice and RETURN-: ');
    READLN(RES);
  END;
  CASE RES OF
    '1': D := 2;
    '2': D := 3;
  END(OF CASE);
  { INITIALIZE MATRIX }
  FOR I := 1 TO (2*D+1) DO A[I] := 0;
  FOR K := 1 TO (D+1) DO
BEGIN
  R[k,D+2]:=0;
  T[k]:=0;
END;
T[d+2]:=0;
{ POPULATE MATRICES WITH SYSTEM OF EQUATIONS }
A[1]::NUM;
FOR i:=2 TO NUM DO
BEGIN
  FOR J:=2 TO (2*D+1) DO
  BEGIN
    JJ:=J-1;
    POWER(V[1],JJ,POW);
    END;
  FOR K:=1 TO (D+1) DO
  BEGIN
    KK:=K-1;
    POWER(V[1],KK,POW);
    R(k,D+2):=T[k]+W[I]*POW;
    T(k):=T(k)+W[I]*POW;
    END;
  T[d+2]:=T[D+I]+W[I]*POW;
END;
{ SOLVE SYSTEM OF EQUATIONS IN MATRICES }
FOR J:=1 TO (D+1) DO
BEGIN
  B[J,J]:=1;
  FOR K:=1 TO (D+1) DO
  R[J,K]:=A[K+J-1];
  END;
FOR J:=2 TO (D+2) DO
BEGIN
  S:=R[J,J];
  R[J,J]:=R[K,I];
  R[K,I]:=S;
  IF I<=(D+2) THEN
  BEGIN
    S:=B[J,I];
    B[J,I]:=B[K,I];
    B[K,I]:=S;
  END;
END;
ZZ:=1/R[J,J];
FOR I:=1 TO (D+2) DO
BEGIN
  R[J,1]:=ZZ*R[J,1];
  IF I<=(D+2) THEN
  BEGIN
    B[J,1]:=ZZ*B[J,1];
  END;
END;
FOR K:=1 TO (D+1) DO
BEGIN
  IF K=J THEN GOTO 470;
  ZZ:=R[K,J];
  FOR I:=1 TO (D+2) DO
  BEGIN
    R[K,1]:=R[K,1]+ZZ*R[J,1];
    IF I<=(D+2) THEN
    BEGIN
      B[K,1]:=B[K,1]+ZZ*B[J,1];
    END;
  END;
END;
GOTO 790;
320:
WRITEL('THERE IS NO WHICH, EQUATION FOR VECTOR ',NON[1],', AND VECTOR ',NON[2],', !');
GOTO 970;
184
ENTRY: 'BAD'
WHILE ENTRY='BAD' DO
BEGIN
STALPHA=''
WHILE LENGTH(STALPHA)<3 DO
BEGIN
CLRSCR;WRITELN;WRITELN('Enter the desired ALPHA for significance testing:');
WRITELN(' (For example, .05 would be entered as .05)');
READLN(STALPHA);
END;
{ CHECK FOR INVALID ALPHA ENTRY }
ALPHACK(STALPHA,ENTRY);
END;
{ CHANGE ALPHA FROM STRING TO REAL }
VAL(STALPHA,ALPHA,CODE);
{ SELECT OUTPUT DEVICE }
OUTPUT='N'
WHILE (OUTPUT='Y') AND (OUTPUT='N') DO
BEGIN
CLRSCR;WRITELN;WRITELN('Would you like a hard copy of the results? (Y/N)');
READLN(OUTPUT);
END;
IF OUTPUT='Y' THEN
BEGIN
CLRSCR;WRITELN;WRITELN('After the results have been printed on the terminal screen, press the');
WRITELN('SHIFT key and the PRTSC KEY to get a hard copy of what is on the screen. ');
WRITELN('Press RETURN to continue.' );
READLN(RES);
END;
IF D=2 THEN WHICH='Quadratic';
IF D=3 THEN WHICH='Cubic';
{ PRINT REGRESSION EQUATION }
CLRSCR;WRITELN('Calculation of ',WHICH,' regression');
WRITELN('VECTOR I',NON,' (Containing the independent variable)');
WRITELN('VECTOR Y',NON,' (Containing the dependent variable)');
WRITELN('Regression Equation:');
{ ADJUST SIGNS FOR EQUATION }
L:=1;
REPEAT
VALUE:=R[L,D+2];
IF ABS(VALUE)=VALUE THEN
SIGN[L]:='+'
ELSE
SIGN[L]:='-';
L:=L+1;
UNTIL L>(D+1); IF D=2 THEN WRITELN('Y = ',SIGN[1],R[1,D+2]:13:4,SIGN[2],R[2,D+2]:13:4,'X',SIGN[3],R[3,D+2]:13:4,'X^2');
{ CALCULATE Sum of SQUARES ERROR }
AN[1]:=R[1,D+2];
FOR J:=1 TO D DO
AN[J+1]:=R[J+1,D+2];
P:=0;
FOR J:=2 TO (D+1) DO
P:=P+R[J,D+2]*T[J]*T[J]/NUM;
Q:=T[D+2]*T[D+2]/NUM;
Z:=Q-P;
I:=NUM-D-1;
AN[D+2]:=SORT(P/Q);
AN[D+3]:=Z;
WHEN:=-D[D+1,D+1];
IF D=2 THEN LET:='c';
IF D=3 THEN LET:='g';
{ STATISTICAL HYPOTHESIS TESTING }
WRITELN;WRITELN('Sum of Square Error = ',AN[D+3]:8:4);
WRITELN;WRITELN(' ',STATISTICAL HYPOTHESIS TEST');
WRITELN;WRITELN('Ho: ',WHICH,' constant (',LET,') = 0');
WRITELN;WRITELN('H1: ',WHICH,' constant (',LET,') is not equal to 0');
WRITELN;WRITELN('Selected alpha = ',ALPHA:8:4);
{ CALCULATE T-VALUE }
DF:=N[1]-(D+1);
INN:=ABS(INN);INN:=SQRT(INN);
ZZ:=SQRT((2/DF));
IF ZZ<0 THEN
BEGIN
  ID:=ABS((AM-|D+1|));
  TI:=10/(ZZ*INN);
  VA:=TI;
  { CALL PROCEDURE TO CALCULATE PROBABILITY }
  TDISR(DF,TI,Z);
END;

{ PRINT RESULTS OF HYPOTHESIS TEST }
WRITELN('Calculated t-value = ',VA:8:4);
WRITELN('Degrees of freedom = ',DF:4);
WRITELN('Probability of both tails = ',ZZ:8:4);
WRITELN;
IF ALPHA<ZZ THEN
  WRITELN('Accept Ho: The ',WHICH,' constant is NOT significantly different than 0');
ELSE
  WRITELN('Reject Ho: The ',WHICH,' constant is significantly different than 0');
IF OUTPUT='Y' THEN READLN(RES);
WRITELN('RETURN to continue:');
READLN(RES);
790:
{ REPEAT THIS STATISTIC ? }
ANS:=';
WHILE (ANS)='Y' AND (ANS)='N' DO
BEGIN
  CLEAR;WRITELN;WRITELN('Would you like to perform another NONLINEAR REGRESSION ? (Y/N)');
  READLN(ANS);
END;
PROGRAM FREHIST;
{ MAIN PROGRAM STORAGE SPECIFICATIONS }
LABEL IS;

TYPE
STRING2=STRING[2];
CHARACTERS=SET OF CHAR;
STRING4=STRING[4];
STRING[13]=STRING[13];
RANGE=ARRAY[1..1000] OF REAL;
RAYJ0=ARRAY[1..30] OF REAL;
RAYJ0=ARRAY[1..30] OF INTEGER;
STRING3=STRING[3];
RAYJ0=ARRAY[1..20] OF STRING3;
RAYJ0=ARRAY[1..20] OF CHAR;

VAR
DRIVE:STRING2;
VIEW:INTEGER;
STAT:CHAR;
REP,OUTPUT,RES,ANS,HISTO:CHAR;
VSTAT:FILE;
VECTORFILE:FILE OF REAL;
MAXFREQ,COUNT,NCELLJ,I,J,K,L,CODE:INTEGER;
YMIN,YMAX,AXV,SSTD,DATA,NCELL:REAL;
OK:BOOLEAN;
ENTRY:STRING[4];
FREQ:STRING[8];
STCELL:STRING13;
LAB,WHICH:STRING[15];
IN1,IN2,IN3,IN4,IN5,IN6,IN7,IN8,IN9,IN10,IN11,IN12,IN13,IN14,IN15:CRAY;
IN16,IN17,IN18,IN19,IN20,IN21,IN22,IN23,IN24,IN25,IN26,IN27,IN28,IN29,IN30:CRAY;
F:CRAY20;
V:RANGE;
MIDPT,FRQ,FRRQ,OUT1,OUT2:RAY30;
CFR,FRRQ:RAY30;
VALIDI,VALID2:CHARACTERS;

MAIN PROGRAM - VARIABLE DEFINITIONS

DRIVE - Current data disk-drive.
STAT - '1' for freq. table, '2' for histogram.
REP - Used as counter for hard copy of result.
VSTAT - File containing main menu.
VECTORFILE - File containing vector of interest.
OUTPUT - 'Y' if hard copy desired by user.
POSITION - A single character in a string.
RES - User's response to menu selection.
ANS - User's response to Y/N questions.
POS - Position in a string.
HISTO - Type of histogram desired.
MAXFREQ - Max. freq. of the observations of a vector.
NCCELL2 - Number of cells (integer).
L,I,K - Vector array subscripts.
CODE - Conversion error signal.
YMIN - Minimum value of the vector.
YMAX - Maximum value of the vector.
AXV - Average of vector values.
SSTD - Standard Deviation of values in vector.
DATA - Elements in the vector.
FRRQ - Frequency of a value.
CFR - Cumulate frequency.
RFHRQ - Relative frequency.
RFRQ - Relative cumulative frequency.
MIDPT - Array of cell midpoints.
LAB - Label for histogram y-axis.
{IN1..IN30 - Arrays to store the "x" character for histo.

CHECK FOR INVALID ENTRIES
PROCEDURE CHECK(VAR ST:STRING[13]; VAR ENTRY:STRING[4]);

TYPE
CHARACTERSET=SET OF CHAR;
VAR
VALID1,VALID2:CHARACTERSET;
COUNT,POS:INTEGER;
POSITION:CHAR;
{ PROCEDURE CHECK - VARIABLE DEFINITIONS }
VALID1,VALID2 - Valid sets of characters.
POS - The position in a string.
COUNT - Count number of valid characters in a string.
POSITION - A single character of a string.
BEGIN
{ INITIALIZE COUNTER }
COUNT:=0;
{ DEFINE VALID DATA SETS }
VALID1:=['0'..'9',',','.'];
VALID2:=['0'..'9'];
{ CHECK FIRST POSITION OF ENTRY }
POSITION:=COPY(ST,1,1);
IF POSITION IN VALID1 THEN
COUNT:=COUNT+1;
{ CHECK ALL OTHER POSITIONS }
FOR POS:=2 TO LENGTH(ST) DO
BEGIN
POSITION:=COPY(ST,POS,1);
IF POSITION IN VALID2 THEN
COUNT:=COUNT+1;
{ REDEFINE VALID SET WHEN DECIMAL POINT ENCOUNTERED }
IF POSITION='.' THEN
VALID2:=['0'..'9'];
END;
{ COMPARE # VALID POSITIONS TO LENGTH OF STRING }
IF COUNT=LENGTH(ST) THEN
ENTRY:='GOOD'
ELSE
ENTRY:='BAD';
END(CHECK);

{ CALCULATE MEAN & STANDARD DEVIATION }
PROCEDURE MEANSTD(VAR K:INTEGER; VAR YMIN,YMAX,AXV,SXTD:REAL);
VAR
SX1,SX2,VXR:REAL;
L:INTEGER;
{ PROCEDURE MEANSTR - VARIABLE DEFINITIONS }
SX1 - Sum of all elements in the data vector.
SX2 - Sum of the square of all elements in the data vector.
AXV - Average of the elements in the vector.
SXTD - Standard Deviation of the elements in the vector.
VXR - Variance of the elements in the vector.
L - Counter for the array containing the vector.
K - Length of the data vector.
YMIN - Minimum value of the vector.
YMAX - Maximum value of the vector.
BEGIN
{ INITIALIZE VARIABLES }
SX1:=0;SX2:=0;
{ DETERMINE MIN AND MAX }
YMIN:=V[I];YMAX:=V[I];
FOR L:=1 TO K DO
BEGIN
SX1:=SX1+V[L];
SX2:=SX2+V[L]*V[L];
IF V[L]<YMIN THEN YMIN:=V[L];
IF V[L]>YMAX THEN YMAX:=V[L];
END;
{ CALCULATE MEAN }
AXV:=SX1/K;
{ CALCULATE VARIANCE }
VXR:=(SXX-AV*X2)/(K-1);
{ CALCULATE STANDARD DEVIATION }
SSTD:=SQRT(VXR);
END(MEANSTD);
{ CALCULATE FREQUENCIES }
PROCEDURE CALCFREQ(VAR K,NCELL2:INTEGER; VAR OK:BOOLEAN; VAR FRQ:RAY30; VAR OUT1,OUT2:RAY30; VAR YMIN,YMAX,AX,AXV,SSTD:REAL); VAR CODE,L,J,FH:INTEGER;
TSTP,YLOW,STP,YHI,A1,X1:REAL;
AGAIN:CHAR;
STYLOW,STSTOP:STRING[13];
PROCEDURE CALCFREQ - VARIABLE DEFINITION
L - Counter for array containing data vector.
J - Counter for array containing frequencies.
FM - Number of cell for table (integer form).
TSTP - Calculated minimum cell width required for table.
STP - Desired cell width entered by user.
YLOW - First cell value for table.
YHI - Last cell value for table.
AI - Sum of first cell value and cell width.
X1 - Used to calculate frequencies.
AGAIN - User's response to Y/N question.
ST - Data entered by user in string form to check validity.
BEGIN
{ INITIALIZE ARRAYS }
FOR L:=1 TO 30 DO BEGIN
FRQ[L]:=0;OUT1[L]:=0;OUT2[L]:=0;
END;
{ CALL PROCEDURE TO CALCULATE MEAN AND STD. DEV. }
MEANSTD(K,YMIN,YMAX,AXV,SSTD);
{ DETERMINE MIN CELL WIDTH }
TSTP:=(YMAX-YMIN)/NCELL2;
{ LOOP FOR RE-ENTERING CELL WIDTH }
AGAIN:='Y';
WHILE AGAIN:='Y' DO BEGIN
BEGIN
STP:=0;
WHILE (STP<0) DO BEGIN
ENTRY:='BAD';
WHILE ENTRY='BAD' DO BEGIN
BEGIN
{ OUTPUT CURRENT CELL INFORMATION }
CLRSCR;WRITELN('The number of cells required is ',NCELL2:4);
WRITELN('The minimum value is ',YMIN:13:4);
WRITELN('The maximum value is ',YMAX:13:4);
{ PROMPT USER WITH MINIMUM CELL WIDTH }
WRITELN('Your cell width should be greater than ',TSTP:13:4);
{ PROMPT USER TO ENTER CELL WIDTH }
WRITELN('Enter the width you want for each cell: ');
READLN(STSTOP);
{ CHECK FOR INVALID ENTRY }
CHECK(STSTOP,ENTRY);
END;
{ CHANGE ENTRY FROM STRING TO REAL }
VAL(STSTOP,STP,CODE);
END;
{ DETERMINE FIRST CELL VALUE (YLOW) }
YLOW:=YMIN+1;
WHILE (YLOW)<YMIN) DO BEGIN
ENTRY:='BAD';
WHILE ENTRY='BAD' DO BEGIN
END;
}
{ PROMPT USER FOR FIRST CELL VALUE }
CLRSCR;WRITELN;WRITELN('Enter the first cell value to start:');
READLN(STYLOW);
{ CALL PROCEDURE TO CHECK FOR INVALID ENTRY }
CHECK(STYLOW,ENTRY);
END;
{ CHANGE FROM STRING TO REAL }
VAL(STYLOW,YLOW,CODE);
{ COMPARE VECTOR MIN W/ FIRST CELL VALUE }
IF (YLOW YMIM) THEN
BEGIN
{ PRINT ERROR MESSAGE. }
WRITELN('Your lower limit is greater than the minimum value of your vector, ',YMIM:8:4,' !');
WRITELN('Press any key, then TRY AGAIN !');
READLN(RES);
END;
END:
{ CHECK TO INSURE ALL DATA WILL FIT }
YHI:=YLOW+(STP*NCELL2);
{ PRINT WARNING - NOT ALL DATA WILL FIT }
IF YHI( YMAX THEN
BEGIN
AGAIN:='M';
WHILE (AGAIN('Y')AND(AGAIN('N')) DO
BEGIN
{ PROMPT USER WITH OPTION OF RE-ENTERING CELL WIDTH & LOWER LIMIT }
CLRSCR;WRITELN;WRITELN('With the width and lower limit you just entered,');
WRITELN('some of your data will not be represented.');
WRITELN('Do you want to re-enter the width and lower limit ? (Y/N)');
READLN(AGAIN);
END;
END
ELSE
AGAIN:='N';
END;
{ CALCULATE THE INTERVAL WIDTH (SCALE) }
FN:=NCELL2;
AI:=YLOW+STP;
{ CALCULATE THE FREQUENCY }
FOR L:=1 TO K DO
BEGIN
X1:=0;
X1:=V[L]-AI-0.001;
IF (X1(0) THEN
J:=1
ELSE
J:=TRUNC(X1/STP)+2;
FRRQ[J]:=FRRQ[J]+1;
END;
{ DETERMINE TABLE ENTRIES }
J:=1;OUT1[J]:=YLOW;
FOR L:=1 TO NCELL2 DO
BEGIN
OUT2[L]:=OUT1[L]+STP;
J:=J1;
IF (J<30) THEN
OUT1[J]:=OUT2[J-1];
END;
END(CALCFREQ);
{ MAIN PROGRAM }
BEGIN
ANS:='Y';
WHILE (ANS='Y') DO { LOOP FOR REPEATING STATISTIC IF DESIRED }
BEGIN
CLRSCR;
IF STAT='1' THEN WHICH:='FREQUENCY TABLE';
IF STAT='2' THEN WHICH:='HISTOGRAM';
REPEAT
{ CHECK FOR RETURN W/O ENTRY }
VFREQ:='';
WHILE (LENGTH(VFREQ)<1) OR (LENGTH(VFREQ)>6) DO
BEGIN
{ PROMPT USER FOR NAME OF VECTOR OF INTEREST }
WRITELN('What is the name of the vector to be displayed in a ',WHICH,' ?');WRITELN;
READLN(VFREQ);
END;
{ ATTACH DISK-DRIVE TO NAME }
VFREQ:=DRIVE+VFREQ;
{ CHECK IF FILE EXISTS }
ASSIGN(VECTORFILE,VFREQ);
$i:=RESET(VECTORFILE);$i*;
OK:=$i=0;
CLSCHR;
IF NOT OK THEN
{ PRINT ERROR MESSAGE }
BEGIN
WRITELN('VECTOR ','VFREQ,' DOES NOT EXIST !');
ANS:='M';
WHILE (ANS='Y') AND (ANS='N') DO
BEGIN
WRITELN;WRITELN;WRITELN('Would you like to try again to locate a vector ? (Y/N)');
READLN(ANS);
IF ANS='N' THEN GOTO 15;
END;
IF OK THEN
{ DETERMINE SIZE OF FILE }
K:=FILESIZE(VECTORFILE);
{ READ VECTOR TO ARRAY }
FOR I:=1 TO K DO
BEGIN
READ(VECTORFILE,DATA);
V[I]:=DATA;
END;
{ CLOSE FILE }
CLOSE(VECTORFILE);
{ CHECK LENGTH OF VECTOR }
IF K:=1 THEN
BEGIN
{ PRINT ERROR MESSAGE }
CLSCHR;WRITELN;WRITELN;
WRITELN;WRITELN;WRITELN('There is not enough data in this vector to produce a ',WHICH,'.');
WRITELN;WRITELN;WRITELN('Press any key to return to the MAIN MENU:');
READLN(RES);GOTO 15;
END;
{ DETERMINE TYPE OF HISTOGRAM TO PRINT }
IF STAT:='2' THEN
BEGIN
HISTO:=0;
WHILE (HISTO='1') AND (HISTO='2') AND (HISTO='3') AND (HISTO='4') DO
BEGIN
CLSCHR;WRITELN;WRITELN;WRITELN;
WRITELN('What type of histogram would you like to display ?');WRITELN;
WRITELN( ' 1 Frequency');
WRITELN( ' 2 Cumulative Frequency');
WRITELN( ' 3 Relative Frequency');
WRITELN( ' 4 Cumulative Relative Frequency');WRITELN;
WRITELN('Enter the number of your selection and press RETURN:');
READLN(HISTO);
END;
END;
{ DEFINE LABEL FOR HISTO Y-AXIS }
CASE HISTO OF
'1': LAB:='Freq.,'
'2': LAB:='Cum. Freq.,'
'3': LAB:='Rel. Freq.,'
'4': LAB:='Cum. Rel. Freq.,'
ENDCASE;
{ PROVIDE OPTION OF HAVING # CELLS CALCULATED OR ENTERED BY USER }
ANS:='M';
WHILE (ANS='Y') AND (ANS='N') DO
BEGIN
CLSCHR;WRITELN;WRITELN('Do you want the program to calculate the number of cells required ? (Y/N)
READLN(ANS);
IF ANS='N' THEN
{ USER ENTERS # OF CELLS }
BEGIN
NCELL2:=0;
WHILE (NCCELL2+3)OR(NCELL>30) DO
BEGIN
ENTRY:='BAD';
WHILE ENTRY='BAD' DO
BEGIN
{ PROMPT USER FOR # OF CELLS }
WRITELN,'Enter the number of cells (max=30, min=3):';
READLN(STCELL);
{ CALL PROCEDURE TO CHECK FOR INVALID ENTRY }
CHECK(STCELL,ENTRY);
END;
{ CHANGE ENTRY FROM STRING TO REAL }
VAL(STCELL,NCELL,CODE);
NCCELL:=ABS(NCELL);
{ CHANGE FROM REAL TO INTEGER }
NCCELL2:=TRUNC(NCELL);
END;
END ELSE
IF ANS='Y' THEN
{ CALCULATE THE NUMBER OF CELLS }
BEGIN
NCCELL:=SORT(X)+0.5;
NCCELL2:=TRUNC(NCELL);
IF NCELL2>30 THEN NCELL2:=30;
END;
{ CALL PROCEDURE TO CALCULATE FREQUENCY }
CALCFREQ(K,NCCELL2,0,FRQQ,OUT1,OUT2,YMIN,YMAX,AXV,SXTD);
IF NOT OK THEN EXIT;
{ INITIALIZE ARRAYS }
FOR L:=1 TO 30 DO
BEGIN
CFR[L]:=O;RFR[L]:=O;RCFR[L]:=O;HIDPT[L]:=O;
END;
{ CALCULATE INTERVAL MIDPOINT VALUES FOR HISTOGRAM }
IF STAT:='2' THEN
BEGIN
FOR L:=1 TO NCELL2 DO
BEGIN
MIDPT[L]::((OUT2[L]*OUT1[L])/2);
END;
END;
{ CALCULATE CUMULATIVE FREQUENCIES }
CRR[L]:=FRQQ[L];
FOR L:=2 TO NCELL2 DO
BEGIN
CRR[L]:=CRR[L-1]+FRQQ[L];
{ CALCULATE RELATIVE FREQUENCIES }
FOR L:=1 TO NCELL2 DO
BEGIN
RFR[L]:=FRQQ[L]/K;
{ CALCULATE CUMULATIVE RELATIVE FREQUENCIES }
RCR[L]:=RFR[L];
END;
{ SELECT OUTPUT DEVICE }
OUTPUT:='M';
WHILE (OUTPUT='Y')AND(OUTPUT='N') DO
BEGIN
CLRSCR;WRITELN;WRITELN('Do you want a HARD COPY of the ',WHICH, '? (Y/N)');
READLN(OUTPUT);
END;
{ OUTPUT FOR FREQUENCY TABLE }
IF STAT:='1' THEN
BEGIN
IF OUTPUT='N' THEN
BEGIN
CLRSCR;WRITELN('FREQUENCY TABLE FOR VECTOR ',VFREQ);WRITELN;
END;
IF OUTPUT='Y' THEN
BEGIN
CLRSCR;WRITELN(LST);
WRITELN(LST,'FREQUENCY TABLE FOR VECTOR ',VFREQ);
END;
FOR L:=1 TO NCELL2 DO
BEGIN
IF OUTPUT='N' THEN
BEGIN
WRITE(N(L:4,'
'),OUT1[L]:10:4,'-',OUT2[L]:10:4,'
',FRRQ[L]:5,,'CFR[L]:4',,'RFR[L]:8:4',,'RCFR[L]:8:4
');WRITELN;
ELSE
WRITELN(LST,L:4,'
'),OUT1[L]:10:4,'-',OUT2[L]:10:4,'
',FRRQ[L]:5,,'CFR[L]:4',,'RFR[L]:8:4',,'RCFR[L]:8:4
');WRITELN;
END;
END;
IF OUTPUT='N' THEN
BEGIN
BEGIN
of Observations :
AXV:13:4)
SXTD:13:4
YHIN:13:4
YHAX:13:4
RETURN to continue:
END;
ELSE
BEGIN
of Observations :
AXV:13:4)
SXTD:13:4
YHIN:13:4
YHAX:13:4
END;
READLN;
{ HISTOGRAM CALCULATIONS }
IF STAT='2' THEN
BEGIN
{ CALCULATE MAX FREQUENCY }
MAXFREQ:=FRRQ[1];
FOR L:=2 TO NCELL2 DO
IF FRRQ[L]>MAXFREQ THEN MAXFREQ:=FRRQ[L];
{ BLANK OUT HISTOGRAM }
FOR J:=1 TO 20 DO
BEGIN
IN1[J]:='\n',IN2[J]:='\n',IN3[J]:='\n',IN4[J]:='\n',IN5[J]:='\n',IN6[J]:='\n',IN7[J]:='\n',IN8[J]:='\n',IN9[J]:='\n',IN10[J]:='\n',IN11[J]:='\n',IN12[J]:='\n',IN13[J]:='\n',IN14[J]:='\n',IN15[J]:='\n',IN16[J]:='\n',IN17[J]:='\n',IN18[J]:='\n',IN19[J]:='\n',IN20[J]:='\n',IN21[J]:='\n',IN22[J]:='\n',IN23[J]:='\n',IN24[J]:='\n',IN25[J]:='\n',IN26[J]:='\n',IN27[J]:='\n',IN28[J]:='\n',IN29[J]:='\n',IN30[J]:='\n';END;
{ FREQ. OR CUM. FREQ. HISTOGRAM }
IF (HISTO='1')OR(HISTO='2') THEN
BEGIN
{ DETERMINE MAX FREQ. }
IF HISTO='2' THEN
BEGIN
HISTO:=K;
MAXFREQ:=K;
FOR L:=1 TO NCELL2 DO
FRRQ[L]:=CFR[L];
{ BLANK OUT Y-AXIS LABEL ARRAY }
FOR I:=1 TO 20 DO

F[1] := '  
{ DETERMINE Y-AXIS LABELS FOR MAX FREQ < 20 }
IF MAXFREQ<20 THEN
BEGIN
END;
{ DETERMINE Y-AXIS LABELS FOR MAX FREQ BETWEEN 20 AND 50 }
IF (MAXFREQ>=20) AND (MAXFREQ<50) THEN
BEGIN
  { SCALE OUTPUT TO FIT }
  FOR L:=1 TO NCELL2 DO
  BEGIN
    IF (FRRQ[L] < 2.5) THEN
      FRRQ[L] := '1';
    IF FRRQ[L] > 2.5 THEN
      FRRQ[L] := TRUNC(FRRQ[L]/2.5);
  END;
END;
{ DETERMINE Y-AXIS LABELS FOR MAX FREQ BETWEEN 50 AND 100 }
IF (MAXFREQ>=50) AND (MAXFREQ<100) THEN
BEGIN
  { SCALE OUTPUT TO FIT }
  FOR L:=1 TO NCELL2 DO
  BEGIN
    IF (FRRQ[L] < 25) THEN
      FRRQ[L] := '1';
    IF FRRQ[L] > 25 THEN
      FRRQ[L] := TRUNC(FRRQ[L]/25);
  END;
END;
{ DETERMINE Y-AXIS LABELS FOR MAX FREQ BETWEEN 100 AND 500 }
IF (MAXFREQ>=100) AND (MAXFREQ<500) THEN
BEGIN
  { SCALE OUTPUT TO FIT }
  FOR L:=1 TO NCELL2 DO
  BEGIN
    IF (FRRQ[L] < 50) THEN
      FRRQ[L] := '1';
    IF FRRQ[L] > 50 THEN
      FRRQ[L] := TRUNC(FRRQ[L]/50);
  END;
END;
{ DETERMINE Y-AXIS LABELS FOR MAX FREQ BETWEEN 500 AND 1000 }
IF (MAXFREQ>=500) AND (MAXFREQ<1000) THEN
BEGIN
  { SCALE OUTPUT TO FIT }
  FOR L:=1 TO NCELL2 DO
  BEGIN
    IF (FRRQ[L] < 100) THEN
      FRRQ[L] := '1';
    IF FRRQ[L] > 100 THEN
      FRRQ[L] := TRUNC(FRRQ[L]/100);
  END;
END;
END;
END;
END;
END;
BEGIN
{ CALCULATIONS FOR REL. FREQ. AND CU. REL. FREQ. HISTOS }
IF (HISTO='1') OR (HISTO='2') THEN
BEGIN
  { DETERMINE Y-AXIS LABELS }
  { SCALE FREQUENCIES TO FIT }
  FOR L:=1 TO NCELL2 DO
  BEGIN
    { SCALE REL. FREQUENCIES }
    IF HISTO='1' THEN
      BEGIN
        REF[L] := (RFR[L]*20)+0.5;
        FRRQ[L] := TRUNC(RFRQ[L]);
      END;
    IF HISTO='2' THEN
      BEGIN
        REF[L] := (RFR[L]*20)+0.5;
        FRRQ[L] := TRUNC(RFRQ[L]);
      END;
  END;
END;
END;
END;
END;
END;
END;
END.
}
IF FRRQ[L] < 1 THEN
  IF FRRQ[L] = 0 THEN FRRQ[L] := 1;
END;
{ SCALE CUM REL. FREQUENCIES }
IF HISTO = '4' THEN
BEGIN
  RCFR[L] := (RCFR[L] * 10) + 0.5;
  FRRQ[L] := TRUNC(RCFR[L]);
  IF FRRQ[L] < 1 THEN
    IF FRRQ[L] = 0 THEN FRRQ[L] := 1;
END;
END;
END{H~s~.~};
SET UP RAYS FOR HISTO
IF FRRQ[1] < 0 THEN
  FOR J := 1 TO FRRQ[1] DO
    IN1(J) := '*';
 IF FRRQ[2] < 0 THEN
  FOR J := 1 TO FRRQ[2] DO
    IN2(J) := '*';
 IF FRRQ[3] < 0 THEN
  FOR J := 1 TO FRRQ[3] DO
    IN3(J) := '*';
 IF FRRQ[4] < 0 THEN
  FOR J := 1 TO FRRQ[4] DO
    IN4(J) := '*';
 IF FRRQ[5] < 0 THEN
  FOR J := 1 TO FRRQ[5] DO
    IN5(J) := '*';
 IF FRRQ[6] < 0 THEN
  FOR J := 1 TO FRRQ[6] DO
    IN6(J) := '*';
 IF FRRQ[7] < 0 THEN
  FOR J := 1 TO FRRQ[7] DO
    IN7(J) := '*';
 IF FRRQ[8] < 0 THEN
  FOR J := 1 TO FRRQ[8] DO
    IN8(J) := '*';
 IF FRRQ[9] < 0 THEN
  FOR J := 1 TO FRRQ[9] DO
    IN9(J) := '*';
 IF FRRQ[10] < 0 THEN
  FOR J := 1 TO FRRQ[10] DO
    IN10(J) := '*';
 IF FRRQ[11] < 0 THEN
    IN11(J) := '*';
 IF FRRQ[12] < 0 THEN
  FOR J := 1 TO FRRQ[12] DO
    IN12(J) := '*';
 IF FRRQ[13] < 0 THEN
  FOR J := 1 TO FRRQ[13] DO
    IN13(J) := '*';
 IF FRRQ[14] < 0 THEN
  FOR J := 1 TO FRRQ[14] DO
    IN14(J) := '*';
 IF FRRQ[15] < 0 THEN
  FOR J := 1 TO FRRQ[15] DO
    IN15(J) := '*';
 IF FRRQ[16] < 0 THEN
  FOR J := 1 TO FRRQ[16] DO
    IN16(J) := '*';
 IF FRRQ[17] < 0 THEN
  FOR J := 1 TO FRRQ[17] DO
    IN17(J) := '*';
 IF FRRQ[18] < 0 THEN
  FOR J := 1 TO FRRQ[18] DO
    IN18(J) := '*';
 IF FRRQ[19] < 0 THEN
  FOR J := 1 TO FRRQ[19] DO
    IN19(J) := '*';
 IF FRRQ[20] < 0 THEN
Once the histogram has been displayed on the terminal screen, press the SHIFT key and the PRTSC key to obtain a hard copy of the histogram. After printing the histogram, press RETURN to display a table of midpoint values. Press the SHIFT key and the PRTSC key to list it on the printer.

```
FOR J:=1 TO FRQ[20] DO
  IM20[J] := '*';
IF FRQ[21] THEN
  FOR J:=1 TO FRQ[21] DO
    IM21[J] := '*';
IF FRQ[22] THEN
  FOR J:=1 TO FRQ[22] DO
    IM22[J] := '*';
IF FRQ[23] THEN
  FOR J:=1 TO FRQ[23] DO
    IM23[J] := '*';
IF FRQ[24] THEN
  FOR J:=1 TO FRQ[24] DO
    IM24[J] := '*';
IF FRQ[25] THEN
  FOR J:=1 TO FRQ[25] DO
    IM25[J] := '*';
IF FRQ[26] THEN
  FOR J:=1 TO FRQ[26] DO
    IM26[J] := '*';
IF FRQ[27] THEN
  FOR J:=1 TO FRQ[27] DO
    IM27[J] := '*';
IF FRQ[28] THEN
  FOR J:=1 TO FRQ[28] DO
    IM28[J] := '*';
IF FRQ[29] THEN
  FOR J:=1 TO FRQ[29] DO
    IM29[J] := '*';
IF FRQ[30] THEN
  FOR J:=1 TO FRQ[30] DO
    IM30[J] := '*';
{ PRINT HISTOGRAM }
IF OUTPUT = 'Y' THEN
BEGIN
  { PROMPT USER WITH INSTRUCTIONS TO OBTAIN PRINT-OUT OF HISTOGRAM }
  CLRSCR;WRITELN;'Once the histogram has been displayed on the terminal screen, press the SHIFT key and the PRTSC key to obtain a hard copy of the histogram.');WRITELN;
  WRITELN('After printing the histogram, press RETURN to display a table of midpoint values. Press the SHIFT key and the PRTSC key to list it on the printer.');WRITELN;
  WRITELN('Press RETURN to continue:');READLN(RES);
END;
{ INITIALIZE COUNTERS }
REP:= '0';COUN*: ':1;
CLRSCR;WRITELN;WRITELN;WRITELN('Calculating ,LAB, Histogram for vector ',WFREQ,'...');WRITELN;WRITELN;WRITELN('Press RETURN to continue:');READLN(RES);
{ DISPLAY HISTOGRAM }
CLRSCR;
J:=20;
REPEAT
  { OUTPUT FOR BETWEEN 20 AND 30 CELLS }
  IF (NCCELL>=20) AND (NCCELL<=30) THEN
  BEGIN
    WRITE(INI[J],',',IN1[J],IN2[J],IN3[J],IN4[J],IN5[J],IN6[J],IN7[J],IN8[J],IN9[J],IN10[J],IN11[J],IN12[J],IN13[J],IN14[J],IN15[J],IN16[J],IN17[J],IN18[J],IN19[J],IN20[J],IN21[J],IN22[J],IN23[J],IN24[J],IN25[J],IN26[J],IN27[J],IN28[J],IN29[J],IN30[J],IN30[J]);
  END;
```
{ OUTPUT FOR BETWEEN 10 AND 20 CELLS }
IF ((NCOLL2) AND (NCOLL2 < 20)) THEN
BEGIN
WRITE(F[1]); 
WRITE(F[2]); 
WRITE(F[3]); 
WRITE(F[4]); 
WRITE(F[5]); 
WRITE(F[6]); 
WRITE(F[7]); 
WRITE(F[8]); 
WRITE(F[9]); 
WRITE(F[10]); 
END;
{ OUTPUT FOR BETWEEN 20 AND 30 CELLS }
IF ((NCOLL2) AND (NCOLL2 < 30)) THEN
BEGIN
WRITE(F[1]); 
WRITE(F[2]); 
WRITE(F[3]); 
WRITE(F[4]); 
WRITE(F[5]); 
WRITE(F[6]); 
WRITE(F[7]); 
WRITE(F[8]); 
WRITE(F[9]); 
WRITE(F[10]); 
END;
{ X-AXIS FOR BETWEEN 10 AND 20 CELLS }
IF ((NCOLL2) AND (NCOLL2 < 20)) THEN
BEGIN
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
END;
{ X-AXIS FOR BETWEEN 20 AND 30 CELLS }
IF ((NCOLL2) AND (NCOLL2 < 30)) THEN
BEGIN
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
END;
{ X-AXIS FOR 10 OR LESS CELLS }
IF ((NCOLL2) AND (NCOLL2 < 10)) THEN
BEGIN
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
END;
{ X-AXIS FOR 10 OR LESS CELLS }
IF ((NCOLL2) AND (NCOLL2 < 10)) THEN
BEGIN
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
WRITE(L); 
END;
WRITELN(' ',MIDPT[2]:9:3,' ',MIDPT[5]:9:3,' ',MIDPT[8]:9:3);
WRITELN(' ',MIDPT[3]:9:3,' ',MIDPT[6]:9:3,' ',MIDPT[9]:9:3);
END;
READLN(RES);
END;
{ DISPLAY ANOTHER HISTOGRAM ? }
ANS:="Y";
WHILE (ANS='Y')AND(ANS='N') DO
BEGIN
  CLRSCR;WRITELN;WRITELN('Would you like to display a ',WHICH,', for another vector ? (Y/N)');
  READLN(ANS);
END;
{ RETURN TO MAIN MENU }
'\$':ASSIGN(VSTAT,'VSTAT.COM');
EXECUTE(VSTAT);
END(FREQ).
{ SCATTER GRAPH }

PROGRAM SCAT;
{ STORAGE SPECIFICATIONS }
LABEL 10;

TYPE
STRING2=STRING[2];
IRAY2=ARRAY[1..2]OF INTEGER;
RRAY2=ARRAY[1..2]OF REAL;
SRAY2=ARRAY[1..2]OF STRING[8];
RANGE=ARRAY[1..1000]OF REAL;
RAY20=ARRAY[1..20]OF REAL;

VAR
DRIVE:STRING2;
VIEW:INTEGER;
HI,LO,XX,YY,DATA,INTY,XMAX,XMIN,YMAX,YMIN,IX,II:REAL;
NUM,J,I: INTEGER;
OUTPUT,RES,ANS:CHAR;
OK:BOOLEAN;
INTER:ARRAY[2];
N:IRAY2;
SCATT:SRAY2;
LABX,LABY:RAY20;
WHICH:STRING[6];
V,W:RANGE;
VECTORFILE:FILE OF REAL;
VSTAT:FILE;

{ VARIABLE DEFINITIONS }

DRIVE - Current data disk-drive.
XX - Scaled element from first vector.
YY - Scaled element from second vector.
DATA - An element of a vector.
INTX - Interval between labels for x-axis.
INTY - Interval between labels for y-axis.
XMAX - Max. value in first vector.
YMAX - Max. value in second vector.
XMIN - Min. value in first vector.
YMIN - Min. value in second vector.
IX - X-axis increments.
IY - Y-axis increments.
X - Scaled x-coordinate to plot.
Y - Scaled y-coordinate to plot.
N,NUM - Number of elements in either vector.
J - Counter for vector arrays.
I - Counter for file name and file size arrays.
RES,ANS - User's response to yes/no questions.
OUTPUT - 'Y' for hard copy of results.
OK - True if files exist.
SCAT - Array containing vector file names.
LABX - Array containing x-axis labels.
LABY - Array containing y-axis labels.
WHICH - 'first' or 'second' for vector name entry.
V,W - Arrays for first and second vectors, respectively.
VECTORFILE - File name variable.
VSTAT - File containing main menu.

{ MAIN PROGRAM }
BEGIN

ANS="Y";
WHILE ANS="Y" DO ( CREATE LOOP TO REPEAT THIS STATISTIC IF DESIRED )
BEGIN
WHICH="first";
FOR J=1 TO 2 DO
BEGIN
CLRSCR;WRITELN;WRITELN(' SCATTER GRAPH');
{ LOCATE DESIRED VECTOR }
REPEAT ( CHECK FOR RETURN W/O ENTRY )
SCATT[J]="";
WHILE (LENGTH(SCATT[J])<1)OR(LENGTH(SCATT[J])>6) DO
BEGIN

END
END
WHILE ANS="Y" DO;
WHICH="second";
FOR J=1 TO 2 DO
BEGIN
CLRSCR;WRITELN;WRITELN(' SCATTER GRAPH');
{ LOCATE DESIRED VECTOR }
REPEAT ( CHECK FOR RETURN W/O ENTRY )
SCATT[J]="";
WHILE (LENGTH(SCATT[J])<1)OR(LENGTH(SCATT[J])>6) DO
BEGIN

END
END
WHILE ANS="Y" DO;
{ PROMPT USER FOR VECTOR NAME }
WHITELN;WHITELN('What is the name of the \',WHICH,\' vector.?');
READLN(SCATT[J]);
END:
{ ATTACH DISK-DRIVE TO NAME }
SCATT[J]:=DRIVE+SCATT[J];
{ CHECK THAT FILE EXISTS }
ASSIGN_VECTORFILE,SCATT[J];
{ reset VECTORFILE \$1+ }
OK:=(result:=0);
CLRSCR;
IF NOT OK THEN
{ PRINT ERROR MESSAGE }
BEGIN
WHITELN;WHITELN('VECTOR ',SCATT[J],', DOES NOT EXIST !');
ANS:="M"
WHILE (ANS)()'Y')AND(ANS)()'N') DO
BEGIN
WHITELN;WHITELN('Would you like to try again to locate a vector to do math with ?

(Y/N)');
READLN(ANS);
IF ANS='N' THEN GOTO 10;
END;
END;
END;
UNTIL OK;
{ DETERMINE LENGTH OF VECTOR }
N[J]:=FILESIZE(VECTORFILE);
{ READ VECTOR TO AN ARRAY }
FOR I:=1 TO N[J] DO
BEGIN
READ(VECTORFILE,DATA);
IF J=1 THEN
V[I]:=DATA
ELSE
W[I]:=DATA;
END:
{ CLOSE FILE }
CLOSE(VECTORFILE); WHICH:="second";
END;
{ CHECK THAT VECTOR LENGTHS ARE EQUAL }
IF (M[I] () N[J]) THEN
{ PRINT ERROR MESSAGE }
BEGIN
CLRSCR;WHITELN;WHITELN('These two vectors are NOT the same length !');
WHITELN;WHITELN('Press RETURN to continue.');
READLN(ANS);GOTO 10;
END;
NUM:=N[I];
{ DETERMINE OUTPUT DEVICE }
OUTPUT:="M"
WHILE (OUTPUT)()'Y')AND(OUTPUT)()'N') DO
BEGIN
CLRSCR;WHITELN;WHITELN('Do you want a HARD COPY of the scatter graph ? (Y/N)');
READLN(OUTPUT);
END;
{ FIND MAXIMUM AND MINIMUM VECTOR VALUES }
XMIN:=0;XMAX:=0;YMIX:=0;YMAX:=0;
XMIN:=V[I];XMAX:=V[I];YMIX:=W[I];YMAX:=W[I];
FOR I:=2 TO NUM DO
BEGIN
* IF V[I]XMIN THEN XMIN:=V[I];
IF V[I]XMAX THEN XMAX:=V[I];
IF W[I]YMIX THEN YMIX:=W[I];
IF W[I]YMAX THEN YMAX:=W[I];
END;
{ CALCULATE AXIS LABELS INTERVALS }
INTX:=(XMAX-XMIN)/5;
INTY:=(YMAX-YMIN)/5;
INTER[1]:=INTX;INTER[2]:=INTY;
FOR L:=1 TO 2 DO
BEGIN
{ ADJUST SCALING }

IF INTER[L]<=0.05 THEN
    INTER[L]:=0.05
ELSE
IF INTER[L]<=0.1 THEN
    INTER[L]:=0.1
ELSE
IF INTER[L]<=0.2 THEN
    INTER[L]:=0.2
ELSE
IF INTER[L]<=0.5 THEN
    INTER[L]:=0.5
ELSE
IF INTER[L]<=1.0 THEN
    INTER[L]:=1.0
ELSE
IF INTER[L]<=2.0 THEN
    INTER[L]:=2.0
ELSE
IF INTER[L]<=5.0 THEN
    INTER[L]:=5.0;
FOR NI:=2 TO 1000 DO
BEGIN
    LOW:=(5.0*(NI-1));
    HI:=(5.0*NI);
    IF ((INTER[L]<=LOW)AND(INTER[L]<=HI)) THEN
        INTER[L]:=HI;
END;
IF INTER[L]>5000 THEN
BEGIN
    CLRSCR;WRITELN('The intervals between data points are much too wide to provide a');
    WRITELN('meaningful scatter graph.');
    READING(S);GOTO 10;
END;
INXT:=INTER[1];INTY:=INTER[2];
END;
{ DETERMINE GRAPH AXIS LABELS }
LABX[1]:=XMIN-LABY[1]:=YMIN;
LABX[20]:=LABX[16]+INTX;LABY[20]:=LABY[16]+INTY;
{ INSTRUCTIONS TO PRINT SCATTER GRAPH }
IF OUTPUT='Y' THEN
BEGIN
    CLRSCR;WRITELN('Once the scatter graph has been displayed on the terminal, press the');
    WRITELN('SHIFT key and the PRTSC key to obtain a hard copy. ');WRITELN;
    READING(S);
END;
END;
{ PRINT GRAPH }
CLRSCR;WRITELN('VECTOR ',SCATT[2],'
{ PRINT Y-AXIS AND LABELS }
WRITELN(' ','LABY[20]:8:3:','-');
FOR I:=1 TO 3 DO
BEGIN
    WRITELN(' ');
END;
WRITELN(' ','LABY[16]:8:3:','-');
FOR I:=1 TO 3 DO
BEGIN
    WRITELN(' ');
END;
WRITELN(' ','LABY[12]:8:3:','-');
FOR I:=1 TO 3 DO
BEGIN
    WRITELN(' ');
END;
WRITELN(' ','LABY[8]:8:3:','-');
FOR I:=1 TO 3 DO
BEGIN

WRITE('');
END:
WRITE(' ',LABY[4]:8:3,' -');
FOR I:=1 TO 3 DO
BEGIN
WRITE('');
END;
PRINT X-AXIS AND LABELS
WRITE(' ',LABY[1]:8:3,' -------|-------|-------|-------|-------| VECTOR ',SCATT[1]);
WRITE(' ',LABX[4]:8:3,' ',LABX[12]:8:3,' ',LABX[20]:8:3);
WRITE(' ',LABX[1]:8:3,' ',LABX[8]:8:3,' ',LABX[16]:8:3);
{ CALCULATE AXIS INTERVALS }
IX:=(XMAX-XMIN)/20;
IY:=(YMAX-YMIN)/20;
{ SCALE DATA POINTS TO FIT GRAPH }
FOR I:=1 TO NUM DO
BEGIN
{ SCALE EACH COORDINATE }
XX:=ROUND((V[I]-XMIN)/IX);
YY:=ROUND((W[I]-YMIN)/IY);
{ DETERMINE COORDINATES TO PLOT }
X:=II+(2*TRUNC(XX));
Y:=TRUNC(22-YY);
{ PLOT EACH POINT }
GOTOXY(X,Y);
WRITE('');
END;
IF OUTPUT='Y' THEN READLN(RES);
READLN(RES);
{ DISPLAY ANOTHER SCATTER GRAPH ? }
ANS:="M";
WHILE (ANS<>"Y")AND(ANS<>"N") DO
BEGIN
CLSSCR;WRITE(ANS);
END;
RETURN TO MAIN MENU
IO:ASSIGN(VSTAT,'VSTAT.COM');
EXECUTE(VSTAT);
END;
PROGRAM FTTEST;
{ MAIN PROGRAM STORAGE SPECIFICATIONS }

LABEL 10;

TYPE
STRING2=STRING[2];
RANGE=ARRAY[1..1000] OF REAL;
IRAY2=ARRAY[1..2] OF INTEGER;
RAY2=ARRAY[1..2] OF REAL;
RAY3=ARRAY[1..3] OF REAL;
SRAY2=ARRAY[1..2] OF STRING[8];
STRING3=STRING[3];
STRING4=STRING[4];
STRING13=STRING[13];

VAR
DRIVE:STRING2;
VIEW:INTEGER;
STAT:CHAR;
J,CODE,1,J,W:INTEGER;
DATA,YTOTAL,XTOTAL,SUMX,SUMY,SUMXX,SUMYY,SUMX2,SUMY2,YO,ALPHA,YY,VXR,VYR,DXA,DYA:REAL;
OUTPUT,ANS,RES:CHAR;
OK:BOOLEAN;
NDF,K:RAY2;
I,F:RAY3;
V,W:RANGE;
STALPHA:STRING3;
ENTRY:STRING4;
WHICH:STRING[10];
FNAME:RAY2;
RESULT:STRING[9];
STY0,STK:STRING13;
VECTORFILE:FILE OF REAL;
VSTAT:FILE;

{ MAIN MENU - VARIABLE DEFINITIONS }

DRIVE - Current data disk-drive.
STAT - Signals which F-test to conduct.
J,I,W - Array counters.
CODE - Signals data conversion error.
DATA - Elements of the data vector.
SUMX,XTOTAL - Sum of elements of first vector.
SUMY,TOTAL - Sum of elements of second vector.
SUMXX - Sum of each element squared of first vector.
SUMYY - Sum of each element squared of second vector.
SUMX2 - Sum of the elements of the first vector squared.
SUMY2 - Sum of the elements of the second vector squared.
YO - Sample variance entered by user.
OUTPUT - User's response to output device desired.
ANS - User's response to Y/N questions.
RES - User's response to menu options.
OK - True if vector exists in a file.
K - Length of the data vector.
NDF - Degrees of freedom.
I,F - Probability of both tails.
Y,N - Calculated F statistic.
V,W - Arrays containing the first and second vectors, respectively.
STALPHA - Alpha value as string for validity check.
RESULT,ENTRY - Signals validity of response entered by user.
WHICH - "first" or "second", referencing the two data vectors.
FNAME - Name of file containing data vector.
STY0 - Variance entered by user as string format for validity check.
STK - Sample size entered by user as string format for validity check.
VECTORFILE - File containing vector of interest.
VSTAT - File containing the main menu.

{ CHECK FOR INVALID ALPHA }
PROCEDURE ALPHACK(VAR STALPHA:STRING3; VAR ENTRY:STRING4);
{ PROCEDURE ALPHACK STORAGE SPECIFICATIONS }

CHARACTERSET=SET OF CHAR;

VAR
COUNT: INTEGER;
POSITION: CHAR;
VALID: CHARACTERSET;

PROCEDURE ALPHACK - VARIABLE DEFINITIONS

COUNT - Counter of valid positions in entry.
POSITION - A character in the string.
POS - Position of a character in the string.
VALID - Set of valid characters.

BEGIN
{ INITIALIZE COUNTER }
COUNT := 0;
{ DEFINE VALID DATA SET }
VALID := ['0'..'9'];
{ CHECK FIRST POSITION OF ENTRY }
POSITION := COPY(STALPHA, 1, 1);
IF POSITION IN VALID THEN
    ENTRY := 'GOOD';
ELSE
    ENTRY := 'BAD';
END;
{ CHECK SECOND POSITION OF ENTRY }
POSITION := COPY(STALPHA, 2, 1);
IF POSITION IN VALID THEN
    COUNT := COUNT + 1;
ELSE
    VALID := ['0'..'9'];
END;
{ CHECK THIRD POSITION OF ENTRY }
POSITION := COPY(STALPHA, 3, 1);
IF POSITION IN VALID THEN
    COUNT := COUNT + 1;
END;
IF COUNT = 3 THEN
    ENTRY := 'BAD';
ELSE
    ENTRY := 'GOOD';
END;

PROCEDURE CHECK - VARIABLE DEFINITIONS

VALID - Valid set of characters.
POS - Position of a character in the string.
COUNT - Counter for valid positions in string.
POSITION - A character in the string.

BEGIN
{ INITIALIZE COUNTER }
COUNT := 0;
{ DEFINE VALID DATA SET }
VALID := ['0'..'9'];
{ CHECK ALL POSITIONS OF THE ENTRY }
FOR POS := 1 TO LENGTH(STK) DO
BEGIN
    POSITION := COPY(STK, POS, 1);
    IF POSITION IN VALID THEN
        COUNT := COUNT + 1;
    END;
    IF POSITION = '.' THEN
        { REDEFINE VALID DATA SET IF DECIMAL ENTERED }
        VALID := ['0'..'9'];
    END;
END;
{ COMPARE # OF VALID POSITIONS TO STRING LENGTH }

IF COUNT(LENGTH(STK)) THEN
  ENTRY:='BAD'
ELSE
  ENTRY:='GOOD'
END;

{ RAISE TO A POWER }
PROCEDURE POWER(VAR X,Y,Pow:REAL);
{ PROCEDURE POWER STORAGE SPECIFICATIONS }
LABEL 5;
VAR
  Num,XX,P:REAL;
  YY:INTEGER;
{ PROCEDURE POWER - VARIABLE DEFINITIONS }
  Num - base times natural log of adjusted exponent.
  XX - Adjusted base.
  YY - Adjusted exponent.
  P - Exponential of Num.
BEGIN
  { ZERO EXPONENT }
  IF (Y=0) THEN
    BEGIN
      Pow:=1.0;
      GOTO 5;
    END;
  { ZERO BASE }
  IF (X=0) THEN
    BEGIN
      Pow:=0.0;
      GOTO 5;
    END
ELSE
  { NEGATIVE BASE }
  IF (X<0) THEN
    XX:=0.0-X
  ELSE
  { POSITIVE BASE }
    XX:=X;
  Num:=Y*LN(XX);
  P:=EXP(Num);
  IF (X<0) THEN
    BEGIN
      YY:=TRUNC(Y);  { ADJUST SIGNS }
      IF (ODD(YY)=TRUE) THEN
        Pow:=0.0-P
      ELSE
        Pow:=P
    END
ELSE
  Pow:=P;
END;

{ CALCULATE F-VALUE }
PROCEDURE FF(VAR NDF:ARRAY[1..2] OF REAL;
VAR
  T,Y2,XX:RAY2;
  DF:RAY2;
  I:INTEGER;
{ PROCEDURE FF - VARIABLE DEFINITION }
{ DF - Degrees of freedom. }
BEGIN [ INITIALIZE ARRAYS ]
FOR I:=1 TO 3 DO
  Z[I]:=0;
  FT:=0;
  DF[1]:=MDF[1];
  DF[2]:=MDF[2];
  IF MDF[1] EVEN & (200 } IF ODD(MDF[1])=FALSE THEN
  IF MDF[1]<200 THEN
    BEGIN
      U[2]:=-0[1];
      DF[1]:=(DF[1]-2;
      WHILE DF[1]:=1 DO
        BEGIN
          VP:=VP-2;
          FT:=(0[1]*VP/DF[1]*(1+FT));
          DF[1]:=(DF[1]-2;
        END;
        { RAISE TO A POWER }
        X:=(0[1];
        Y:=0.5*DF[2];
        POWER(X,Y,POW);
        FT:=(POW*(1+FT));
        GOTO 10;
      END;
      IF ODD(MDF[2])=FALSE THEN
        IF MDF[2]<200 THEN
          BEGIN
            DF[2]:=(DF[2]-2;
            WHILE (DF[2]):1 DO
              BEGIN
                VP:=VP-2;
                FT:=(0[1]*VP/DF[2]*(1+FT));
                DF[2]:=(DF[2]-2;
              END;
              { RAISE TO A POWER }
              X:=(0[1];
              Y:=0.5*DF[1];
              POWER(X,Y,POW);
              FT:=(POW*(1+FT));
              GOTO 10;
            END;
\[
\text{AJ} := \text{SH*CH*(1+AJ)}; \\
\text{END}; \\
\text{AJ} := \text{TA*AJ}; \\
\text{IF} (\text{NDF}[1]+1) \text{ THEN} \\
\begin{align*}
\text{DF}[1] &:= \text{TRUNC}[2/(9*\text{DF}[1])]; \\
\text{DF}[2] &:= \text{TRUNC}[2/(9*\text{DF}[2])]; \\
\end{align*}
\text{RAISE TO A POWER} \\
\begin{align*}
X &:= \text{CH}; \\
Y &:= \text{NDF}[2]; \\
\text{POW} &:= (X,Y,\text{POW}); \\
\end{align*}
\text{CB} := ((1-\text{DF}[2])*\text{CB}+\text{DF}[1])-1/\text{SQRT}(\text{DF}[2]*(\text{POW}+\text{DF}[1])); \\
\text{YX}[1] := \text{VA} \cdot \text{YX}[2] := 0; \\
\text{DISTRI} &:= \text{N} \\
\text{YX}[2] &:= \text{YX}[1]; \\
\text{IF} (\text{YX}[2]) &:= 4.17 \text{ THEN} \\
\text{VN} &:= 1 \\
\text{ELSE} \\
\text{IF} (\text{YX}[2]) &:= -4.17 \text{ THEN} \\
\text{VN} &:= 0 \\
\text{ELSE} \\
\begin{align*}
\text{Y}[2][1] &:= \text{YX}[2]; \\
\text{IF} (\text{Y}[2][1]) &:= 0 \text{ THEN} \\
\text{Y}[2][2] &:= \text{Y}[2][2]; \\
\end{align*}
\begin{align*}
\text{D} &:= (((1.4306384-4*\text{T}[2]+0.2765672E-3)*\text{T}[2]+0.1520143E-3)*\text{T}[2]+0.9270527E-2)*\text{T}[2]+0.42282012E-1)*\text{T}[2] \times 0.70523078E-1/\text{T}[2][1]) \\
\end{align*}
\text{D} := \text{D*D}; \text{D} := \text{D*D}; \text{D} := \text{D*D}; \text{D} := \text{D*D}; \\
\text{VN} := 0.5-0.5/\text{D}; \\
\text{IF} (\text{YX}[2]) &:= 0 \text{ THEN} \text{VN} := 0.5-\text{VN}; \\
\text{IF} (\text{YX}[2]) &:= 0 \text{ THEN} \text{VN} := 0.5; \\
\text{IF} (\text{YX}[2]) &:= 0 \text{ THEN} \text{VN} := 0.5+\text{VN}; \\
\end{align*}
\text{END} \\
\{ END N-DISTRIBUTION \} \\
\text{FT} := \text{VN} \\
\text{END}; \\
\text{IO}; \\
\text{IF} \text{FT} &:= 0 \text{ THEN} \\
\text{Z}[1] := 0 \\
\text{ELSE} \\
\text{Z}[1] := \text{FT}; \\
\text{END} (\text{FP}) ;
I

HAIN

PROGRAH

BEGIN

'CLUDSCR;WRITELN;WRITELN('F-TEST SECTION');
WRITELN('An F-TEST will be performed on 2 samples of data to test the hypothesis');
WRITELN('that the variances of the samples are equal. Normality is assumed.');
WRITELN('Press RETURN to continue:');
READLN(RES);
ANS:='Y';
LOOP TO REPEAT TEST IF DESIRED
WHILE ANS:='Y' DO
BEGIN
WHICH:='first';
FOR Q:=1 TO 2 DO
BEGIN
{ F-TEST WHERE N, MU AND SIGMA-SQUARED KNOWN }
IF STAT:='1' THEN
BEGIN
{ ENTER FIRST SAMPLE SIZE }
ENTRY:='BAD';
WHILE ENTRY:='BAD' DO
BEGIN
CLSCR;WRITELN;WRITELN('Please enter the following information for the ',WHICH,' sample:');
WRITELN('Sample size:');
READLN(STK);
{ CHECK FOR INVALID ENTRY }
CHECK(STK,ENTRY);
END;
{ CHANGE STRING TO INTEGER }
VAL(STK,K[Q],CODE);
K[Q]:=TRUNC(K[Q]);
{ ENTER FIRST VARIANCE }
ENTRY:='BAD';
WHILE ENTRY:='BAD' DO
BEGIN
WRITELN('Sample variance (sigma-squared):');
READLN(StY0);
{ CHECK FOR INVALID ENTRY }
CHECK(STY0,ENTRY);
END;
{ CHANGE STRING TO REAL }
VAL(STY0,YO,CODE);
IF Q=1 THEN VX:=-YO;
IF Q=2 THEN VY:=YO;
END;
{ F-TEST WHERE N, MU AND SIGMA-SQUARED UNKNOWN }
IF STAT:='2' THEN
BEGIN
J:=Q;
REPEAT
{ CHECK FOR RETURN W/O ENTRY }
FNAME[J]:=''
WHILE (LENGTH(FNAME[J])<1)OR(LENGTH(FNAME[J])>6) DO
BEGIN
PROMPT FOR NAME OF VECTOR
CLSCR;WRITELN;WRITELN('What is the name of the ',WHICH,' vector ?');
READLN(FNAME[J]);
END;
{ ATTACH DISK-DRIVE TO NAME }
FNAME[J]:=DRIVE+FNAME[J];
{ CHECK THAT VECTOR EXISTS }
ASSIGN VECTORFILE,FNAME[J];
{ IF NOT RESET VECTORFILE }
OK:=[IOresult=0];
CLSCR;
IF NOT OK THEN
{ PRINT ERROR MESSAGE }
BEGIN
WRITELN('VECTOR ',FNAME[J],', DOES NOT EXIST(');
ANS:='M';
WHILE ANS='M' AND ANS='N' DO
BEGIN
WRITELN('Would you like to try again to locate a vector ? (Y/N)');
END;
READLN(ANS);
END;
IF ANS='N' THEN GOTO 10;
END;
UNTIL OK;
{ DETERMINE SIZE OF VECTOR }
K[J]:=FILESIZE(VECTORFILE);
{ READ VECTOR TO ARRAY }
FOR J:=1 TO K[J] DO
BEGIN
READ(VECTORFILE,DATA);
IF J=1 THEN
V[I]:=DATA
ELSE
W[I]:=DATA;
END;
{ CLOSE FILE }
CLOSE(VECTORFILE);
END;
WHICH:='second';
END-
{ ENTER ALPHA VALUE }
ENTRY:='BAD';
WHILE ENTRY='BAD' DO
BEGIN
{ CHECK FOR RETURN W/O ENTRY }
STALPHA:=''
WHILE LENGTH(STALPHA)<3 DO
BEGIN
{ PROMPT FOR ALPHA LEVEL }
CLRSRC;WRITELN;WRITELN('Enter the desired ALPHA (0 < ALPHA < 1) in decimal form for significance test:');
WRITELN('Examples: 5% is entered as .05, 10% is entered as .10');
READLN(STALPHA);
END;
{ CHECK FOR INVALID ENTRY }
ALPHACK(STALPHA,ENTRY);
END;
{ CHANGE STRING TO REAL }
VAL(STALPHA,ALPHA,ENTRY);
{ CALCULATE VARIANCES IF UNKNOWN }
IF STAT='2' THEN
BEGIN
{ VAR OF FIRST VECTOR }
XTOTAL:=0;
FOR J:=1 TO K[1] DO
XTOTAL:=XTOTAL+W[J];
SUMX:=XTOTAL*SUMX:=0;
FOR J:=1 TO K[1] DO
SUMX:=SUMX*(V[J]*W[J]);
SUMX2:=SUMX*SUMX;
VXR:=(SUMX-SUMX2/K[1])/(K[1]-1);
{ VAR OF SECOND VECTOR }
YTOTAL:=0;
FOR J:=1 TO K[2] DO
YTOTAL:=YTOTAL+W[J];
SUMY:=TOTAL*SUMY:=0;
FOR J:=1 TO K[2] DO
SUMY:=SUMY+(W[J]*V[J]);
SUMY2:=SUMY*SUMY;
VYR:=(SUMY-SUMY2/K[2])/(K[2]-1);
END;
{ CALCULATE STD. DEV. }
DXA:=SQRT(VXR);DYA:=SQRT(VYR);
{ CALCULATE DEGREES OF FREEDOM }
NDOF:=K[1]-1;NDOF:K[2]-1;
{ CALCULATE F-VALUE }
F[I]:=VXR/VYR;
IF F[I]=0 THEN
Z[I]:=1
ELSE
{ CALL PROCEDURE TO CALCULATE PROBABILITY }
IF \( I[1] \geq \text{ALPHA} \) THEN
\[
\text{RESULT::'ACCEPT Ho'}
\]
ELSE
\[
\text{RESULT::'REJECT Ho'};
\]
\[
\text{OUTPUT::'H1-'}
\]
WHILE \((\text{OUTPUT}='Y')\) AND \((\text{OUTPUT}='N')\) DO
BEGIN
\text{CLSCHR;WRITELN('Would you like a HARD COPY of the results ? (Y/N)');}
\text{READLN(OUTPUT)};
END;
\[
\text{RESULTS OF F-TEST}
\]
IF OUTPUT: 'N' THEN
BEGIN
\text{PRINT OUTPUT ON CRT}
\text{CLSCHR;WRITELN('F-TEST');WRITELN};
\text{IF STAT:'I' THEN}
\text{WRITELN('Sample 1')}
ELSE
\text{WRITELN('VECTOR ',FNAME[1]);}
\text{WRITELN('Number of Observations = ',K[1]:4);}
\text{WRITELN('Variance = ',VAR:8:4);}
\text{WRITELN('Standard Deviation = ',DXA:8:4);}
\text{WRITELN('Sample 2.');}
\text{WRITELN('Number of Observations = ',K[2]:4);}
\text{WRITELN('Variance = ',VAR:8:4);}
\text{WRITELN('Standard Deviation = ',DXA:8:4);}
\text{WRITELN('Selected ALPHA = ',ALPHA:8:4);}
\text{WRITELN('Calculated F-Value (Fo) = ',F[3]:8:4);}
\text{WRITELN('D.F. in numerator (Sample 1) = ',NDF[1]:4);}
\text{WRITELN('D.F. in denominator (Sample 2) = ',NDF[2]:4);}
\text{WRITELN('Probability of right tail = ',Z[3]:8:4);WRITELN;}
\text{WRITELN('Thus: ',RESULT,)
\text{Press RETURN to continue:');}
\text{READLN(RES)};
\text{END:
\text{BEGIN
\text{SEND OUTPUT TO PRINTER }
\text{WRITELN(LST,'F-TEST');WRITELN(LST)
\text{IF STAT='I' THEN}
\text{WRITELN(LST,'Sample 1')}
ELSE
\text{WRITELN(LST,'VECTOR ',FNAME[1]);}
\text{WRITELN(LST,'Number of Observations = ',K[1]:4);}
\text{WRITELN(LST,'Variance = ',VAR:8:4);}
\text{WRITELN(LST,'Standard Deviation = ',DXA:8:4);}
\text{WRITELN(LST,'Sample 2');}
\text{WRITELN(LST,'VECTOR ',FNAME[2]);}
\text{WRITELN(LST,'Number of Observations = ',K[2]:4);}
\text{WRITELN(LST,'Variance = ',VAR:8:4);}
\text{WRITELN(LST,'Standard Deviation = ',DXA:8:4);}
\text{WRITELN(LST,'Selected ALPHA = ',ALPHA:8:4);}
\text{WRITELN(LST,'Calculated F-Value (Fo) = ',F[3]:8:4);}
\text{WRITELN(LST,'D.F. in numerator (Sample 1) = ',NDF[1]:4);}
\text{WRITELN(LST,'D.F. in denominator (Sample 2) = ',NDF[2]:4);}
\text{WRITELN(LST,'Probability of right tail = ',Z[3]:8:4);WRITELN(LST);}
\text{WRITELN(LST,'Thus: ',RESULT);
\text{Press RETURN to continue:');}
\text{READLN(RES)};
\text{END
\text{BEGIN
\text{SEND OUTPUT TO PRINTER }
\text{WRITELN(LST,'F-TEST');WRITELN(LST)
\text{IF STAT='I' THEN}
\text{WRITELN(LST,'Sample 1')}
ELSE
\text{WRITELN(LST,'VECTOR ',FNAME[1]);}
\text{WRITELN(LST,'Number of Observations = ',K[1]:4);}
\text{WRITELN(LST,'Variance = ',VAR:8:4);}
\text{WRITELN(LST,'Standard Deviation = ',DXA:8:4);}
\text{WRITELN(LST,'Selected ALPHA = ',ALPHA:8:4);}
\text{WRITELN(LST,'Calculated F-Value (Fo) = ',F[3]:8:4);}
\text{WRITELN(LST,'D.F. in numerator (Sample 1) = ',NDF[1]:4);}
\text{WRITELN(LST,'D.F. in denominator (Sample 2) = ',NDF[2]:4);}
\text{WRITELN(LST,'Probability of right tail = ',Z[3]:8:4);WRITELN(LST);}
\text{WRITELN(LST,'Thus: ',RESULT);
\text{Press RETURN to continue:');}
\text{READLN(RES)};
\text{END
\text{ANS='W';}
WHILE (ANS(!'y') AND (ANS(!'n')) DO
BEGIN
CLRSCR;WRITELN;WRITELN('Would you like to run this section again ? (Y/N)');
READLN(ANS);
END;
END;

RETURN TO MAIN MENU;
10:assign(vstat,'vstat.com');
execute(vstat);
{ T - T E S T }
PROGRAM TTEST;
LABEL 10;
TYPE
STRING2:STRING[2];
IRAY2=ARRAY[1..2] OF INTEGER;
RANGE=ARRAY[1..1000] OF REAL;
STRING3:STRING[3];
STRING4:STRING[4];
STRING13:STRING[13];
STRING60:STRING[60];
VAR
DRIVE:STRING2;
VIEW:INTEGER;
STAT:CHAR;
ANS,RES,POSITION,OUTPUT:CHAR;
VECTORFILE:FILE OF REAL;
N:ARRAY;
V,W:RANGE;
SAMP1,SAMP2,SAMP:STRING[8];
VSTAT:FILE;
OK:BOOLEAN;
CODE,J,I,ALPHA,DF,NDF,DFD2:INTEGER;
SIX,DATA,SX,SXX,SY,SYY,S1,S2,S1,D1,D2,TD,SD,T1,21,YX1,DAX,AXV,AIVYRX,VRX,VRY,2,T3,T4,T3,T3:REAL;
STALPHA:STRING3;
ENTRY:STRING4;
WHICH:STRING[4];
DECISION:STRING[6];
ST:STRING13;
ERR:STRING60;

VARIABLE DEFINITIONS - MAIN PROGRAM

DRIVE - Current data disk-drive.
STAT - Designates which t-test is to be performed.
ANS - User's response to Y/N questions.
RES - User's response to menu selections.
POSITION - A single character of a string.
VECTORFILE - A file containing the vector of interest.
N - Array containing the lengths of each vector.
V,W - Arrays containing the first and second vectors, respectively.
SAMP1,SAMP2,SAMP - Vector names.
VSTAT - File containing the main menu.
OK - True if file of interest exists.
CODE - Signals error in data conversion.
J,I - Counters for the arrays containing the vectors.
ALPHA - Alpha level entered by user.
DF - Length of vector minus one.
NDF - Degrees of freedom for Paired T-test.
FD2 - Degrees of freedom for T-test.
DATA - Element of the data vector.
AXV - Mean of the first vector.
VXR - Variance of the first vector.
DAX - Standard Deviation of the first vector.
ATV - Mean of the second vector.
VVX - Variance of the second vector.
DVA - Standard Deviation of the second vector.
SX - Sum of the elements in the first vector.
SY - Sum of the elements in the second vector.
SXX - Sum of each element squared in the first vector.
SYY - Sum of each element squared in the second vector.
D1 - Difference between elements of first vector and elements of second.
D2 - Sum of difference.
SIX - Average difference.
SD - Standard Deviation of the difference.
TD - Used in calculation of SD.
YX1,T1,T1,T1,S1,S2 - Used in the calculation of the t-value.
T3 - Calculated t-value for Paired T-test.
T4 - Calculated t-value for T-test.
Z3 - Probability of both tails for Paired T-test.
Z4 - Probability of both tails for T-test.

PROCEDURE ALPHACK(VAR STALPHA:STRING3; VAR ENTRY:STRING4);

TYPE
CHARACTERSET=SET OF CHAR;

VAR
COUNT:INTEGER;
POSITION:CHAR;
VALID:CHARACTERSET;

BEGIN

PROCEDURE ALPHACK - VARIABLE DEFINITIONS

COUNT - Used to count number of valid positions in a string.
POSITION - A single character of a string.
VALID - A set of valid characters.

BEGIN
{ INITIALIZE COUNTER }
COUNT:=0;
{ DEFINE VALID DATA SET }
VALID:=['0'..'9'];
{ CHECK FIRST POSITION OF ENTRY }
POSITION:=COPY(STALPHA,1,1);
{ IF FIRST NOT DECIMAL THEN ENTRY INVALID }
IF POSITION('.' THEN
ENTRY:= 'BAD'
ELSE
ENTRY:= 'GOOD';
END;
{ CHECK SECOND POSITION OF ENTRY }
POSITION:=COPY(STALPHA,2,1);
IF POSITION IN VALID THEN
COUNT:=COUNT+1;
IF POSITION='D' THEN
{ REDEFINE VALID DATA SET }
VALID:=['1'..'9'];
{ CHECK THIRD POSITION OF ENTRY }
POSITION:=COPY(STALPHA,3,1);
IF POSITION IN VALID THEN
COUNT:=COUNT+1;
{ COMPARE # VALID POSITIONS TO LENGTH OF STRING }
IF COUNT>2 THEN
ENTRY:= 'BAD'
ELSE
ENTRY:= 'GOOD';
END;
END(ALPHACK);

PROCEDURE CHECK(VAR ST:STRING13; VAR ERR:STRING60);

TYPE
CHARACTERSET=SET OF CHAR;

VAR
VALID:CHARACTERSET;
POS,COUNT:INTEGER;
POSITION:CHAR;

BEGIN

PROCEDURE CHECK - VARIABLE DEFINITIONS

VALID - Set of valid characters.
POS - Position of a character within a string.
COUNT - Used to count the number of valid characters in a string.
POSITION - A single character of a string.

BEGIN
{ DEFINE VALID SET }
VALID:=['0'..'9',',',.''];
{ INITIALIZE COUNTER }
COUNT:=0;
{ CHECK FIRST POSITION }
POSITION:=COPY(ST,1,1);
IF POSITION IN VALID THEN
COUNT:=COUNT+1;
IF POSITION='.' THEN
   { REDEFINE VALID SET }
   VALID:=['0'..'9']
ELSE
   VALID:=['0'..'9'];
   { CHECK REMAINING POSITIONS }
FOR POS:=2 TO LENGTH(ST) DO BEGIN
   POSITION:=COPY(ST,POS,1);
   IF POSITION IN VALID THEN
      COUNT:=COUNT+1;
   IF POSITION='.' THEN
      VALID:=['0'..'9'];
END;
{ COMPARE # OF VALID POSITIONS TO LENGTH OF STRING }
IF COUNT<LENGTH(ST) THEN BEGIN
   ERR:='INVALID ENTRY! - Please try again.';
   EXIT
END
ELSE
   ERR:='NONE';
END(CHECK);
{ T DISTRIBUTION }
PROCEDURE TDISTR(VAR Z1,B1,YX1:REAL; VAR DF1:INTEGER);
VAR
   A1:REAL;
BEGIN
   { PROCEDURE TDISTR - VARIABLE DEFINITION }
   { Z1,B1,YX1 - Used to calculate probability of both tails. }
   { DF1 - Degrees of freedom. }
   { CALCULATE PROBABILITY }
   YX1:=SQRT(YX1);
   A1:=YX1;
   IF (DF1=1) THEN A1:=0;
   DF1:=DF1-2;
   WHILE (DF1>1) DO BEGIN
      A1:=(DF1-1)/(B1*DF1)*A1+YX1);
      DF1:=DF1-2;
      END;
   IF (DF1<0) THEN
      A1:=(ARCTAN(YX1)+A1/B1)*0.63661977236
   ELSE
      A1:=A1/SQRT(B1);
   IF (A1<1) THEN A1:=0;
   Z1:=1-A1;
   END(TDISTR);
{ M DISTRIBUTION }
PROCEDURE NDISTR(VAR VN,YX1:REAL);
VAR
   YX2,Y2,Z2,T2,D:REAL;
BEGIN
   YX2:=-YX1;
   IF (YX2<-4.17) THEN BEGIN
      VN:=0;
      EXIT;
   END;
   IF (YX2>-4.17) THEN BEGIN
      VN:=1;
      EXIT;
   END;
   Y2:=YX2;
IF (YX2(0)) THEN YZ2:=(-YX2);
T2:=YX2/1.4142142;
D:=(((0.430638E-4*T2+0.27657672E-3)*T2+0.1520143E-3)*T2+0.92705272E-2)*T2+0.42202012E-1)*T2+0.705230
70E-1)*T2+1.0);
D:=D^2;
D:=D^2*D:=D^2*D:=D^2;
VM:=D^5/5*D;
IF (YX2(0)) THEN VM:=D^5/5*D;
IF (YX2(0)) THEN VM:=D^5/5*D;
IF (YX2(0)) THEN VM:=D^5/5*D;
END(ENDISTR);
{ T & N DISTRIBUTION }
PROCEDURE INDISTR(VAR DF:INTEGER; VAR Z1,T1:REAL);
VAR
VM,A1,ITM,YX1,B1,TV:REAL;
DF1:INTEGER;
BEGIN
DF1:=DF1;
TV:=T1*T1;
YX1:=TV/DF1;
B1:=1+YX1;
IF (DF1<=20) THEN
BEGIN
{ CALL PROCEDURE FOR T-DISTRIBUTION }
TDISTR(Z1,B1,YX1,DF1);
EXIT;
END
ELSE
{ASYMPTOTIC SERIES FOR LARGE ?}
IF (YX1)(0.1E-5) THEN
YX1:=LN(B1);
A1:=DF1*0.5;
B1:=4B1+A1*A1;
YX1:=A1/YX1;
YX1:=(((0.4*YX1-3.3)*YX1-24)*YX1-85.5)/(0.8*(YX1)*YX1+100+B1)*YX1+3/B1+1)*SQRT(YX1);
{ CALL PROCEDURE N-DISTRIBUTION }
NDISTR(VN,YX1);
Z1:=2*VM;
IF (Z1) THEN Z1:=0;
END(ENDISTR);
{ CALCULATE t FOR AND VARIANCES }
PROCEDURE CALCT(VAR N:IRAY2; VAR FDF:INTEGER; VAR TS,TT,TS:REAL);
VAR
ATST1,ST1,DF1:INTEGER;
ASX,ATST,S,T1,T2,A1,A2,TEST,TT,TS,Z1:REAL;
BEGIN
{ CALCULATE t FOR VARIANCES }
DF1:=DF1;
S:=SQRT(((N[1]-1)*VXR+(N[2]-1)*VYR)/NDF);
T1:=((AXV-AYV)/(S)*(1/SQRT(((N[1]+1)/N[2])));
{ CALCULATE t FOR VARIANCES }
T2:=(AXV-AYV)/(S*VYR/(N[1]+1)*VYR/(N[2]));
TEST:=A1/A2; INT(A1/A2);
TEST:=10*TEST; TS:=INT(THS); TS:=TT-TS;
IF (TS) THEN
TEST:=TS/10+0.1
ELSE
TEST:=TS/10;
ASX:=INT(A1/A2)+TEST;
FDF:=N[1]+N[2];
{ CALL PROCEDURE T & N DISTRIBUTION }
INDISTR(T2,T1,T1);
T4:=T1;T4:=T1;T4:=T1;T4:=T1;T4:=T1;
END(CALC);  

{ t - T E S T ( N , M U , A N D S I G M A - S Q U A R E D K N O W N ) }

PROCEDURE TKNOWN(VAR AXV, VXV, AVY, VYR: REAL; VAR ERR: STRING60; VAR DF: INTEGER; VAR N: IRAY2);

TYPE
  RAY2 = ARRAY[1..2] OF REAL;

VAR
  AVG, VARI: RAY2;
  ST: STRING[13];
  WHICH: STRING[6];
  I, CODE: INTEGER;

BEGIN
  WHICH := 'first';
  FOR I := 1 TO 2 DO
  BEGIN
    CLRSCR; WRITELN; WRITELN('Please enter the following information for the ', WHICH, ', sample:');
    ERR := 'TESTING';
    WHILE LENGTH(ERR) > 4 DO
      BEGIN
        ST := '':
        WHILE LENGTH(ST) < 1 DO
          BEGIN
            WRITELN; WRITELN('What is the sample size (n) ?');
            READLN(ST);
          END;
          CHECK(ST, ERR);
          IF ERR = 'NONE' THEN
            VAL(ST, M[I], CODE)
          ELSE
            WRITELN(ERR);
          END;
          ERR := 'TESTING';
          WHILE LENGTH(ERR) > 4 DO
            BEGIN
              ST := '':
              WHILE LENGTH(ST) < 1 DO
                BEGIN
                  WRITELN; WRITELN('What is the sample mean (mu) ?');
                  READLN(ST);
                END;
                CHECK(ST, ERR);
                IF ERR = 'NONE' THEN
                  VAL(ST, AVG[I], CODE)
                ELSE
                  WRITELN(ERR);
                END;
                ERR := 'TESTING';
                WHILE LENGTH(ERR) > 4 DO
                  BEGIN
                    ST := '':
                    WHILE LENGTH(ST) < 1 DO
                      BEGIN
                        WRITELN; WRITELN('What is the sample variance (sigma-squared) ?');
                        READLN(ST);
                      END;
                      CHECK(ST, ERR);
                      IF ERR = 'NONE' THEN
                        VAL(ST, VARI[I], CODE)
                      ELSE
                        WRITELN(ERR);
                      END;
                      WHICH := 'second';
                  END;

  AXV := AVG[1]; AVY := AVG[2];
  VXV := VARI[1]; VYR := VARI[2];
  DF := N[I] - 2;

  TKNOWN;

  { M A I N P R O G R A M }

  BEGIN
    ANS := 'y';
WHILE (ANS='Y') DO
BEGIN
IF (STAT='A') OR (STAT='B') THEN
    BEGIN
        CLS; Writeln('t-TEST SECTION');
        Writeln('A t-Test will be performed on two vectors of data to test the hypothesis');
        Writeln('that the population means of the vectors are equal.');
        Writeln('Two assumptions will be tested:');
        Writeln('1) The population variances are unknown but assumed EQUAL.');
        Writeln(In both cases, normality is assumed.');
        (Press RETURN to continue:');
        READLN(RES);
    END;
IF STAT='A' THEN
    BEGIN
        WHICH='first';
    END;
FOR J:=1 TO 2 DO BEGIN
    SAMP:="";
    WHILE (LENGTH(SAMP)<>1) OR (LENGTH(SAMP)>6) DO
    BEGIN
        CLS; Writeln('What is the name of the vector you want to use as the ', WHICH, ' sample ? ');
        READLN(SAMP);
    END;
    IF WHICH='first' THEN
        SAMP1:=SAMP;
    IF WHICH='second' THEN
        SAMP2:=SAMP;
    IF NOT OK THEN BEGIN
        ANS:='M';
        WHILE (ANS<>"Y") AND (ANS<>"N") DO
        BEGIN
            Writeln('VECTOR ', SAMP, ' DOES NOT EXIST !');
            Writeln('Would you like to try again to locate a vector for this test ? (Y/N)');
            READLN(ANS);
            IF ANS='N' THEN GOTO 10;
        END;
    END;
    UNTIL OK;
    N[J]:=FILESIZE(VECTORFILE);
    IF N[J]<1 THEN
    BEGIN
        CLS; Writeln('YOUR SAMPLE DOES NOT HAVE ENOUGH DATA TO PERFORM THIS TEST. ');
        Writeln('Press RETURN to continue:');
        READLN(RES);
    END;
{READ SAMPLE TO ARRAY}
FOR I:=1 TO N[J] DO
BEGIN
    READ(VECTORFILE, DATA);
    IF J=1 THEN V[1]:=DATA;
    IF J=2 THEN V[2]:=DATA;
END;
CLOSE(VECTORFILE);
WHICH:='second';
END;
IF STAT='C' THEN
BEGIN
    BEGIN
        CLS; Writeln('The two vectors must be the SAME LENGTH for this test !');
    END;
END;
WRITELN;WRITELN;WRITELN('Press RETURN to continue.');
READLN(RES);GOTO 10;
END;
BEGIN
{CALCULATE SAMPLE AVERAGE AND VARIANCE}
SX:=0;SXX:=0;SY:=0;SYY:=0;
FOR I:=1 TO N[1] DO
BEGIN
SX:=SX+W[I];
SXX:=SXX+W[I]*W[I];
END;
FOR I:=1 TO N[2] DO
BEGIN
SY:=SY+W[I];
SYY:=SYY+W[I]*W[I];
END;
AXV:=SX/N[1];
AYV:=SY/N[2];
S1:=SXX-AXV*SX;
S2:=SYY-AYV*SY;
VXR:=S1/(N[1]-1); 
VYR:=S2/(N[2]-1);
IF STAT='B' THEN DF:=N[1]-1;
IF STAT='C' THEN
BEGIN
D1:=0;DF:=0;
FOR I:=1 TO N[1] DO
BEGIN
D1:=D1+W[I];
D1:=D1*W[I];
D2:=D2+D1*D1;
END;
S1:=D1/N[1];
END;
ENTRY:='BAD';
WHILE ENTRY='BAD' DO
BEGIN
STALPHA:='';
WHILE LENGTH(STALPHA)<3 DO
BEGIN
CLRSCR;WRITELN;WRITELN('Enter the desired ALPHA (in decimal form) for testing (Example: .05 is .05)');
END;
ALPHACK(STALPHA,ENTRY);
END:
BEGIN
{CALCULATE STANDARD DEVIATION}
DXA:=SQRT(VXR);DYA:=SQRT(VYR);
ELSE
BEGIN
T1:=-((D2-((D1*D1)/N[1]))/(N[1]-1));
IF T1<0 THEN T1:=0;
SD:=SD/SD;
END;
T1:=(AXV-AYV)/(SD/SQRT(N[1]));
END;
WRITELN;WRITELN('NOTE: 0 < ALPHA < 1');
READLN(STALPHA);
END;
ALPHACK(STALPHA,ENTRY);
BEGIN
{CALCULATE T-VALUE}
IF (STAT='A') OR (STAT='B') THEN
CALC(N,FDF2,T3,T4,Z3,T41);
ELSE
IF (DI) THEN
T1:=999999.0;
ELSE
IF (DI) THEN
T1:=-999999.0;
{CALL T & N DISTRIBUTION}
TNDISTR(DF,Z1,T1);
T3 := T1; Z3 := Z1;
END;
FOR I := 3 TO 4 DO
BEGIN
IF (I = 3) THEN
BEGIN
IF (Z2 = 3) THEN
DECISION := 'REJECT Ho'
ELSE
DECISION := 'ACCEPT Ho';
END;
IF (I = 4) THEN
BEGIN
IF (Z2 = 4) THEN
DECISION := 'REJECT Ho'
ELSE
DECISION := 'ACCEPT Ho';
END;
END;
{ PRINT RESULTS }
OUTPUT := 'M';
WHILE ((OUTPUT = 'Y') AND (OUTPUT = 'N')) DO
BEGIN
CLRSCR; WRITELN; WRITELN('Do you want a HARD COPY of the output?');
READLN(OUTPUT);
END;
IF OUTPUT = 'Y' THEN
BEGIN
CLRSCR; WRITELN; WRITELN('Once the result has been displayed on the terminal screen, press the');
WRITELN('SHIFT KEY and the PR'TSC KEY to obtain a hard copy. ');
WRITELN; WRITELN('Press RETURN to continue:');
READLN(RES);
END;
CLRSCR;
IF STAT = 'C' THEN
WRITELN('HYPOTHESIS TEST FOR PAIRED t-TEST')
ELSE
WRITELN('HYPOTHESIS TEST FOR t-TEST');
IF STAT = 'A' THEN
WRITELN('SAMPLE 1')
ELSE
WRITELN('VECTOR ', SAMP1);
WRITELN('Average ', AYV := 8:4); WRITELN('Variance ', VXR := 8:4); WRITELN('Standard Deviation ', DYA := 8:4); WRITELN;
IF STAT = 'A' THEN
WRITELN('SAMPLE 2')
ELSE
WRITELN('VECTOR ', SAMP2);
WRITELN('Average ', AYV := 8:4); WRITELN('Variance ', VXR := 8:4); WRITELN('Standard Deviation ', DYA := 8:4);
IF STAT = 'C' THEN
BEGIN
WRITELN;WRITELN('Sum of difference ', D1 := 8:4);
WRITELN('Sum of difference squared ', D2 := 8:4);
WRITELN('Average difference ', SIX := 8:4);
WRITELN('Std. Dev. of difference ', SD := 8:4);
END;
IF OUTPUT = 'Y' THEN READLN(RES);
WRITELN; WRITELN; WRITELN('Press RETURN to continue:');
READLN(RES);
CLRSCR;
IF (STAT = 'A') OR (STAT = 'B') THEN
WRITELN('HYPOTHESIS TEST FOR EQUAL VARIANCES')
ELSE
WRITELN('HYPOTHESIS TEST FOR PAIRED t-TEST');
IF (STAT = 'A') OR (STAT = 'B') THEN
BEGIN


WRITELN('Ho: \mu_1 = \mu_2');
WRITELN('H_1: \mu_1 ≠ \mu_2');
END
ELSE
BEGIN
WRITELN('Ho: \mu_1 = SAMP1');
WRITELN('H_1: \mu_1 ≠ SAMP2');
END
WRITELN('Selected Alpha = 0.05');
WRITELN('Calculated t-value = 2.024');
IF (STAT='C') THEN
NDF:=N[1]-1
ELSE
WRITELN('Degrees of Freedom = 4');
WRITELN('Probability of both tails = 0.05');
WRITELN('THUS: ',DECISION);
END
IF OUTPUT='Y' THEN READLN(RES);
WRITELN('Press RETURN to continue:');
READLN(RES);
END
IF (STAT='A')OR(STAT='B') THEN
BEGIN
WRITELN('HYPOTHESIS TEST FOR UNEQUAL VARIANCES:');
WRITELN('Ho: \mu_1 = \mu_2');
WRITELN('H_1: \mu_1 ≠ \mu_2');
WRITELN('Selected Alpha = 0.05');
WRITELN('Calculated t-value = 2.024');
WRITELN('Degrees of Freedom = 4');
WRITELN('Probability of both tails = 0.05');
IF OUTPUT='N' THEN
WRITELN('THUS: ',DECISION);
ELSE
WRITELN('THUS: ',DECISION);
END
END
END}
{CHI-SQUARED TEST}

PROGRAM CHI:
{MAIN PROGRAM STORAGE SPECIFICATIONS}

LABEL 15,20;

TYPE
CHARACTERSET-SET OF CHAR;
STRING2-STRING[2];
STRING3-STRING[3];
STRING4-STRING[4];
STRING60-STRING[60];
STRING13-STRING[13];

RANGE-ARRAY[1..1000] OF REAL;
RAY30-ARRAY[1..30] OF REAL;

RAY30-ARRAY[1..30] OF INTEGER;

VAR

DRIVE:STRING2;
VIEW:INTEGER;

RES,ANS,ANS2,DIS,OUTPUT:CHAR;

SI,ALP,ALPHA,A1,TAIL,XE,EX,VHCI,VCHI,SVUM,MEAN,VARI,YX1,YX2,VN,YMIN,YMAX,VXR,AXV,DATA,NCELL:REAL;

SE,CODE,DF,S2,JFR,NINT,MDTA,JX,JY,OB50,SBO,NCELL2,II,I,X,L:INTEGER;

OK:BOOLEAN;
STALP:STRING4;
ENTRY:STRING[7];
VGGOOD:STRING[8];
RESULT,STCELL,STHEAN,STVAR,Z:STRING13;

V: RANGE;

FT,YCEL,ZCEL,OUT1,OUT2:RAY30;
FRRO:IRAY30;
VECTORFILE:FILE OF REAL;

VSTAT:FILE;

VARIABLE DEFINITIONS - MAIN PROGRAM

DRIVE - Current data disk-drive.

ANS - User's response to Y/N questions.

RES,DIS - User's response to menu selections.

ANS2 - 'N' if population mean and variance are estimates.

OUTPUT - Output device.

ALP - Desired significance level entered by user.

STALP - ALP entered as a string to check validity.

ALPHA - Area to right-hand side of chi-squared statistic.

TAIL - The frequency of the final cell.

VN,YX1,YX2,A1 - Used in calculation of normally distr. expected freq.

VHCI, VCHI,S1 - Observed minus expected squared, divided by expected.

SVUM,VHCI - Calculated chi-square value.

XE,EX,YCEL,ZCEL - The expected frequencies.

OUT1 - Array of beginning of cell boundaries.

OUT2 - Array of end of cell boundaries.

FT - Expected negative exponential cumulative density function.

OB50,SBO,FREQ - The observed frequencies.

JFR,DF,S2 - Degrees of freedom.

MEAN - Population mean.

STHEAN - Population mean entered as string to check validity.

VARI - Population variance.

STVAR - Population variance entered as string to check validity.

AXV - Sample mean

VXR - Sample variance.

DATA - An element of a data vector.

NCELL2 - Number of cells (integer).

NCELL - Number of cells (real).

STCELL - Number of cells entered as string to check validity.

YMIN - Minimum value of the vector.

YMAX - Maximum value of the vector.

OK - True if file exists.

CODE - Signals error in data conversion.

I,II,III,SE - Vector array subscript.

I,MDTA - Number of elements in the vector.

JX,JY - Number of adjusted cells.

ENTRY - Signals invalid entry by user.

VGGOOD - Name of vector to be tested.
FIT,RESULT - Character string providing information about results.

HYP - String containing type of distribution to test for.

V - Vector of data.

VECTORFILE - File containing data vector of interest.

VSTAT - File containing the main menu.

CHECK ENTERED ALPHA}

PROCEDURE ALPHACK(VAR ST:STRING3; VAR ENTRY:STRING4);

PROCEDURE ALPHACK STORAGE SPECIFICATIONS

TYPE

CHARACTERSET=SET OF CHAR;

VAR

POS,COUNT:INTEGER;

POSITION:CHAR;

VALID:CHARACTERSET;

PROCEDURE ALPHACK - VARIABLE DEFINITIONS

POS - A position in a string.

COUNT - Used to count number of valid positions in a string.

POSITION - A character of a string.

VALID - A set of valid characters.

BEGIN

{ INITIALIZE COUNTER }

COUNT:=0;

{ DEFINE VALID CHARACTERSET }

VALID:="0'..9'1;

{ ENTRY MUST HAVE THREE CHARS }

IF LENGTH(STALP)=3 THEN

ENTRY:="BAD"

ELSE

BEGIN

{ CHECK FIRST POSITION OF ENTRY FOR DECIMAL }

POSITION:=COPY(STALP,1,1);

IF POSITION()'=.' THEN

ENTRY:='BAD'

ELSE

BEGIN

{ CHECK SECOND POSITION OF ENTRY }

POSITION:=COPY(STALP,2,1);

{ INCREMENT COUNTER IF CHAR IS VALID }

IF POSITION IN VALID THEN

COUNT:=COUNT+1;

{ REDEFINE VALID SET }

IF POSITION='0' THEN

VALID:="1'..9'1;

{ CHECK THIRD CHAR OF ENTRY }

POSITION:=COPY(STALP,3,1);

{ INCREMENT COUNTER IF CHAR IS VALID }

IF POSITION IN VALID THEN

COUNT:=COUNT+1;

{ ENTRY MUST HAVE 2 VALID CHARs }

IF COUNT()2 THEN

ENTRY:='BAD'

ELSE

ENTRY:='GOOD';

END;

END;

END(ALPHACK);

{ CHECK FOR INVALID ENTRIES }

PROCEDURE CHECK(VAR ST:STRING13; VAR ENTRY:STRING4);

PROCEDURE CHECK STORAGE SPECIFICATIONS

TYPE

CHARACTERSET=SET OF CHAR;

VAR

VALID1,VALID2:CHARACTERSET;

COUNT,POS:INTEGER;

POSITION:CHAR;

PROCEDURE CHECK - VARIABLE DEFINITIONS
VALID1, VALID2 - Valid sets of characters.
POS - The position in a string.
POSITION - A single character of a string.
COUNT - # of valid characters in a string.

BEGIN
{ INITIALIZE COUNTER }
COUNT:=0;
{ DEFINE VALID DATA SETS }
VALID1:=["0", "9", ";" ];
VALID2:=["0", "9", ";" ];
{ CHECK FIRST POSITION OF ENTRY }
POSITION:=COPY(ST,1,1);
IF POSITION IN VALID1 THEN
COUNT:=COUNT+1;
{ CHECK ALL OTHER POSITIONS }
FOR POS:=2 TO LENGTH(ST) DO
BEGIN
POSITION:=COPY(ST,POS,1);
IF POSITION IN VALID2 THEN
COUNT:=COUNT+1;
{ REDEFINE VALID SET WHEN DECIMAL POINT ENCOUNTERED }
IF POSITION="." THEN
VALID2:=["0", "9", ";" ];
END;
{ COMPARE # VALID POSITIONS TO LENGTH OF STRING }
IF COUNT=LENGTH(ST) THEN
ENTRY:="GOOD1";
ELSE
ENTRY:="BAD1";
END(CHECK);

{ CALCULATE MEAN & VARIANCE }
PROCEDURE MEANVAR(VAR V: RANGE; VAR K: INTEGER; VAR AXV, VX: REAL);
{ PROCEDURE MEANVAR STORAGE SPECIFICATIONS }
VAR
SX1, SX2: REAL;
L: INTEGER;

PROCEDURE MEANSTR - VARIABLE DEFINITIONS
{ SX1 - Sum of all elements in the data vector. }
{ SX2 - Sum of the square of all elements in the data vector. }
{ AXV - Average of the elements in the vector. }
{ VX - Variance of the elements in the vector. }
{ L - Counter for the array containing the vector. }
{ K - Length of the data vector. }
BEGIN
{ INITIALIZE VARIABLES }
SX1:=0; SX2:=0;
{ CALCULATE SUMATIONS }
FOR L:=1 TO K DO
BEGIN
SX1:=SX1+V[L];
SX2:=SX2+V[L]*V[L];
END;
{ CALCULATE MEAN }
AXV:=SX1/K;
{ CALCULATE VARIANCE }
VX:=(SX2-AXV*SX1)/(K-1);
END(MEANSTD);

{ CALCULATE FREQUENCIES }
PROCEDURE CFREQ(VAR V: RANGE; VAR K, NCELL2: INTEGER; VAR X1: REAL; VAR RRQ: RAY30; VAR OUT1, OUT2: RAY30
; VAR AXV, VX; YMIN, YMAX: REAL);
LABEL 5, 100
PROCEDURE CFREQ STORAGE SPECIFICATIONS }
VAR
TEMP, TSTP, YLOW, STP, YHI, A1, X1: REAL;
LAST, NUM, J, L, I: INTEGER;
PROCEDURE CALCFC - VARIABLE DEFINITION

L - Counter for array containing data vector.
I,J - Counter for array containing frequencies.
NUM - Number of data points in a vector.
FM - Number of cell for table (integer form).
TEMP - Temporary variable used for sorting data.
LAST - Vector subscript used when sorting data.
SWAP - Boolean used for sorting data.
TSTP - Calculated minimum cell width required for table.
STP - Desired cell width entered by user.
STYLOW,STYLOW - Cell width entered as string.
YLOW - First cell value for table.
YHI - Last cell value for table.
X - Used to calculate frequencies.
RES,AGAIN - User's response to Y/N question.

BEGIN
{ INITIALIZE ARRAYS }
FOR L:=1 TO 30 DO
BEGIN
FRQ[L]:=0;OUT1[L]:=O;OUT2[L]:=O;
END;
IF (ANS2='Y') THEN
{ CALL PROCEDURE TO CALCULATE MEAN AND STD. DEV. FOR SAMPLE }
MEANVAR(V,K,AKV,VXR);
{ SORT DATA (IN ASCENDING ORDER) }
SWAP:=TRUE;
LAST:=K-1;
WHILE (SWAP) DO
BEGIN
SWAP:=FALSE;
{ SORT FROM END OF VECTOR }
FOR I:=1 TO LAST DO
BEGIN
{ IF PRECEEDING NUMBER LARGER, SWAP POSITIONS }
IF (V[I])V[I+1]) THEN
BEGIN
TEMP:=V[I];
V[I]:=V[I+1];
V[I+1]:=TEMP;
SWAP:=TRUE;
END;
END;
{ EVALUATE NEXT VALUE FROM END OF VECTOR }
LAST:=LAST-1;
END;
{ OUTPUT CURRENT CELL INFORMATION }
CLRSCR;Writeln;Writeln('The number of cells required is ',NCELL2:4);
Writeln('The minimum value is ',YMIN:15:4);
Writeln('The maximum value is ',YMAX:15:4);
{ DETERMINE MIN CELL WIDTH }
TSTP:=(YMAX-YMIN)/NCELL2;
{ PROMPT USER WITH MINIMUM CELL WIDTH }
Writeln;'The above information,');
Writeln('your cell width should be greater than ',TSTP:15:4);
{ LOOP FOR RE-ENTERING CELL WIDTH }
AGAIN:='Y';
WHILE AGAIN='Y' DO
BEGIN
{ CELL WIDTH MUST BE A POSITIVE NUMBER }
STP:=0;
WHILE STP=O DO
BEGIN
{ LOOP FOR INVALID ENTRY BY USER }
ENTRY:='BAD';
END;
{ OTHERWISE CONTINUE }
END:
WHILE ENTRY='BAD' DO
BEGIN
  { CHECK FOR RETURN w/o ENTRY }
  STSTP:='';
  WHILE LENGTH(STSTP)<1 DO
    BEGIN
      { PROMPT USER TO ENTER CELL WIDTH }
      WRITELN;WRITELN;WRITELN('Enter the WIDTH you want for each cell:');
      READLN(STSTP);
    END;
    { CHECK FOR INVALID ENTRY }
    CHECK(STSTP,ENTRY);
  END;
  { CHANGE ENTRY FROM STRING TO REAL }
  VAL(STSTP,STP,CODE);
END;
{ DETERMINE FIRST CELL VALUE (YLOW) }
YLOW:=YMIN+1;
{ CHECK THAT FIRST CELL WILL INCLUDE MIN VALUE }
WHILE (YLOW)<YMIN) DO
BEGIN
  { LOOP FOR INVALID ENTRY BY USER }
  ENTRY:='BAD';
  WHILE ENTRY='BAD' DO
 BEGIN
    { CHECK FOR RETURN w/o ENTRY }
    STYLOW:='';
    WHILE LENGTH(STYLOW)<1 DO
      BEGIN
        { PROMPT USER FOR FIRST CELL-VALUE }
        WRITELN;WRITELN('Enter the first cell-value to start:');
        READLN(STYLOW);
      END;
      { CALL PROCEDURE TO CHECK FOR INVALID ENTRY }
      CHECK(STYLOW,ENTRY);
    END;
    { CHANGE FROM STRING TO REAL }
    VAL(STYLOW,YLOW,CODE);
    { COMPARE VECTOR MIN w/ FIRST CELL VALUE }
    IF (YLOW)<YMIN) THEN
    BEGIN
      { PRINT ERROR MESSAGE }
      WRITELN('Your lower limit is greater than the minimum value of your vector, ',YMIN:8:4,' !');
      WRITELN('Press any key, then TRY AGAIN !');
      READLN(RES);
    END;
    END;
    { CHECK TO INSURE ALL DATA WILL FIT }
    YHI:=YLOW+(STP*NCELL2);
    IF YHI:=YMAX THEN
    BEGIN
      { PRINT WARNING MESSAGE }
      AGAIN:='N';
      WHILE (AGAIN)'Y'AND(AGAIN)'N') DO
 BEGIN
      { PROMPT USER WITH OPTION OF RE-ENTERING CELL WIDTH & LOWER LIMIT }
      CLRSCR;WRITELN;WRITELN('With the width and lower limit you just entered,');
      WRITELN('some of your data will not be represented. ');
      WRITELN('Do you want to re-enter the width and lower limit ? (Y/N)');
      READLN(AGAIN);
    END;
    END
  ELSE
  AGAIN:='N';
END;
{ DETERMINE FIRST CELL BOUNDRY }
OUT1[1]:=YLOW;OUT2[1]:=YLOW+STP;
{ DETERMINE REMAINING CELL BOUNDARIES }
FOR L:=2 TO NCELL2 DO
BEGIN
  OUT1[L]:=OUT1[L-1]+STP;
  OUT2[L]:=OUT1[L]+STP;
END;
{ CALCULATE FREQUENCIES FOR EACH CELL }
FOR L:=1 TO K DO
BEGIN
IF (V[L]=OUT1[1]) THEN
BEGIN
SE:=1;
FRRQ[SE]:=FRRQ[SE]+1;
END
ELSE
BEGIN
FOR SE:=1 TO MCELL2 DO
IF ((V[L]=OUT1[SE]) AND (V[L]=OUT2[SE])) THEN
FRRQ[SE]:=FRRQ[SE]+1;
END;
END;
END(CALCFREQ);
{ NORMAL DISTRIBUTION }
PROCEDURE NORMAL(VAR YX1,YN: REAL);
{ PROCEDURE NORMAL STORAGE SPECIFICATIONS }
VAR
TT,D,YZ:REAL;
BEGIN
YX2:=YX1;
IF YX2<4.17 THEN
YN:=1
ELSE
IF YX2<-4.17 THEN
YN:=0
ELSE
BEGIN
YZ:=YX2;
IF YX2<0 THEN YZ:=-YX2;
TT:=Y2/1.4142142;
D:=((((0.430638E-3*TT+0.276572E-3)*TT+0.152014E-3)*TT+0.70523078E-1)*TT+0.0585923078E-1)*TT+0.705;
D:=D*D;D:=D*D;D:=D*D;
YN:=0.5-0.5*Y2;
IF YX2<0 THEN YN:=0.5-YN;
IF YX2>0 THEN YN:=0.5;
IF YX2>0 THEN YN:=0.5+YN;
END;
END(NORMAL);
{ CHI-SQUARED DISTRIBUTION }
PROCEDURE CHI(VAR S2:INTEGER; VAR S1,ALPHA:REAL);
{ PROCEDURE CHI STORAGE SPECIFICATIONS }
LABEL 10;
VAR
DIFF: INTEGER;
SE,YX,TT,JX,EE,DI,MIN,MAX:REAL;
{ VARIABLE DEFINITIONS - PROCEDURE CHI }
BEGIN
DIFF:=S2;
MAX:=178;
DI:=-2*TRUNC(DIFF/2)-DIFF+1;
IF S1>MAX THEN
MAX:=0
ELSE
MAX:=1;
MIN:=0.5*S1;
IF (DI=1) OR (DIFF>2) AND (MAX=0) THEN
BEGIN
IF MIN<10 THEN
S2:=0
ELSE
S2:=TRUNC(EXP(-MIN));
END;
END;
IF DI=1 THEN
BEGIN
  SE:=SQRT(S1);
  YX:=SE;
  NORMAL(YX,VN);
  TI:=2*VN;
END
ELSE
  TI:=S2;
  IF DFF(-2 THEN GOTO 10;
  S1:=0.5*(DFF-1);
  IF DI=1 THEN
    ZX:=1
  ELSE
    ZX:=0.5;
  IF MAX=1 THEN
  BEGIN
     IF DI=1 THEN
       EE:=0.572364942925
       ELSE
         EE:=0;
         CX:=LN(MIN);
         WHILE ZX(S1 DO
         BEGIN
           EE:=LN(ZX)+EE;
           TI:=EXP(CX*ZX-MIN-EE)+TI;
           ZX:=ZX+1;
         END;
         IF ZX(S1 THEN GOTO 10;
       END
     IF DI=1 THEN
       EE:=0.564189583548/SQRT(MIN)
     ELSE
       EE:=1;
       CX:=0;
       WHILE ZX:=S1 DO
       BEGIN
         EE:=EE*MIN/ZX;
         CX:=CX*EE;
         ZX:=ZX+1;
       END;
       ALPHA:=CX*S2+TI;
       EXIT;
       IO:ALPHA:=TI;
       END{CHI};
{ MAIN PROGRAM }
BEGIN
  ANS:="Y";
  WHILE ANS='Y' DO
  { LOOP TO REPEAT STATISTIC IF DESIRED }
BEGIN
  { DISPLAY AVAILABLE DISTRIBUTIONS }
  DIS:="0";
  WHILE (DIS='1')AND(DIS='2') DO
BEGIN
  CLS;WRITELN;WRITELN;WRITELN('CHI-SQUARE TEST');
  WRITELN(WRITELN('A Chi-square goodness-of-fit test will be performed to test how well');
  WRITELN('the sample data provided by you fits a:');WRITELN;WRITELN;
  WRITELN('1 Theoretical Normal Distribution');
  WRITELN(2 Negative Exponential Distribution');
  WRITELN(WRITELN('It is recommended that the Chi-Squared goodness-of-fit test be used for');
  WRITELN('a sample size of 50 or more data points.'););
  WRITELN(WRITELN;WRITELN("Enter the number of the distribution you desire and press RETURN:"));
  READLN(DIS);
  END;
  CASE DIS OF
    1: HYP:='Normal Distributed';
    2: HYP:='Neg. Exponentially Distributed';
  END{OF CASE};
{ DETERMINE VECTOR TO BE TESTED }
REPEAT
VGOOD := '';
WHILE (LENGTH(VGOOD) < 1 OR LENGTH(VGOOD) > 6) DO
BEGIN
  WRITELN; WRITELN; WRITELN('What is the name of the vector to be tested?');
  READLN(VGOOD);
END;
{ ATTACH DISK-DRIVE TO NAME }
VGOOD := DRIVE + VGOOD;
{ CHECK THAT VECTOR EXISTS }
ASSIGN(VECTORFILE, VGOOD);
{$I-RESET(VECTORFILE)($I+)};
OK := IOresult = 0;
CLRSCR;
{ IF NOT OK THEN }
BEGIN
  { PRINT ERROR MESSAGE }
  WRITELN('VECTOR ', VGOOD, ' DOES NOT EXIST!');
  { GIVE OPTION TO SEARCH FOR ANOTHER VECTOR OR EXIT }
  ANS := 'Y';
  WHILE (ANS('<' 'Y') AND (ANS('<' 'N')) DO
  BEGIN
    WRITELN; WRITELN('Would you like to try again to locate a vector? (Y/N)');
    READLN(ANS);
  END;
  IF ANS='N' THEN GOTO 15;
END;
UNTIL OK;
{ DETERMINE LENGTH OF VECTOR }
K := FILESIZE(VECTORFILE);
{ READ VECTOR TO ARRAY }
FOR I := 1 TO K DO
BEGIN
  READ(VECTORFILE, DATA);
  V[I] := DATA;
END;
{ CLOSE FILE }
CLOSE(VECTORFILE);
{ PROMPT USER TO ENTER MEAN & VAR OR CALCULATE }
ANS2 := 'M';
WHILE (ANS2('<' 'Y') AND (ANS2('<' 'N')) DO
BEGIN
  CLRSCR; WRITELN; WRITELN('Do you know the mean and variance of the population from which your sample was drawn? (Y/N)');
  READLN(ANS2);
END;
{ IF ANS2='Y' THEN }
BEGIN
  { LOOP FOR INVALID ENTRY BY USER }
  ENTRY := 'BAD';
  WHILE ENTRY='BAD' DO
  BEGIN
    CLRSCR; WRITELN; WRITELN('Please enter the following information concerning the population');
    WRITELN('from which your sample was drawn.');
    { CHECK FOR RETURN W/O ENTRY }
    STMEAN := '';
    WHILE LENGTH(STMEAN) < 1 DO
    BEGIN
      { PROMPT USER TO ENTER SAMPLE MEAN }
      WRITELN; WRITELN('Mean (mu) = ');
      READLN(STMEAN);
    END;
    { CHECK ENTRY FOR VALIDITY }
    CHECK(STMEAN, ENTRY);
    END;
    { CHANGE STRING TO REAL }
    VAL(STMEAN, MEAN, CODE);
    { VARIANCE MUST BE POSITIVE }
    VARI := 0;
    WHILE (VARI<=0) DO
    BEGIN
      { LOOP FOR INVALID ENTRY BY USER }
      ENTRY := 'BAD';
    END;
WHILE ENTRY='BAD' DO
BEGIN
{ CHECK FOR RETURN W/O ENTRY }
STVARI:='';
WHILE LENGTH(STVARI)<1 DO BEGIN
{ PROMPT USER TO ENTER SAMPLE VARIANCE }
WRITELN;WRITELN('Variance (sigma-squared) = '); READLN(STVARI);
END;
{ CHECK ENTRY FOR VARIANCE }
CHECK(STVARI,ENTRY);
END;
{ CHANGE STRING TO REAL }
VAL(STVARI,VARI,CODE);
END;
ELSE
BEGIN
{ CALL PROCEDURE TO CALCULATE MEAN AND VAR (USE SAMPLE AS ESTIMATE OF POP)}
MEANVAR(V,X,AXV,VXR);  
{ USE SAMPLE MEAN AND VAR AS BEST ESTIMATE OF POPULATION }
MEAN:=AXV;VAR:=VXR;
END;
{ PROVIDE OPTION OF HAVING # CELLS CALCULATED OR ENTERED BY USER }
ANS:='N';
WHILE (ANS='Y')AND(ANS='N')DO
BEGIN
CLRSER;WRITELN;WRITELN('Do you want the program to calculate the number of cells required ? (Y/N)');
READLN(ANS);
IF ANS='N' THEN
{ USER ENTERS # OF CELLS }
BEGIN
NCELL2:=0;
{ NUMBER OF CELLS MUST BE BETWEEN 3 AND 30 }
WHILE (NCELL2<3)OR(NCELL2>30) DO BEGIN
{ LOOP FOR INVALID ENTRY BY USER }
ENTRY='BAD';
WHILE ENTRY='BAD' DO
BEGIN
{ CHECK FOR RETURN W/O ENTRY }
STCELL:='';
WHILE LENGTH(STCELL)<1 DO BEGIN
{ PROMPT USER FOR # OF CELLS }
WRITELN;WRITELN;WRITELN('Enter the number of cells (min=3, max=30):'); READLN(STCELL);
END;
{ CALL PROCEDURE TO CHECK FOR INVALID ENTRY }
CHECK(STCELL,ENTRY);
END;
{ CHANGE ENTRY FROM STRING TO REAL }
VAL(STCELL,NCELL,CODE);
NCELL:=ABS(NCELL);  
{ CHANGE FROM REAL TO INTEGER }
NCELL2:=TRUNC(NCELL);
END;
ELSE
IF ANS='Y' THEN
{ CALCULATE THE NUMBER OF CELLS (# CELLS = SQRT(K) ) }
BEGIN
NCELL:=SQRT(K)+0.5;
{ CHANGE REAL TO INTEGER }
NCELL2:=TRUNC(NCELL);
IF NCELL2=3 THEN NCELL2:=30;
{ PRINT ERROR MESSAGE IF NUMBER OF CELLS < 3 }
IF NCELL2<3 THEN
BEGIN
CLRSER;WRITELN;WRITELN('The vector does not contain enough data points to perform the test.');
END;
WRITELN;WRITELN('Press RETURN to exit:');
READLM(RES);GOTO 15;
END;
END;
{ DETERMINE VECTOR MIN AND MAX VALUES }
YMIN:=V[1];YMAX:=V[1];
FOR L:=2 TO K DO BEGIN
IF V[L]<YMIN THEN YMIN:=V[L];
IF V[L]>YMAX THEN YMAX:=V[L];
IF L=(NCELL2+1) THEN FRQ[L-1]:=0;
END;
{ CALL PROCEDURE TO CALCULATE FREQUENCY }
CFREQ(V,K,NCELL2,ANS,FRQ,OUT1,OUT2,AXV,VXR,YMIN,YMAX);
{ CHECK FOR ALL IDENTICAL ENTRIES }
IF (DIS='1')OR(DIS='2') THEN BEGIN
IF (YMIN-YMAX)=0 THEN BEGIN
PRINT ERROR MESSAGE
CLRSCR;WRITELN;WRITELN('ALL THE VALUES IN THIS VECTOR ARE THE SAME !');
WRITELN;WRITELN('Press RETURN to exit:');
READLM(RES);GOTO 15;
END;
END;
{ INITIALIZE ARRAYS }
FOR L:=1 TO 30 DO BEGIN
YCEL[L]:=0;ZCEL[L]:=0;
END;
{ CALCULATED EXPECTED FREQUENCIES }
CASE DIS OF
'1':BEGIN { FOR NORMAL DISTRIBUTION }
FOR L:=1 TO NCELL2 DO BEGIN
A1:=(OUT2[L]-MEAN)/SQRT(VARI);
YX1:=A1;
{ CALL NORMAL DISTRIBUTION PROCEDURE }
NORMAL(YX1,VN);
YCEL[L]:=VN;
IF L=1 THEN
YCEL[1]:=VN*A
ELSE
YCEL[L]:=(ZCEL[L]-ZCEL[L-1])*K;
END;
YCEL[NCELL2]:=YCEL[NCELL2]+K*(1-ZCEL[NCELL2]);
END;
'2':BEGIN { NEGATIVE EXPONENTIAL DISTRIBUTION }
FOR L:=1 TO NCELL2 DO BEGIN
FT[L]:=(1-EXP((-OUT2[L])/AXV));
{ EXPE. CUMULATIVE DISTR. FUNCTION }
IF L=1 THEN
YCEL[L]:=FT[L]
ELSE
YCEL[L]:=FT[L]-FT[L-1];
{ EXPECTED FREQUENCY }
YCEL[L]:=YCEL[L]*K;
END;
END;
END OF CASE;
JX:=NCELL2;
20: JY:=JX;
{ CHECK THAT EXPECTED FREQUENCIES FOR EACH CELL }=5 }
FOR I:=1 TO JY DO BEGIN
IF (YCEL[I]<5) THEN BEGIN
ADJUST CELLS IF CONTENTS <5 }
IF I=JX THEN BEGIN
END;
{ MODIFY_FINAL_CELL }
YCEL[1]:=YCEL[1]+YCEL[1];
FRQQ[1]:=FRQQ[1]+FRQQ[1];
OUT2[1]:=OUT2[1];
JX:=JX-1;
GOTO 20;
END
ELSE
BEGIN
{ MODIFY_OTHER.Cells }
YCEL[1]:=YCEL[1]+YCEL[1];
FRQQ[1]:=FRQQ[1]+FRQQ[1];
OUT2[1]:=OUT2[1];
JX:=JX-1;
GOTO 20;
END;
END;
END;
END;
END;
END;
END;
{ CHECK IF # OF CELLS WAS ALTERED }
IF NCELL2/JX THEN
BEGIN
{ INFORM USER THAT # OF CELLS WAS ALTERED }
CLRSCR;WRITELN;WRITELN(NCELL2,' cells were specified, however, the Chi-Square test requires');
WRITELN('that the expected frequency in each cell is at least five. To meet');
WRITELN('this requirement, the number of cells has been reduced to ',JX,'.');
WRITELN('Press RETURN to continue:');
READLN(RES);
END;
{ DETERMINE DEGREES OF FREEDOM }
IF ANS2:'Y' THEN DF:=JY-1;
IF ANS2:'N' THEN DF:=JY-3;
BEGIN
{ CHECK FOR DEGREES OF FREEDOM ERROR }
IF DF<=0 THEN
BEGIN
{ PRINT ERROR MESSAGE }
CLRSCR;WRITELN;WRITELN('There is insufficient information to perform this test.');
WRITELN('Press RETURN to exit:');
READLN(RES);GOTO IS;
END;
{ INITIALIZE SUM }
SVUM:=0;
{ CALCULATE CHI-SQUARE VALUE }
FOR L:=1 TO JY DO
BEGIN
OBS:=FRQQ[L];
EX:=YCEL[L];
YCHI:=(OBS-EX)*((OBS-EX))/EX;
SVUM:=SVUM+YCHI;
END;
{ ASSIGN ADDITIONAL VARIABLE NAMES }
VCHI:=SVUM;FR:=DF;
S1:=SVUM;S2:=DF;
{ CALL CHI-SQUARE PROCEDURE TO CALCULATE ACTUAL ALPHA }
CHI(S2,S1,ALPHA);
ALPHA:=ALPHA2;
{ LOOP FOR INVALID ENTRY BY USER }
ENTRY:='BAD';
WHILE ENTRY='BAD' DO
BEGIN
{ PROMPT USER TO ENTER DESIRED SIGNIFICANCE LEVEL }
Enter the desired ALPHA level in decimal form for the significance test.

Example: 5% is entered as .05; 10% is entered as .10.

Check ALPHA level entered by user

Convert string to real

Select output device

While (output = 'Y') AND (output = 'N') do

Clear screen; write 'Do you want a HARD COPY of the results? (Y/N)?'; readln(output);

If output = 'N' then

Clear screen; write 'Chi-square goodness-of-fit Test on VECTOR ', vgood; writeln; writeln('CELL CELL BOUNDARIES OBSERVED EXPECTED (Ob.-Ex.)**2/Ex.'); writeln;

Else

Clear screen; writeln('Chi-Square goodness-of-fit Test on VECTOR ', vgood); writeln('CELL CELL BOUNDARIES OBSERVED EXPECTED (Ob.-Ex.)**2/Ex.'); writeln;

For l := 1 to j do

Clear screen; writeln('Press RETURN to continue with hypothesis testing:'); readln(res);

Display results of hypothesis test

If output = 'N' then

Clear screen; writeln('Statistical hypothesis test'); writeln('Ho: Data in vector ', vgood, ' is ', hyp.'); writeln('Number of observations = ', k:4); writeln('Selected Alpha = ', alp:8:4); writeln('Calculated Chi-Square value = ', vchi:8:4); writeln('Degrees of Freedom = ', df:4); writeln('Prob.(chi-sq.) = ', vchi:0:4, ', ', df:4, ') = ', alp:8:4);

Else

Clear screen; writeln('Statistical hypothesis test'); writeln('Ho: Data in vector ', vgood, ' is ', hyp.'); writeln('Number of observations = ', k:4); writeln('Selected Alpha = ', alp:8:4); writeln('Calculated Chi-Square value = ', vchi:8:4);
WRITE(LST, 'Degrees of Freedom', DF:4);
WRITE(LST, 'Prob.(chi-sq.) = ', VCHI:8:4, ', ALPHA = ', ALPHA:8:4);

{ DETERMINE RESULT OF HYPOTHESIS TEST }
IF ALPHA THEN BEGIN
  RESULT := 'REJECT Ho';
  FIT := 'is NOT';
END ELSE BEGIN
  RESULT := 'ACCEPT Ho';
  FIT := 'is';
END;

IF OUTPUT = 'N' THEN BEGIN
  WRITELN(LST); WRITELN(LST);
  WRITELN('Thus; ', RESULT, ', Vector ', VG00D, ', FIT; ', HYP);
END ELSE BEGIN
  WRITELN(LST); WRITELN(LST);
  WRITELN(RESULT, ', Vector ', VG00D, ', FIT; ', HYP);
END;

{ PERFORM ANOTHER TEST ? }
ANS := 'N';
WHILE (ANS = 'Y') AND (ANS = 'N') DO BEGIN
  WRITELN(WRITELN(LST)); WRITELN('Would you like to perform another Chi-Square test? (Y/N)');
  READLN(ANS);
END;

IS := '{ RETURN TO MAIN MENU }
BEGIN
  ASSIGN(VSTAT, 'VSTAT.COM');
  EXECUTE(VSTAT);
PROGRAM ISI;
{ MAIN PROGRAM STORAGE SPECIFICATIONS }
LABEL 15,20;
TYPE
   CHARACTERSET=SET OF CHAR;
   STRING2=STRING[2];
   STRING3=STRING[3];
   STRING4=STRING[4];
   STRING60=STRING[60];
   STRING13=STRING[13];
   RANGE=ARRAY[1.1000] OF REAL;
   RAY30=ARRAY[1.30] OF REAL;
VAR
   DRIVE:STRING2;
   VIEW:INTEGER;
   RES,ANS,ANS2,DIS,OUTPUT,POSITION:CHAR;
   D2EV,ALPHA,AIL,TAIL,EX,EXH,ECH1,CHV1,CHV,EVH1,CHV1,SVUM,MEAN,VARI,YX,VM,SMI,YMN,YMX,XY,AXY,DATA,NCELL,
   L,RFRRQ,CFRRQ:REAL;
   DELTA,MAX,MINF,KS,S2,CODE,JFR,DF,II,NDTA,JX,JY,OB8,S80,NCELL2,II,IV, INTEGER;
   OK:BOOLEAN;
   STALP:STRING3;
   STS2,STS3,ENTRY,OKAY:STRING4;
   FTT,KOLI:STRING[8];
   RESULT,STELL,STMEAN,STVAR,STS1:STRING13;
   HYP:STRING[30];
   V:RANGE;
   FTT,DEV,CICEL,ZUM,FT,YCEL,ZCEL,OUT1,OUT2:RAY30;
   RFRRQ:RAY30;
   VALID1,VALID2:CHARACTERSET;
   VECTORFILE:FILE OF REAL;
   VSTAT:FILE;

VARIABLE DEFINITIONS - MAIN MENU

DRIVE - Current disk drive for data.
ANS - User's response to Y/N questions.
RES - User's response to menu selections.
ALPHA - Area to right-hand side of chi-squared statistic.
KS,S2 - Area to right-hand side of Z, F and t statistics.
YX,VM - Used in calculation of area to right of Z-statistic.
SI - Statistic entered by user to be evaluated.
S2 - Degrees of freedom (DF of numerator for F statistic).
S3 - Degrees of freedom of denominator for F statistic.
CODE - Signals error in data conversion.
STS1 - Statistic entered as string by user.
STS2,STS3 - Degrees of freedom entered as strings.
ERR,ENTRY,OKAY - Strings used to signal invalid entry by user.
DD - The number corresponding to the statistic selected by the user.
VSTAT - File containing the main menu.
OUTPUT - 'Y' if hard copy desired by user.
POSITION - A single character in a string.
POS - Position in a string.
NCELL2 - Number of cells (integer).
NCELL - Number of cells (real).
L,I,X - Vector array subscripts.
YMIN - Minimum value of the vector.
YMIX - Maximum value of the vector.
DATA - Elements in the vector.
FRQQ - Frequency of a value.
CFRQQ - Cumulative frequency.
RFRRQ - Relative frequency.
RFRQQ - Relative cumulative frequency.

{ CHECK ALPHA }

PROCEDURE ALPHACK(VAR STALP:STRING3; VAR ENTRY:STRING4);
TYPE
   CHARACTERSET=SET OF CHAR;
VAR
   POS,COUNT:INTEGER;
   POSITION:CHAR;
   VALID:CHARACTERSET;

{ ... }
BEGIN
  COUNT:=0;
  VALID:=[10',9'];
  IF LENGTH(STALP)>3 THEN
    ENTRY:='BAD'
  ELSE
    BEGIN
      POSITION:=COPY(STALP,1,1);
      IF POSITION>9 THEN
        ENTRY:='BAD'
      ELSE
        BEGIN
          POSITION:=COPY(STALP,2,1);
          IF POSITION IN VALID THEN
            COUNT:=COUNT+1;
            IF POSITION='0' THEN
              VALID:=[1,9]
            END;
          END;
          IF COUNT>2 THEN
            ENTRY:='BAD'
          ELSE
            BEGIN
              POSITION:=COPY(STALP,3,1);
              IF POSITION IN VALID THEN
                COUNT:=COUNT+1;
            END;
          END;
        END;
    END;
END;
END;{ALPHABETIC};
{CHECK FOR INVALID ENTRIES}
PROCEDURE CHECK(VAR ST:STRING13; VAR ENTRY:STRING4);
TYPE
  CHARACTERS=SET OF CHAR;
VAR
  VALID1,VALID2:CHARACTERS;
  COUNT,POS:INTEGER;
  POSITION:CHAR;
BEGIN
  {initialize counter}
  COUNT:=0;
  {define valid data sets}
  VALID1:=[10',9',.1'];
  VALID2:=[1',9'];
  {check first position of entry}
  POSITION:=COPY(ST,1,1);
  IF POSITION IN VALID1 THEN
    COUNT:=COUNT+1;
  {check all other positions}
  FOR POS:=2 TO LENGTH(ST) DO
    BEGIN
      POSITION:=COPY(ST,POS,1);
      IF POSITION IN VALID2 THEN
        COUNT:=COUNT+1;
      {redefine valid set when decimal point encountered}
      IF POSITION='.' THEN
        VALID2:=[1',9'];
    END;
  {compare # valid positions to length of string}
  IF COUNT>LENGTH(ST) THEN
    ENTRY:='BAD'
  ELSE
    ENTRY:='GOOD';
END(CHECK);
{CALCULATE MEAN & VARIANCE}
PROCEDURE MEANVAR(VAR V:RANGE; VAR K:INTEGER; VAR YMIN,YMAX,AXV,VXR:REAL);
VAR
  SX1,SX2:REAL;
PROCEDURE MEANVAR - VARIABLE DEFINITIONS

SX1 - Sum of all elements in the data vector.
SX2 - Sum of the square of all elements in the data vector.
AXV - Average of the elements in the vector.
VXR - Variance of the elements in the vector.
L - Counter for the array containing the vector.
K - Length of the data vector.
YMIN - Minimum value of the vector.
YMAX - Maximum value of the vector.

BEGIN
{ INITIALIZE VARIABLES }
SX1 := 0; SX2 := 0;
YMIN := V[1]; YMAX := V[1];
FOR L := 1 TO K DO
BEGIN
SX1 := SX1 + V[L];
SX2 := SX2 + V[L] * V[L];
IF V[L] < YMIN THEN YMIN := V[L];
IF V[L] > YMAX THEN YMAX := V[L];
END;
{ CALCULATE MEAN }
AXV := SX1 / K;
{ CALCULATE VARIANCE }
VXR := (SX2 - AXV * SX1) / (K - 1);
END(MEANVAR);

PROCEDURE MEANVAR - VARIABLE DEFINITIONS

BEGIN
{ INITIALIZE ARRAYS }
FOR L := 1 TO NCELL2 DO
BEGIN
FREQ[L] := 0; OUT1[L] := 0; OUT2[L] := 0;
END;
IF (ANS2 = 'Y') THEN
{ CALL PROCEDURE TO CALCULATE MEAN AND STD. DEV. FOR SAMPLE }
MEANVAR(V, K, YMIN, YMAX, AXV, VXR);
{ SORT DATA ( IN ASCENDING ORDER ) }
SWAP := TRUE;
LAST := K;
WHILE (SWAP) DO
BEGIN
SWAP := FALSE;
FOR I := 1 TO LAST DO
BEGIN
IF (V[I]=V[I+1]) THEN
BEGIN
    TEMP:=V[I];
    V[I]:=V[I+1];
    V[I+1]:=TEMP;
    SNAP:=TRUE;
END;
END;
LAST:=LAST-1;
END;

{ OUTPUT CURRENT CELL INFORMATION }
CLRSCR;WRITELN;WRITELN('The number of cells required is ',NCCELL2:4);
WRITELN;WRITELN('The minimum value is ',YMIN:13:4);
WRITELN;WRITELN('The maximum value is ',YMAX:13:4);
{ DETERMINE MIN CELL WIDTH }
STP:=(YMAX-YMIN)/NCCELL2;
{ PROMPT USER WITH MINIMUM CELL WIDTH }
WRITELN;WRITELN;WRITELN('Given the above information,');
IF DIS='2' THEN
BEGIN
    STP:=STP+0.05;
    WRITELN('the specified cell width is ',STP:13:4);
    WRITELN;WRITELN;WRITELN('Press RETURN to continue:');
    READLN(RES);
    IF DIS='2' THEN
    YLOW:=YMIN-STP;
ELSE
BEGIN
    WRITELN('your cell width should be greater than ',STP:13:4);
    { LOOP FOR RE-ENTERING CELL WIDTH }
AGAIN:="Y";
WHILE AGAIN="Y" DO
BEGIN
    STP:=0;
    WHILE STP<=0 DO
BEGIN
    ENTRY:="BAD";
    WHILE ENTRY="BAD" DO
BEGIN
    STSTP:="-";
    WHILE LENGTH(STSTP)<1 DO
BEGIN
    { PROMPT USER TO ENTER CELL WIDTH }
    WRITELN;WRITELN;WRITELN('Enter the WIDTH you want for each cell:');
    READLN(STSTP);
    END;
    { CHECK FOR INVALID ENTRY }
    CHECK(STSTP,ENTRY);
    END;
    { CHANGE ENTRY FROM STRING TO REAL }
    VAL(STSTP,STP,CODE);
    END;
    { DETERMINE FIRST CELL VALUE (YLOW) }
    YLOW:=YMIN+1;
    WHILE (YLOW)<YMIN) DO
BEGIN
    ENTRY:="BAD";
    WHILE ENTRY="BAD" DO
BEGIN
    { PROMPT USER FOR FIRST CELL VALUE }
    WRITELN;WRITELN('Enter the first cell value to start:');
    READLN(STYLOW);
    { CALL PROCEDURE TO CHECK ENTRY }
    CHECK(STYLOW,ENTRY);
    END;
    { CHANGE FROM STRING TO REAL }
    VAL(STYLOW,YLOW,CODE);
    { COMPARE VECTOR MIN W/ FIRST CELL VALUE }
    IF (YLOW)<YMIN) THEN
BEGIN
    { PRINT ERROR MESSAGE }
WRITELN('Your lower limit is greater than the minimum value of your vector, ",YMIN:8:4,"');
WRITELN('Press any key, then TRY AGAIN!');
READLN(RES);
END;
{ CHECK TO INSURE ALL DATA WILL FIT }
YHI:=YLO+STP*NCELL2;
IF YHI<=YMAX THEN
BEGIN
AGAIN:='Y';
WHILE (AGAIN='Y')AND(AGAIN='N') DO
BEGIN
{ PROMPT USER WITH OPTION OF RE-ENTERING CELL WIDTH & LOWER LIMIT }
CLRSCR;WRITELN;WRITELN('With the width and lower limit you just entered,');
WRITELN('some of your data will not be represented.');
WRITELN;'Do you want to re-enter the width and lower limit ? (Y/N)');
READLN(AGAIN);
END;
ELSE
AGAIN:='N';
END;
END;
{ DETERMINE TABLE ENTRIES }
J:=1;OUT1[1]:=YLO;NUM:=N;
FOR L:=1 TO NCELL2 DO
BEGIN
OUT2[L]:=OUT1[L]*STP;
J:=J+1;
IF J=30 THEN GOTO 10;
OUT1[J]:=OUT2[J-1];
END;
10:j:=1;
FOR L:=1 TO NUM DO
BEGIN
FRQ[J]:=FRQ[J]+1
ELSE
BEGIN
J:=J+1;
IF J=30 THEN EXIT;
GOTO 5;
END;
END;
END(CALCFREQ);
{ NORMAL DISTRIBUTION }
PROCEDURE NORMAL(VAR YX,VN:REAL);
{ PROCEDURE NORMAL STORAGE SPECIFICATIONS }
VAR
TT,D,YZ:REAL;
BEGIN
YX:=-YX;
IF YX>4.17 THEN
VN:=1
ELSE
IF YX<-4.17 THEN
VN:=0
ELSE
BEGIN
Y2:=YX;
IF YX<0 THEN Y2:=-YX;
T1:=YX/1.141421;
D:=((0.436536*Y1+0.2765672E-3)*T1+0.1520141E3)*T1+0.92705272E-2)*T1+0.4220212E-1)*T1+0.70523078E-1)*T1+1.0;
D:=D+D*D:D:=D*D:D:=D*D;
VN:=0.5-D/D;
IF YX<0 THEN VN:=0.5-VN;
IF YX>0 THEN VN:=0.5;
IF YX=0 THEN VN:=0.5+VN;
END;
END(NORMAL);
{ MAIN PROGRAM }
BEGIN
ANS:= 'Y';
WHILE ANS='Y' DO { LOOP TO REPEAT STATISTIC IF DESIRED }
BEGIN
DIS:= '0';
WHILE (DIS('1')) AND (DIS('2')) DO
BEGIN
CLSCHR;WRITELN;WRITELN;WRITELN('KOLMOGOROV-SMIRNOV');
WRITELN('A Kolmogorov-Smirnov One Sample test will be performed to determine how');
WRITELN('well the sample data provided by you fits a');WRITELN('Normal Distribution');
WRITELN('Negative Exponential Distribution');
WRITELN('Enter the number of the distribution you desire and press RETURN:');
READLN(DIS);
END;
CASE DIS OF
'1': HYP:= 'Normally Distributed';
'2': HYP:= 'Neg. Exponentially Distributed';
END OF CASE;
REPEAT
KOL:= '';
WHILE (LENGTH(KOL)<1) OR (LENGTH(KOL)>6) DO
BEGIN
WRITELN;WRITELN('What is the name of the vector to be tested?');
READLN(KOL);
END;
{ ASSIGN DISK-DRIVE TO NAME }
KOL:= DRIVE+KOL;
{ ATTEMPT TO LOCATE FILE }
ASSIGN(VECTORFILE,KOL);
{ (I)RESET(VECTORFILE)\{I+1\};
OK:= (I)RESULT=0;
CLSCHR;
IF NOT OK THEN
BEGIN
WRITELN('VECTOR ',KOL,' DOES NOT EXIST !');
ANS:= 'M';
WHILE (ANS('Y')) AND (ANS('N')) DO
BEGIN
WRITELN('Would you like to try again to locate a vector? (Y/N)');
READLN(ANS);
END;
IF ANS='N' THEN GOTO 15;
END;
UNTIL OK;
{ DETERMINE LENGTH OF VECTOR }
K:= FILES\{VECTORFILE\};
{ READ VECTOR TO ARRAY }
FOR I:= 1 TO K DO
BEGIN
READ(VECTORFILE,DATA);
V[I]:= DATA;
END;
{ CLOSE FILE }
CLOSE(VECTORFILE);
{ PROMPT USER TO ENTER MEAN & VAR OR CALCULATE }
ANS2:= 'M';
WHILE (ANS2('Y')) AND (ANS2('N')) DO
BEGIN
WRITELN;WRITELN('Do you know the mean and variance of the population from which your');
WRITELN('sample was drawn? (Y/N)');
READLN(ANS2);
END;
IF ANS2='Y' THEN
BEGIN
ENTRY:= 'BAD';
WHILE ENTRY='BAD' DO
BEGIN
CLSCHR;WRITELN;WRITELN('Please enter the following information concerning the population');
WRITELN('from which your sample was drawn:');
STEAM:= '';
WHILE LENGTH(STMEAN) > 1 DO
  BEGIN
    WRITEW; WRITELN(' Mean (\mu) = ');
    READLN(STMEAN);
  END;
  { CHECK ENTRY FOR MEAN }
  CHECK(STMEAN,ENTRY);
END;
{ CHANGE STRING TO REAL }
VAL(STMEAN,MEAN,CODE);
VAR1:=0;
WHILE (VAR1=0) DO
  BEGIN
    ENTRY:= 'BAD';
    WHILE ENTRY=: 'BAD' DO
      BEGIN
        ENTRY:= 'BAD';
        WHILE LENGTH(STVAR1) > 1 DO
          BEGIN
            WRITEW; WRITELN(' Variance (\sigma^2) = ');
            READLN(STVAR1);
          END;
          { CHECK ENTRY FOR VARIANCE }
          CHECK(STVAR1,ENTRY);
        END;
        { CHANGE STRING TO REAL }
        VAL(STVAR1,VAR1,CODE);
      END;
END ELSE BEGIN
  { CALL PROCEDURE TO CALCULATE MEAN AND VARIANCE (USE SAMPLE AS ESTIMATE OF POPULATION) }
  MEANVAR(V,XMIN,YMAX,AXV,VXR);
  { SAMPLE MEAN AND VARIANCE ARE BEST ESTIMATE OF POPULATION }
  MEAN:=AXV; VAR1:=VXR;
END;
{ PROVIDE OPTION OF HAVING # CELLS CALCULATED OR ENTERED BY USER }
IF (DIS='I') THEN
  BEGIN
    ANS:= 'Y';
    WHILE (ANS='Y') AND (ANS='N') DO
      BEGIN
        CLRSCR; WRITELN; WRITELN('Do you want the program to calculate the number of cells required? (Y/N)');
        READLN(ANS);
      END;
    IF ANS='Y' THEN
      BEGIN
        NCELL2:=0;
        WHILE (NCCELL2<3) OR (NCCELL2>30) DO
          BEGIN
            ENTRY:= 'BAD';
            WHILE ENTRY=: 'BAD' DO
              BEGIN
                { PROMPT USER FOR # OF CELLS }
                WRITEW; WRITELN; WRITELN('Enter the number of cells (min=3, max=30): ');
                READLN(STCELL);
                { CALL PROCEDURE TO CHECK FOR INVALID ENTRY }
                CHECK(STCELL,ENTRY);
              END;
              { CHANGE ENTRY FROM STRING TO REAL }
              VAL(STCELL,NCCELL,CODE);
              NCELL2: =ABS(NCCELL);
              { CHANGE FROM REAL TO INTEGER }
              NCELL2:=TRUNC(NCCELL);
            END;
          END ELSE
            IF ANS='Y' THEN
              BEGIN
                NCELL2:= SQRT(K)+0.5;
NCELL2:=TRUNC(NCELL);
IF NCELL>30 THEN NCELL2:=30;
IF NCELL2<3 THEN BEGIN
  CLRSCR;WRITELN;WRITELN('The vector does not contain enough data points to perform the test.');
  WRITELN('Press RETURN to exit.');
  READLN(RES);GOTO 15;
END;

IF DIS='2' THEN BEGIN
  NCELL2:=K;
  IF NCELL2>30 THEN NCELL2:=30;
END;

{ CALL PROCEDURE TO CALCULATE FREQUENCY }
CFREQ(V,K,NCELL2,DIS,FRQQ,OUT1,OUT2,AXV,VXR);
{ CALCULATE MAX AND MIN FREQUENCIES }
MIN:=FRQ0[1];MAXF:=FRQ0[1];
FOR L:=2 TO NCELL2 DO BEGIN
  IF FRQ0[L](MIN THEN MIN:=FRQ0[L];
  IF FRQ0[L]MAX THEN MAXF:=FRQ0[L];
END;

{ INITIALIZE ARRAYS }
FOR L:=1 TO 30 DO BEGIN
  FT[L]:=0;DEV[L]:=0;CICE[L]:=0;ZUCH[L]:=0;FT[L]:=0;YCEL[L]:=0;ZCEL[L]:=0;
END;

{ CALCULATE THEORETICAL FREQ. FOR APPROPRIATE DISTRIBUTION }
CASE DIS OF
  '1':BEGIN { NORMAL DISTRIBUTION }
    A1:=(OUT2[1]-MEAN)/SORT(VARI);
    VX:=-A1;
    NORMAL(VX,VN);
    ZCEL[L]:=vn;
    YCEL[L]:=YC[L];
    FOR L:=2 TO NCELL2 DO BEGIN
      A1:=(OUT2[L]-MEAN)/SORT(VARI);
      VX:=-A1;
      NORMAL(VX,VN);
      ZCEL[L]:=vn;
      YCEL[L]:=YCEL[L]-ZCEL[L-1];
      A1:=0;
      JX:=NCELL2;MDTA:=K;
      OUT1[I]:=9999.9999;
      FOR L:=I TO JX DO BEGIN
        CICEL[L]:=FRQ0[L]/MDTA;
        ZUCH[L]:=CICEL[L];
        FOR L:=2 TO JX DO BEGIN
          KS:=L-1;
          ZUCH[L]:=CICEL[L]+ZUCH[KS];
        END;
      END;
    END;
  '2':BEGIN { NEG. EXPONENTIAL DISTR. }
    FOR L:=1 TO NCELL2 DO BEGIN
      EXP. CUM. DISTR. FUNCTION }
      FT[L]:=(1-EXP((-OUT2[L])/AXV));
      { EXPECTED FREQUENCY }
      YCEL[L]:=FT[L];
      ZCEL[L]:=YCEL[L];
    END;
  '3':BEGIN { OBSERVED CUM. DISTR. FUNCTION }
    FOR L:=1 TO NCELL2 DO BEGIN
      ZUCH[L]:=L/(K*1);
      JX:=NCELL2;
END:
END(CASE OF):
{ CALCULATE DEVIATION OF OBSERVED FROM EXPECTED }
DDEV:=ABS(ZCEL[1]-ZUCH[1]);
DEV[1]:=DDEV;
FOR L:=2 TO JX DO
BEGIN
DEV[L]:=ABS(ZCEL[L]-ZUCH[L]);
IF (DEV[L])=DDEV THEN
DDEV:=DEV[L];
END;
{ LOOP FOR INVALID ALPHA ENTRY }
ENTRY:=''BAD'';
WHILE ENTRY=''BAD'' DO
BEGIN
{ PROMPT USER TO INPUT DESIRED SIGNIFICANCE LEVEL }
CLRSR;WRITELN;WRITELN('Enter the desired alpha level ( 0 < ALPHA < 1) in decimal form.');
WRITELN('Note: 5% is entered as .05;');
WRITELN('10% entered as .10');
READLN(STALP);
{ CHECK ALPHA }
ALPHACK(STALP,ENTRY);
END;
{ CONVERT STRING TO REAL }
VAL(STALP,ALP,CODE);
{ SELECT OUTPUT DEVICE }
OUTPUT:='Y';
WHILE (OUTPUT='Y')AND(OUTPUT='N') DO
BEGIN
CLRSR;WRITELN;WRITELN('Do you want a HARD COPY of the results ? (Y/N)');
READLN(OUTPUT);
END;
IF OUTPUT='N' THEN
BEGIN
CLRSR;WRITELN;WRITELN('KOLMOGOROV-SMIRNOV ONE SAMPLE TEST ON VECTOR ',KOL1);WRITELN;WRITELN;
WRITELN('CELL RANGE FREQ. CUM. CUM. KOL-SMI');
WRITELN(' From To OBS. EXP. STAT. ');WRITELN;
END;
ELSE
WRITELN(LST);WRITELN(LST,' KOLMOGOROV-SMIRNOV ONE SAMPLE TEST ON VECTOR ',KOL1);WRITELN(LST);WRITELN(LST);
WRITELN(LST,' KOLMOGOROV-SMIRNOV ONE SAMPLE TEST ON VECTOR ',KOL1);WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST);
WRITELN(LST,' KOLMOGOROV-SMIRNOV ONE SAMPLE TEST ON VECTOR ',KOL1);
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITE ('K:4');
ALPHA:=SORT(-LN(ALP/2)/(2*K));
WRITELN('Press RETURN to continue with hypothesis testing.');
READLN(RES);
IF OUTPUT='N' THEN
BEGIN
CLRSR;WRITELN;WRITELN('HYPOTHESIS TEST');
WRITELN('HI: Data in vector ',KOL1,' is ',HYP,');
WRITELN('Ho: Data in vector ',KOL1,' is NOT ',HYP,');WRITELN;
WRITELN('Number of Observations = ',X:4);
WRITELN('Selected Alpha = ',ALPHA:8:4);
WRITELN('Maximum Absolute Difference = ',DDEV:8:4);
WRITELN('Degrees of Freedom = ',X:4);
WRITELN('K-S Prob. two-tailed value = ',ALPHA:8:4);
END;
ELSE
BEGIN
WRITELN(LST);WRITELN(LST);WRITELN(LST,' KOLMOGOROV-SMIRNOV ONE SAMPLE TEST ON VECTOR ',KOL1);WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITELN(LST)
WRITE ('K:4');
ALPHA:=SORT(-LN(ALP/2)/(2*K));
WRITELN('Press RETURN to continue with hypothesis testing.');
READLN(RES);
IF OUTPUT='N' THEN
BEGIN
CLRSR;WRITELN;WRITELN('HYPOTHESIS TEST');
WRITELN('HI: Data in vector ',KOL1,' is ',HYP,');
WRITELN('Ho: Data in vector ',KOL1,' is NOT ',HYP,');WRITELN;
WRITELN('Number of Observations = ',X:4);
WRITELN('Selected Alpha = ',ALPHA:8:4);
WRITELN('Maximum Absolute Difference = ',DDEV:8:4);
WRITELN('Degrees of Freedom = ',X:4);
WRITELN('K-S Prob. two-tailed value = ',ALPHA:8:4);
END;
IF D2EV(ALPHA) THEN
BEGIN
RESULT:='REJECT Ho';
FIT:='is NOT';
END
ELSE
BEGIN
RESULT:='ACCEPT Ho';
FIT:='is';
END;
IF OUTPUT='N' THEN
BEGIN
WRITELN;
WRITELN;
WRITELN;
WRITELN('Thus; ',RESULT,', Vector ',KOLI,', ',FIT,', ',HYP);
END
ELSE
BEGIN
WRITELN(LST);
WRITELN(LST);
WRITELN(LST,'Thus; ',RESULT,', Vector ',KOLI,', ',FIT,', ',HYP);
END;
{ PERFORM ANOTHER TEST ? }
ANS:='M';
WHILE (ANS='Y')AND(ANS='N') DO
BEGIN
WRITELN;
WRITELN;
WRITELN('Would you like to perform another Kolmogorov-Smirnov test ? (Y/N),'
');
READLN(ANS);
END;
{ RETURN TO MAIN MENU }
IS:ASSIGN(VSTAT,'VSTAT.COM');
EXECUTE(VSTAT);
END.
KOLMOGOROV-SNIRNOV TWO-SAMPLE TEST

PROGRAM KS2;
{ MAIN PROGRAM STORAGE SPECIFICATIONS }
LABEL 15,100,240,300,401,450,600;
TYPE
IRAY=ARRAY[1..2] OF INTEGER;
RANGE=ARRAY[1..150] OF REAL;
STRING2=STRING[2];
STRING8=STRING[8];
RAY2=ARRAY[1..15] OF STRING8;
RAY2D=ARRAY[1..152] OF ARRAY[1..152] OF INTEGER;
VAR
DRIVE:STRING2;
VIEW:INTEGER;
RES,ANS,OUTPUT:CHAR;
P,%,Y1,Y2,DATA,VLUE,W,U,V,UU:REAL;
H7,N7,Y3,N,Y2,H, N, BC:REAL;
I,J,K1,K2,I6,N6:INTEGER;
OK:BOOLEAN;
W,V,UU:RANGE;
NUM:IRAY;
L:RAY2D;
O,S,WHICH:STRING[6];
KOL:RAY2;
VECTORFILE:FILE OF REAL;
VSTAT:FILE;

VARIABLE DEFINITIONS - MAIN MENU

DRIVE - Currently assigned disk drive for data.
AMS - User's response to Y/N questions.
RES - User's response to menu selections.
OUTPUT - User's response to output device (terminal or printer).
P - Probability of value being out of bounds.
X,Y1,Y2 - Used to calculate number of interior paths.
DATA - An element of a data vector.
VLUE - Number of test values.
D - Maximum ordered cumulative distribution function difference.
U,V,UU,D1,D2,D3,D4, - Used to calculate D.
M,N,M,C - Used to calculate probability of value being out of bounds.
M7,N7 - Used to count boundary paths.
K - Number of values in vector to be sorted.
MPLUSI - The value of M plus 1.
MPLUS2 - The value of M plus 2.
NGROUP,IG,NTAIL,MGROUP,IX,IX - Used to calculate which probability to calculate.
K1 - Number of elements in first vector.
K2 - Number of elements in second vector.
I,J,M,N - Counters for arrays.
OK - True if file exists.
V - First vector.
W - Second vector.
UU - Vector to be sorted.
NUM - Array containing the number of elements in each vectorfile.
KOL - Array containing names of two vectors.
VECTORFILE - File variable name.
VSTAT - Name of program containing the main menu.

PROCEDURE DATA
PROCEDURE SORT(VAR UU:RANGE; VAR X:INTEGER);
{ PROCEDURE SORT STORAGE SPECIFICATIONS }
LABEL 10,20,30;
VAR
LL,II,J,M,N:INTEGER;
TEMP:REAL;

PROCEDURE SORT - VARIABLE DEFINITIONS
LL,II,J,M - Array counter.
TEMP - Temporary holding variable for sort.

BEGIN
M:=X;
20:M:=TRUNC(M/2);
IF M=0 THEN EXIT;
N:=M;
J:=1;
10:LL:=I+M;
IF UU[II]:UU[LL] THEN
BEGIN
TEMP:=UU[II];
UU[II]:=UU[LL];
UU[LL]:=TEMP;
II:=II+M;
IF II>=L THEN GOTO 40;
END;
J:=J+1;
IF J>N THEN GOTO 20;
GO TO 30;
END(SORT);
BEGIN
END;
-mainprogram
END;
ANS:='Y';
WHILE ANS='Y' DO { LOOP TO REPEAT STATISTIC IF DESIRED }
BEGIN
IF M=0 THEN EXIT;
N:=M;
J:=1;
10:LL:=I+M;
IF UU[II]:UU[LL] THEN
BEGIN
TEMP:=UU[II];
UU[II]:=UU[LL];
UU[LL]:=TEMP;
II:=II+M;
IF II>=L THEN GOTO 40;
END;
J:=J+1;
IF J>N THEN GOTO 20;
GO TO 30;
END(SORT);
BEGIN
END;
-mainprogram
END;
ANS:='Y';
WHILE ANS='Y' DO { LOOP TO REPEAT STATISTIC IF DESIRED }
BEGIN
CLRSCL;WRITELN;WRITELN('KOLMOGOROV-SMIRNOV Two-Sample Test');
WRITELN('A Kolmogorov-Smirnov two-sample test is a way to compare the cumulative');
WRITELN('distribution functions (CDFs) of two independent samples.');
WRITELN('NOTE: The maximum number of data points for each vector is 150.');WRITELN;
{ DETERMINE INFO FOR FIRST VECTOR }
WHICH:='first';
{ DEFINE TWO DATA VECTORS }
FOR J:=1 TO 2 DO
BEGIN
{ LOCATE DESIRED VECTOR }
REPEAT
{ CHECK FOR RETURN W/O ENTRY }
KOL[J]:="";
WHILE (LENGTH(KOL[J])<1)OR(LENGTH(KOL[J]))>6 DO
BEGIN
{ PROMPT USER FOR VECTOR NAME }
WRITELN;WRITELN('What is the name of the ',WHICH,' vector to be tested?');
READLN(KOL[J]);
END;
{ ASSIGN DISK DRIVE TO NAME }
KOL[J]:=DRIVE+KOL[J];
{ CHECK THAT FILE EXISTS }
ASSIGN(VECTORFILE,KOL[J]);
{SI-}RESET(VECTORFILE){$I+1};
OK:=(0);result:=0;
CLRSCL;
END;
{ PRINT ERROR MESSAGE }
BEGIN
WRITELN;WRITELN('VECTOR ',KOL[J],', DOES NOT EXIST!');
{ PROVIDE OPTION TO TRY AGAIN OR EXIT }
ANS:="M"
WHILE (ANS='Y')AND(ANS='N') DO
BEGIN
WRITELN;WRITELN('Would you like to try again to locate a vector? (Y/N)');
READLN(ANS);
END;
{ EXIT THIS SECTION }
IF ANS='M' THEN GOTO 15;
END;
UNTIL OK;
{ DETERMINE LENGTH OF VECTOR }
NUM[J]:=FILESIZE(VECTORFILE);
{ MAXIMUM VECTOR SIZE IS 150 DATA POINTS }
IF NUM[J]>150 THEN
BEGIN
CLRSCL;WRITELN;WRITELN('The maximum number of data points permitted for each vector is 150.');
WRITELN('Your vector violates this requirement.');
WRITELN;WRITELN('Press RETURN to exit:');
END;
READLN(res);GOTO 15;
END;
{ READ VECTOR TO ARRAY }
FOR I:=1 TO NUM[J] DO
BEGIN
READ(vectorfile,data);
IF J=1 THEN
  Y[I]:=DATA
ELSE
  W[I]:=DATA;
END;
{ CLOSE FILE }
CLOSE(vectorfile);
{ REPEAT SECTION FOR SECOND VECTOR }
WHICH:="second";
END;
K1:=NUM[1];K2:=NUM[2];
{ SORT DATA IN FIRST VECTOR }
K:=K1;
{ READ FIRST VECTOR TO SORTING ARRAY }
FOR I:=1 TO K1 DO
  UU[I]:=V[I];
{ CALL SORTING PROCEDURE }
SORT(UU,K);
{ ASSIGN SORTED ARRAY TO FIRST VECTOR }
FOR I:=1 TO K1 DO
  V[I]:=UU[I];
{ SORT DATA IN SECOND VECTOR }
K:=K2;
{ READ SECOND VECTOR TO SORTING ARRAY }
FOR I:=1 TO K2 DO
  W[I]:=UW[I];
{ CALL SORTING PROCEDURE }
SORT(UU,K);
{ ASSIGN SORTED ARRAY TO SECOND VECTOR }
FOR I:=1 TO K2 DO
  W[I]:=UU[I];
{ SELECT OUTPUT DEVICE }
output:='M';
while (output:='Y') AND (output:='N') do
BEGIN
 CLRSCR;WRITELN;WRITELN('Do you want a HARD COPY of the results ? (Y/N)');
READLN(output);
END;
{ OUTPUT TABLE HEADINGS }
CLRSCR;
IF output:='N' THEN
BEGIN
 WRITELN('Kolmogorov-Smirnov Two-Sample Test');WRITELN;
 WRITELN('TEST FREQUENCY % CUMULATIVE % DIFF. ');
 WRITELN('VALUES ',KOL[1],', ',KOL[2],', ',KOL[1],', ',KOL[2]);WRITELN;
 END
ELSE
BEGIN
 WRITELN(LST);WRITELN(LST,'Kolmogorov-Smirnov Two-Sample Test');WRITELN(LST);
 WRITELN(LST,'TEST FREQUENCY % CUMULATIVE % DIFF. ');
 WRITELN(LST,'VALUES ',KOL[1],', ',KOL[2],', ',KOL[1],', ',KOL[2]);WRITELN(LST);
 END;
{ CALCULATE CUMULATIVE DIFFERENCE }
M:=[M:=0];D:=0;D1:=0;D3:=0;
FOR I:=1 TO N DO
  IF (M<K1) OR (M<K2) THEN
BEGIN
  VALUE:=V[M];
  IF M=K2 THEN
BEGIN
    IF (M<K1) OR (V[M]>W[M]) THEN
BEGIN
      VALUE:=W[M];
      GOTO 240;
    END;
  END;
END;
WHILE (N+1=K) AND (V[N]=V[N+1]) DO
  BEGIN
    I:=I+1;
    N:=N+1;
  END;
  IF M=K THEN
  BEGIN
    IF V[N](M) THEN
      BEGIN
        J:=J-1;
        M:=M-1;
      END;
    IF ((M+1)=K2) AND (V[N](M+1)) THEN GOTO 300;
    240: IF (M+1)=K2 AND (W[M]=M+1)) THEN
      BEGIN
        J:=J+1;
        M:=M+1;
        GOTO 240;
      END;
    IF (N=K) AND (V[N]=W[M]) THEN GOTO 300;
  END;
  IF (M=K2) THEN
    J:=J-1
  ELSE
    BEGIN
      I:=I-1;
      M:=M-1;
    END;
  IF M=K THEN
    M:=M-1;
  300: U:=100*M/K1;
  VV:=100*K2;
  D2:=U-VV;
  D4:=ABS(U-VV);
  { PRINT TABLE VALUES }
  IF OUTPUT='N' THEN
    WRITE(V,N:8:4,','I:3,','J:3,','U:13:4,','VV:13:4,','D2:13:4);
  ELSE
    WRITE(LST,N:8:4,','I:3,','J:3,','U:13:4,','VV:13:4,','D2:13:4);
  IF D4 D3 THEN D:=D4;
  IF D2 D3 THEN D3:=D2;
  IF D2 D1 THEN D1:=D2;
  IF U=0 THEN M:=0;
  IF VV=0 THEN M:=0;
  GOTO 100;
{ PRINT TOTAL OF EACH VECTOR AND MAX. CUM. DIFF. }
IF OUTPUT='N' THEN
  BEGIN
    WRITE('Total',',K1,','K2);
    WRITE;WRITE('Maximum Cumulative Difference = ',D:7:4,','I');WRITE;
  END;
ELSE
  BEGIN
    WRITE(LST,'Total',',K1,','K2);
    WRITE(LST);WRITE(LST,'Maximum Cumulative Difference = ',D:7:4,','I');
  END;
  IF OUTPUT='N' THEN
    BEGIN
      WRITE('Press RETURN to continue:');
      READLN(RE);CLRSCR;
    END;
{ PROBABILITY COMPUTATIONS }
IG:=2;NHYP:=1;LTAIL:=1;
IF K1< K2 THEN
  BEGIN
    M:=K1;
    M:=K2;
  END;
ELSE
  BEGIN
    M:=K2;
  END.
M:=XI;
END;
MPLUS1:=M+1;MPLUS1:=M+1;
NGROUP:=0;
BC:=1;
FOR I:=1 TO N DO
BC:=BC*(N+M+I-1)/(N+1-I);
IF (ABS(D1)>D3)AND(X2>2) THEN NGROUP:=1;
IF (D2)>ABS(D1)AND(X2>2) THEN NGROUP:=1;
401:FOR I:=1 TO MPLUS1 DO
BEGIN
FOR J:=1 TO MPLUS1 DO
L[I,J]:=0;
END;
M7:=TRUNC(D/100*M+0.9999);
M7:=TRUNC(D/100*M+0.9999);
IF LTAIL=2 THEN
BEGIN
{ COUNT THE BOUNDARY PATHS }
IF NGROUP(I) THEN
BEGIN
FOR I:=2 TO M7 DO
BEGIN
L[I,1]:=1;
IF M7/I THEN L[I,1]:=0;
END;
FOR J:=2 TO MPLUS1 DO
L[I,1]:=1;
END;
FOR I:=2 TO MPLUS1 DO
L[I,1]:=1;
FOR J:=2 TO M7 DO
BEGIN
L[I,1]:=1;
IF M7/I THEN L[I,1]:=0;
END;
GOTO 450;
END;
FOR I:=2 TO M7 DO
BEGIN
L[I,1]:=1;
IF M7/I THEN L[I,1]:=0;
END;
FOR J:=2 TO M7 DO
BEGIN
L[I,1]:=1;
IF M7/I THEN L[I,1]:=0;
END;
BEGIN:
{ COUNT THE INTERIOR PATHS }
450:FOR I:=2 TO MPLUS1 DO
BEGIN
FOR J:=2 TO MPLUS1 DO
BEGIN
X:=(J-1)/N;
Y:=(I-1)/M+D/100-0.0001;
Y2:=(I-1)/M+D/100-0.0001;
L[I,J]:=L[I,J]+L[I,J-1];
IF LTAIL=2 THEN
BEGIN
IF (X>Y1)OR(X>Y2) THEN L[I,J]:=0;
END;
ELSE BEGIN
IF (ABS(NGROUP-1)<0.001)AND(D1) Then L[I,J]:=0;
IF (ABS(NGROUP)<0.001)AND(D1) Then L[I,J]:=0;
END;
END;
BEGIN
{ CALCULATE P, THE PROBABILITY OF BEING OUT OF BOUNDS }
P:=1-L[M+1,N+1]/BC;
600:
{ CALCULATE INFORMATION FOR ONE-TAIL }
IF LTAIL<2 THEN
BEGIN
IF NHYP<2 THEN
BEGIN
{ CALCULATE FIRST CUM. FUNCTION DIFFERENCE }
IF D3-D THEN IG:=1;
N3:=3-IG;
END;
IF OUTPUT='M' THEN
BEGIN
WRITE(LST,'The exact ',LTAIL,'-tailed probability is ',P,':4,' that any apparent');
END
ELSE
BEGIN
WRITE(LST,'The exact ',LTAIL,'-tailed probability is ',P,':4,' that any apparent);
END;
IF NHYP<2 THEN
BEGIN
IF IG=1 THEN
BEGIN
S:=XOL[1];
D:=XOL[2];
END;
IF IG=2 THEN
BEGIN
S:=XOL[2];
D:=XOL[1];
END;
IF OUTPUT='M' THEN
BEGIN
WRITE('exceeding of ',S,' values by ',D,' values is due to chance.');
ELSE
WRITE(LST,'exceeding of ',S,' values by ',D,' values is due to chance.');
END;
IF D<(-D1) THEN D:=D1;
IF OUTPUT='M' THEN
BEGIN
WRITE('Max. ordered cum. dist. function difference = ',D,':4,' Z');
ELSE
WRITE(LST,'Max. ordered cum. dist. function difference = ',D,':4,' Z');
END;
NHYP:=2;
IF D3<(-D1) THEN
D:=D3
ELSE
D:=D1;
GOTO 401;
END;
IF N3=1 THEN
BEGIN
S:=XOL[1];
D:=XOL[2];
END;
IF N3=2 THEN
BEGIN
S:=XOL[2];
D:=XOL[1];
END;
IF OUTPUT='M' THEN
BEGIN
WRITE('exceeding of ',S,' values by ',D,' values is due to chance.');
ELSE
BEGIN
WRITE(LST,'exceeding of ',S,' values by ',D,' values is due to chance.');
END;
IF D<(-D1) THEN D:=D1;
IF OUTPUT='M' THEN
BEGIN
WRITE('Max. cum. dist. function difference = ',D,':4,' Z');
END;
ELSE
BEGIN
WRITE(LST,'Max. cum. dist. function difference = ',D,':4,' Z');
END;
END;
ELSE
D:=ABS(D3)
ELSE
D:=ABS(-D1);
LTAIL:=2;
GOTO 401;
END;
IF OUTPUT='N' THEN
BEGIN
   WRITELN('The exact ',LTAIL,'-tailed probability is ',P:8:4,' that a cum. dist.');</n
   WRITELN('function difference of ',D:7:4,' % or more is random for samples');</n
   WRITELN('of sizes ',K1,' and ',K2,' ');<n
END<strong> ELSE</strong>
BEGIN
   WRITELN(LST,'The exact ',LTAIL,'-tailed probability is ',P:8:4,' that a cum. dist.');</n
   WRITELN(LST,'function difference of ',D:7:4,' % or more is random for samples');</n
   WRITELN(LST,'of sizes ',K1,' and ',K2,' ');<n
END</strong>;
ANS::'H'.
WHILE (ANS()='Y')AND(ANS()='N') DO
BEGIN
   WRITELN(WRITEI;
WRITELN('Would you like to perform another KOLMOGOROV-Smirnov Two-Sample t'
   test ? (Y/N)');
   READLN(ANS);<n
END;
{ RETURN TO MAIN MENU }
IS:ASSIGN(VSTAT,'VSTAT.COM');
EXECUTE(VSTAT);<n
END.}
\[ \text{Z, } F, \text{ and Chi-Squared Values} \]

PROGRAM VALUE:
{ MAIN PROGRAM STORAGE SPECIFICATIONS }

LABEL 15:

TYPE
STRING2=STRING[2];
STRING4=STRING[4];
STRING13=STRING[13];
STRING60=STRING[60];

VAR
DRIVE:STRING2;
VIEW:INTEGER;
RES,CH,ANS:CHAR;
ALPHA,Z,YX,VN,S1,K1J:REAL;
S2,S3,CODE:INTEGER;
STS2,STS3,ENTRY,OKAY:STRING4;
DD:STRING[6];
STS1:STRING13;
ERR:STRING60;
VSTAT:FILE;

VARIABLE DEFINITIONS - MAIN MENU

AMS - User's response to Y/N questions.
RES,CH - User's response to menu selections.
ALPHA - Area to right-hand side of chi-squared statistic.
KIJ,Z - Area to right-hand side of 2, F and t statistics.
YY,VN - Used in calculation of area to right of 2-statistic.
S1 - Statistic entered by user to be evaluated.
S2 - Degrees of freedom (DF of numerator for F statistic).
S3 - Degrees of freedom of denominator for F statistic.
CODE - Signals error in data conversion.
STS1 - Statistic entered as string by user.
STS2,STS3 - Degrees of freedom entered as strings.
ERR,ENTRY,OKAY - Strings used to signal invalid entry by user.
DD - The number corresponding to the statistic selected by the user.
VSTAT - File containing the main menu.

RAISE TO A POWER}
POW = X ** Y }

PROCEDURE POWER(VAR X,Y,POW:REAL);
{ PROCEDURE POWER STORAGE SPECIFICATIONS }
LABEL 5;
VAR
NUM,XX,P:REAL;
YY:INTEGER;

PROCEDURE POWER - VARIABLE DEFINITIONS

P - Number raised to a power.
NUM,XX,YY - Used in calculation of raising to a power.

BEGIN
{ EXONENT OF ZERO }
IF (Y=0) THEN BEGIN
POW:=1.0;
GOTO 5;
END;
{ BASE OF ZERO }
IF (X=0) THEN BEGIN
POW:=0.0;
GOTO 5;
END ELSE
{ NEGATIVE BASE }
IF (X<0) THEN XX:=-0.0-X
{ POSITIVE BASE }
ELSE XX:=X;
NUM:=Y*LN(XX); P:=EXP(NUM); { CHECK SIGN }
IF (X(0) THEN BEGIN YY:=TRUNC(Y); IF (ODD(YY)=TRUE) THEN POW:=0.0-P ELSE POW:=P; END ELSE POW:=P; 5: END(func power);

{ CHECK FOR INVALID DATA ENTRIES } PROCEDURE CHECK(VAR STSI:STRING13; VAR ERR:STRING60); { PROCEDURE CHECK STORAGE SPECIFICATIONS }
TYPE CHARACTERSET=SET OF CHAR;
VAR VALID:CHARACTERSET;
POS,COUNT: INTEGER;
POSITION:CHAR;
PROCEDURE CHECK - VARIABLE DEFINITIONS
VALID - Set of valid characters. POS - Position in a string.
COUNT - Counter for invalid elements. POSITION - A single character of a string.
BEGIN { DEFINE VALID CHARACTERSET }
VALID:=[O..Z,'_','.'']; { INITIALIZE COUNTER }
COUNT:=0;
{ CHECK EACH POSITION IN THE STRING }
FOR POS:=1 TO LENGTH(STSI) DO BEGIN
POSITION:=COPY(STSI,POS,1);
IF POSITION IN VALID THEN COUNT:=COUNT+1;
END; { COMPARE # OF VALID POSITIONS TO LENGTH OF STRING }
IF COUNT)LENGTH(STSI) THEN ERR:='BAD' ELSE ERR:='GOOD'; END(CHECK);

{ CHECK DEGREES OF FREEDOM }
PROCEDURE CHECKDF(VAR STSS,ENTRY:STRING41; { PROCEDURE CHECKDF STORAGE SPECIFICATIONS }
TYPE CHARACTERSET=SET OF CHAR;
VAR COUNT,POS: INTEGER;
POSITION:CHAR;
VALID:CHARACTERSET;
PROCEDURE CHECKDF - VARIABLE DEFINITIONS
COUNT - Used to count number of valid positions in entry.
POS - The position of a character in a string.
POSITION - A single character of a string.
VALID - A set of characters.
BEGIN { INITIALIZE COUNTER }
COUNT:=0;
{ DEFINE VALID DATA SET }
VALID:=['0'..'9'];
{ CHECK EACH POSITION OF THE STRING }
FOR POS:=1 TO LENGTH(STSS) DO
BEGIN
  POSITION:=COPY(STSS,POS,1);
  IF POSITION IN VALID THEN
    COUNT:=COUNT+1;
END;
{ COMPARE # VALID POSITIONS TO LENGTH OF STRING }
IF COUNT<LENGTH(STSS) THEN
  ENTRY:='BAD'
ELSE
  ENTRY:='GOOD'
END(CHECKDF);
{ NORMAL DISTRIBUTION }
PROCEDURE NORMAL(VAR YX,VN:REAL);
{ PROCEDURE NORMAL STORAGE SPECIFICATIONS }
VAR
  TT,YZ:REAL;
BEGIN
  YX:=-YX;
  IF YX>4.17 THEN
    VN:=1
  ELSE IF YX<-4.17 THEN VN:=0
  ELSE BEGIN
    YZ:=YX;
    IF YX<0 THEN YZ:=-YX;
    TT:=Y/1.4142142;
    D:=(0.43065+4*TT+0.2765672+3)*TT+0.152014E-3+0.92705272E-2*TT+0.42282012E-1
    *TT+0.70523078E-1)*TT+1.0;
    D1:=D*D;D2:=D*D;D:=D*D;D:=D*D
    VN:=0.5-0.5*V;
    IF YX<0 THEN VN:=0.5-V;
    IF YX=0 THEN VN:=0.5+V;
  END;
END(NORMAL);
{ F DISTRIBUTION }
PROCEDURE FDISTR(VAR S1,Z:REAL);
{ PROCEDURE FDISTR STORAGE SPECIFICATIONS }
VAR
  LABEL 10,15,20,25,30,35;
  TYPE
    CHARACTERSET=SET OF CHAR;
    STRING4=STRING[4];
  CODE:INTEGER;
  STS2,STS3,ENTRY:STRING4;
  VALID:CHARACTERSET;
{ PROCEDURE FDISTR - VARIABLE DEFINITIONS }
X - A base number.
Y - An exponent.
POW - A number raised to a power. (X * Y)
P - Used to determine if df is even or odd number.
SI,F - F statistic entered by user.
S2,DF1,DD1 - Degrees of freedom of numerator.
S3,DF2,DD2 - Degrees of freedom of denominator.
CODE - Signals data conversion error.
STS3,STS2 - df entered as strings so validity can be checked.
ENTRY - Signals invalid entry by user.
VALID - Set of characters.
BEGIN
  ENTRY:='BAD';
  WHILE ENTRY='BAD' DO
    BEGIN

WRITELM('Enter the Degrees of Freedom for the NUMERATOR:');
READLM(STS2);
{ CALL PROCEDURE TO CHECK DF ENTRY }
CHECKDF(STS2,ENTRY);
IF STS2='Q' THEN
  ENTRY:='BAD';
END;
{ CHANGE STRING TO INTEGER }
VAL(STS2,S2,CODE);
ENTRY:='BAD';
WHILE ENTRY='BAD' DO
BEGIN
  WRITELM('Enter the Degrees of Freedom for the DENOMINATOR:');
  READLM(STS3);
  { CALL PROCEDURE TO CHECK DF ENTRY }
  CHECKDF(STS3,ENTRY);
  IF STS3='O1 THEN
    ENTRY:='BAD';
  END;
  { CHANGE STRING TO INTEGER }
  VAL(STS3,S3,CODE);
  F:=S1;DF1:=S2;DF2:=S3;DD1:=S2;DD2:=S3;
  IF F=0 THEN
    BEGIN
      Z:=1;EXIT;
    END;
  O1:=DF2/(DF2+DF1*F);
  VP:=DF1+DF2-2;
  P:=DF1-TRUNC(DF1/2)*2;
  IF (P@0.OR(DD1@200) THEN GOTO 15;
  O2:=1-01;
  DF1:=DF1-2;
  FT:=0;
  WHILE DF1@1 DO
  BEGIN
    VP:=VP-2;
    FT:=02*VP/DF1*(1+FT);
    DF1:=DF1-2;
  END;
  X:=01;
  Y:=(1.5*DF2);
  POWER(X,Y,POW);
  FT:=POW*(1+FT);
  GOTO 10;
  IF DF1.000 OR:200) }
  IS:=DD2-TRUNC(DD2/2)*2;
  IF (P@0.OR(DD2@200) THEN GOTO 20;
  DF2:=DF2-2;
  FT:=0;
  WHILE DF2@1 DO
  BEGIN
    VP:=VP-2;
    FT:=01*VP/DF2*(1+FT);
    DF2:=DF2-2;
  END;
  X:=1-01;
  Y:=(0.5*DF1);
  POWER(X,Y,POW);
  FT:=1-(POW*(1+FT));
  GOTO 10;
  IF (DD1+DD2@200) THEN GOTO 25;
  TA:=ARCTAN(SQRT(DF1*F/DF2));
  SH:=SIN(TA);CHH:=COS(TA);ST:=SH*SH;CS:=CHH*CHH;
  IF DD2@1 THEN GOTO 30;
  DF2:=DF2-2;
  AJ:=0;
  WHILE DF2@2 DO
  BEGIN
    AJ:=CS*(DF2-1)/DF2*(1+AJ);
    DF2:=DF2-2;
  END;
  AJ:=SH*CHH*(1+AJ);
30:AJ:=TA+AJ;
IF DFI=1 THEN GOTO 35;
DF1:=DF1-2;
BJ:=0;
WHILE DFI>2 DO
BEGIN
    VP:=VP-2;
    BJ:=ST+VP/DF1*(1+BJ);
    DFI:=DFI-2;
END:
GA:=1;
DF2:=0.5*DD2;
XI:=0;XI:=XI+1;
WHILE XI<DF2 DO
BEGIN
    GA:=XI*GA/(XI-O.5);
    XI:=XI+1;
END:
X:=CHH;
Y:=DD2;
POWER(X,Y,POW);
BJ:=GA*SH*POW*(1+BJ);
35:FT:=1+0.636619772368*(BJ-AJ);
GOTO 10;
25:DF1:=2/(9*DF1);
DF2:=2/(9*DF2);
X:=F;
Y:=I/3;
POWER(X,Y,POW);
CB:=POW;
YX:=(1-DF2)*CB+DF1-1)/SORT(DF2*(CB*CB)+DF1));
NORMAL(YX,VM);
FT:=VM;
10:Z:=FT;
IF Z<0 THEN Z:=0;
END(FDISTR):
{ T D I S T R I B U T I O N }
PROCEDURE TDISTR(VAR S2:INTEGER; VAR S1,Z:REAL);
{ PROCEDURE TDISTR STORAGE SPECIFICATIONS }
VAR
    T1,T2,D,VM,A,YX,YZ,B,TV:REAL;
    DFF:INTEGER;
{ PROCEDURE TDISTR - VARIABLE DEFINITIONS }
{ DFF- Degrees of freedom. }
{ T2,D,VM,A,YX,YZ,B,TV- Used in the calculation of the probability. }
BEGIN
    T1:=S1;
    TV:=T1#T1;
    DFF:=S2;
    YX:=TV/DF1;
    B:=1+YX;
    { FOR SMALL SAMPLE (LESS THAN 20) }
    IF (DF2<20) THEN
BEGIN
    YX:=SORT(YX);
    A:=YX;
    IF (DF1<1) THEN A:=0;
    DFF:=DFF-2;
    WHILE (DFF)1 DO
BEGIN
    A:=(DF2-1)/(B*DF1)*A+YX);
    DFF:=DFF-2;
    END;
    IF (DF<0) THEN
A:=(ARCTAN(YX)+A/B)*0.63661977236
ELSE
A:=A/SORT(B);
IF (A)<1 THEN
Z:=0;
ELSE
  Z:=1-A;
END;
{ FOR LARGE SAMPLE (GREATER THAN OR = 20 )}
IF DFF>20 THEN
BEGIN
  { ASYMPTOTIC SERIES FOR LARGE SAMPLE } 
  IF (YX)(0.1E-5)) THEN
    YX:=LN(B);
    A:=DFF-0.5;
    B:=48*A**;
    YX:=#YX;
    YX:=((((0.4#YX-3.3)*YX-24)*YX-85.5)/(0.8*(YX**YX)+100+B)+YX+3)/B+1)*SQRT(YX);
  { NORMAL DISTRIBUTION } 
  YX:=YX;
  IF (YX>4.17) THEN
    VN:=0
  ELSE
  IF (YX>4.17) THEN
    VN:=1
  ELSE
BEGIN
  YZ:=YX;
  IF (YX<0) THEN YZ:=-YX;
  T2:=Y2/1.4142142;
  D:=((((0.430638E-4*#T2+0.276562E-3)*#T2+0.152014E-3)*#T2+0.92705272E-2)*#T2+0.42282012E-1)*#T2+0.7052
  3078E-1)*#T2+1.0;
  D:=#D*D:D:=#D*D:D:=#D*D:D:=#D*D;
  VN:=0.5-0.5/VN;
  IF (YX<0) THEN VN:=0.5-VN;
  IF (YX=0) THEN VN:=0.5; 
  IF (YX=0) THEN VN:=0.5+VN;
  Z:=#*VN;
  IF (Z<0) THEN Z:=0;
END;
END(TDISTR);
{ CHI-SQUARED DISTRIBUTION } 
PROCEDURE CHI(VAR S2:INTEGER; VAR S1,ALPHA:REAL);
{ PROCEDURE CHI STORAGE SPECIFICATIONS }
LABEL 10;
VAR
  DFF:INTEGER;
  SE,YX,TI,ZX,EE,CX,DI,MIN,MAX:REAL;
{ VARIABLE DEFINITIONS - PROCEDURE CHI }
DIFF - Degrees of freedom.
SE,YX,TI,2X,EE,CX,DI,MI,MAX - Used in calculation of area.
BEGIN
  DFF:=S2;
  MIN:=0.5*S1;
  IF (((DI<1)OR(DIFF>2))AND(MAX=0)) THEN
BEGIN
  IF MIN<2SO THEN
    S2:=0
  ELSE
    S2:=TRUNC(EXP(-MIN));
END;
IF DI<1 THEN
BEGIN
  SE:=SQRT(S1);
  YX:=SE;
  NORMAL(YX,VN);
  TI:=#*VN;
END;
ELSE
  TI:=S2;
  IF DFF<2 THEN GOTO 10;
  S1:=0.5*(DFF-1);
  IF DI=1 THEN
    ZX:=1
  ELSE
    ZX:=0.5;
  IF MAX=1 THEN
  BEGIN
    IF DI1 THEN
      EE:=0.572364912925
    ELSE
      EE:=0;
    CX:=LN(MIN);
    WHILE ZX<S1 DO
    BEGIN
      EE:=LN(ZX)+EE;
      TI:=EXP(CX*ZX-MIN-EE)+TI;
      ZX:=ZX+1;
    END;
    IF ZX>S1 THEN GOTO 10;
  END;
  IF DI1 THEN
    EE:=0.56149583548/SQRT(MIN)
  ELSE
    EE:=1;
    CX:=0;
    WHILE ZX<S1 DO
    BEGIN
      EE:=EE-MIN/ZX;
      CX:=CX+EE;
      ZX:=ZX+1;
    END;
    ALPHA:=CX*S2*EE;
    EXIT;
    IF IO=ALPHA:=TI;
  END;
  { MAIN PROGRAM }
  BEGIN
    ANS:=\"y\";
    WHILE ANS=\"y\" DO
      { LOOP TO REPEAT STATISTIC IF DESIRED }
    BEGIN
      { INITIALIZATION }
      RES:=\"0\";CH:=\"0\";
      { PROMPT USER WITH MENU OF AVAILABLE TEST STATISTICS }
      WHILE ((CH\"1\")AND(CH\"2\")AND(CH\"3\")AND(CH\"4\")AND(CH\"5\")) DO
      BEGIN
        CLEARSCR;WRITE;WRITE;WRITE;WRITE('TEST STATISTIC SECTION');
        WRITE;WRITE('1:Normal Distribution');
        WRITE;WRITE('2: F Distribution');
        WRITE;WRITE('4: Chi-Square Distribution');
        WRITE;WRITE('5: EXIT this section');
        READLN(CH);
      END;
      { ASSIGN APPROPRIATE STRING }
      CASE CH OF
        \"1\": DD:=\"Z\";
        \"2\": DD:=\"F\";
        \"4\": DD:=\"Chi-Sq\";
        \"5\": GOTO 15;
      END OF CASE;
      OKAY:=\"BAD\";
      WHILE OKAY=\"BAD\" DO
      BEGIN
        ERR:=\"BAD\";
        WHILE ERR=\"BAD\" DO
      BEGIN

{ PROMPT USER TO ENTER STATISTIC VALUE }
CLSCHR;WRITELN(WRITELN('Enter the ','DD', ' Statistic Value:');
READLH(STSI);
{ CHECK FOR INVALID ENTRY }
CHECK(STSI,ERR);
{ CHANGE STRING TO REAL }
VAL(STSI,SI,CODE);
{ CHECK VALIDITY }
IF CH='2' THEN
  IF SI=0 THEN OKAY:='GOOD';
  IF CH='4' THEN
    IF SI<0 THEN OKAY:='GOOD';
    IF ('='='1') THEN OKAY:='GOOD';
    IF (CH='3') THEN OKAY:='GOOD';
END;
IF (CH='3')OR(CH='4') THEN
BEGIN
ENTRY:='BAD';
WHILE ENTRY='BAD' DO
BEGIN
{ PROMPT USER TO ENTER DEGREES OF FREEDOM }
WRITELN;WRITELN(WRITELN('Enter the DEGREES OF FREEDOM:');
READLH(STS2);
{ CALL PROCEDURE TO CHECK DF ENTRY }
CHECKDF(STS2,ENTRY);
{ ZERO OF INVALID }
IF STS2='0' THEN
  ENTRY:='BAD';
END;
{ CHANGE STRING TO INTEGER }
VAL(STS2,S2,CODE);
END;
{ CALL APPROPRIATE PROCEDURE TO CALCULATE AREA }
CASE CH OF
  '1': BEGIN
    YX:=-SI;
    { NORMAL DISTRIBUTION }
    NORMAL(YX,VN);
    KIJ:=-1-VN;
    END;
  '2': BEGIN
    { F DISTRIBUTION }
    FDISTR(S1,Z);
    IF Z<0 THEN Z:=-Z;
    KIJ:=Z;
    END;
  '3': BEGIN
    { T DISTRIBUTION }
    TDISTR(S2,S1,Z);
    KIJ:=Z/S;
    IF SI<0 THEN KIJ:=-KIJ;
    END;
  '4': BEGIN
    { CHI-SQUARED DISTRIBUTION }
    CHI(S2,S1,ALPHA);
    IF ALPHA<1 THEN ALPHA:=1;
    KIJ:=ALPHA;
    END;
ENDOF(CASE);
{ PRINT THE AREA TO THE RIGHT-HAND SIDE OF GIVEN STATISTIC }
WRITELN;WRITELN;WRITELN('The area to the right of the ','DD', value is ','KIJ:8:4);
{ OPTION TO REPEAT STATISTIC }
ANS:='N';
WHILE (ANS('Y')AND(ANS('N')) DO
BEGIN
  WRITELN;WRITELN;WRITELN('Would you like to determine another value ? (Y/N)');
  READLN(ANS);
END;
END;
{ RETURN TO MAIN MENU }
IS:ASSIGN('VSTAT','VSTAT.COM');
EXECUTE(VSTAT);
END(VALUE).
{ RANDOM SAMPLING/NUMBER GENERATION }
PROGRAM RAND;
LABEL S,10;
TYPE
STRING2=STRING[2];
STRING43=STRING[43];
RAY43=ARRAY[1..4] OF STRING43;
RAY4=ARRAY[1..4] OF REAL;
RANGE=ARRAY[1..1000] OF REAL;
VAR
DRIVE=STRING2;
VIEW:INTEGER;
STAT:CHAR;
ANS,RES,CH:CHAR
SI,YY,AMT,C,ALL,CH:INTEGER;
RNUM,SEED,STSC,DATASEE,Y:REAL;
OK:BOOLEAN;
RNDV:STRING[8];
ST:STRING[13];
PL,WHICH:STRING[30];
M,RAY43;
RAY4:[I..4] OF REAL-
RAN6E:ARRAY[1..1000] OF REAL;
VSTAT:FILE;
VECTORFILE:FILE OF REAL;
VECTORFILE:FILE;

PROGRAM RAND - VARIABLE DEFINITIONS

DRIVE - Currently assigned disk-drive for data.
STAT - Signals which section is to be executed.
ANS - User's response to yes/no questions.
RES - Type of random sampling to be done.
CH - Type of distribution random #'s are to come from.
CODE - Signals error in type conversion.
ALL - The size of the random sample.
YY,C - Used as counters for arrays.
NM - Amount of random numbers to be generated.
SE - Number of inputs to expect from user.
ST - Info. entered by user as string to check validity.
STSC,S - Info. entered by user converted to REAL or INTEGER.
DATA - Data points created by program to be written to new file.
KU,X,Y - Used in calculating random numbers for various distributions.
OK - True if file already exists.
VRAND - Name of file for result to be stored in.
PL,WHICH - Strings containing appropriate titles.
M - Strings of prompts for user.
X - Array that sample or random numbers are put in.
VECTORFILE - File containing the new vector.
VSTAT - File containing the main menu.

{ RANDOM NUMBER GENERATOR }
PROCEDURE RNDM(VAR SEE,RNUM:REAL; VAR AMT:INTEGER);
VAR
SD,SD1,SD2:REAL;
BEGIN
SD:=(9192*SEED);
SD1:=INT(6/709547.0);
SD2:=SD1*709547.0;
RNUM:=SD-SD2;
SEE:=RNUM;
RNUM:=FRAC(RNUM/(1E+7));
END(RNDM);

MAIN PROGRAM

BEGIN
ANS:="Y"; { LOOP TO REPEAT SECTION IF DESIRED }
WHILE (ANS='Y') DO
BEGIN
IF STAT='1' THEN WHICH:='RANDOM NUMBER GENERATOR';
IF STAT='2' THEN WHICH:='RANDOM SAMPLING';
CLSCHR;WRITELN;WRITELN(''
,WHICH,' SECTION');
// SELECT NAME OF VECTOR FOR RESULT

// CHECK FOR RETURN W/O ENTRY

// WHILE (LENGTH(RANDV) < 1 OR (LENGTH(RANDV) > 6)) DO

// BEGIN

// WRITELN(WRITELN('What is the name of the vector that the result is to be stored in ?'));

// WRITELN('Maximum 6 characters');

// READLN(RANDV);

// END;

// ASSIGN DISK-DRIVE TO FILENAME

// RANDV: DRIVE = RANDV;

// { CHECK IF FILE W/ THIS NAME EXISTS }

// ASSIGN VECTORFILE, RANDV;

// {I-}RESET(VECTORFILE){+$};

// OK := [FILERESULT = O];

// IF OK THEN

// BEGIN

// CLEARSCREEN; WRITELN('WARNING - There already exists a vector file with this name !');

// WRITELN('Please press RETURN and choose another name: ');

// END;

// UNTIL NOT OK;

// SELECT TYPE OF RANDOM SAMPLING

// IF STAT = '2' THEN

// BEGIN

// READLN(RES);

// WHILE (RES = '1' AND (RES = '2')) DO

// BEGIN

// CLEARSCREEN; WRITELN('Enter the number of your selection and press RETURN: ');

// READLN(RES);

// END;

// END;

// SELECT DISTRIBUTION FOR RANDOM NOS.

// IF STAT = '1' THEN

// BEGIN

// CH := '0';

// WHILE (CH = '1' AND (CH = '2') AND (CH = '3')) DO

// BEGIN

// CLEARSCREEN; WRITELN('Enter the number of your selection and press RETURN: ');

// END;

// END;

// SELECT INPUT PROMPTS

// IF STAT = '2' THEN PI := 'Random Sampling';

// IF STAT = '1' THEN

// BEGIN

// CASE CH OF

// '1': PI := 'Uniformly Distributed';

// '2': PI := 'Normal Distribution';

// '3': PI := 'Negative Exponential Distribution';

// END(OF CASE);

// END;

// M[1] := 'Seed:';

// IF STAT = '2' THEN

// BEGIN


// DE := 3;

// END;

// IF STAT = '1' THEN

// BEGIN

// IF CH = '3' THEN

// ...
SE:=3
ELSE
  SE:=4;
M[4]:='How many numbers do you want? (max = 1000)';
CASE CH OF
  '1':BEGIN
    M[2]:='Lower limit:';
    M[3]:='Upper limit:';
    END;
  '2':BEGIN
    M[2]:='Mean: ';
    M[3]:='Standard Deviation: ';
    END;
  '3':BEGIN
    M[2]:='Mean: ';
    M[3]:=M[4];
    END;
END OF CASE;
END;
{ PROMPT USER TO ENTER NEEDED INFO. }
FOR C:=1 TO SE DO
BEGIN
  CLRSCR;WRITELN;WRITELN('Please enter the information for the '.PI,' sample.');
  IF C:=1 THEN
    BEGIN
      WRITELN;WRITELN('(NOTE: Seed must be an odd integer between 1 and 32767)');
      END;
    WRITELN;WRITELN(M(C));
    READLN(ST);
    { CHANGE FROM STRING TO REAL }
    VAL(ST,STSC,CODE);
    S[C]:=STSC;
    { CHECK VALIDITY OF ENTERED DATA }
    IF STAT='1' THEN
      BEGIN
        IF (CH='1')AND(C=3)AND(S[3]<S[2]) THEN
          GOTO 5;
        IF (CH='2')AND(C=3)AND(S[3]<0) THEN
          GOTO 5;
        IF (C=SE)AND(S[C]<0) THEN
          GOTO 5;
        END;
    IF STAT='2' THEN
      BEGIN
        IF (C=2)AND(S[2]<0) THEN
          GOTO 5;
        IF (C=3)AND(RES='2')AND(S[3]<S[2]) THEN
          GOTO 5;
        END;
    { CHECK SEED VALUE }
    SI:=TRUNC(S[1]);
    IF (ODD(SI))TRUE THEN GOTO 5;
    IF (S[I]<100)OR(S[I]>32767) THEN GOTO 5;
    END;
    AMT:=TRUNC(S[4]);SEED:=S[1];
    { RANDOM SAMPLING SECTION }
    IF STAT='2' THEN
      BEGIN
        { DETERMINE NUMBER IN SAMPLE }
        NM:=TRUNC(S[3]);
        S[2]:=TRUNC(S[2]);
        { SAMPLING WITH REPLACEMENT }
        IF (RES='1') THEN
          BEGIN
            FOR C:=1 TO NM DO
              BEGIN
                { RANDOM - GENERATES RANDOM NUMBERS }
                RAND(SEED,RNUM,AMT);
                Y:=(S[2]-1)*RNUM+.5;
                X[C]:=INT(Y);
                END;
            END;
          END;
        END;
      END;
    END;
{ SAMPLING WITHOUT REPLACEMENT }
IF (KES='2') THEN
BEGIN
FOR C:=1 TO NM DO
BEGIN
10:RNDM(SEED,RNUM,AMT);
   Y:=1+(S[2]-1)*RNUM+.5;
   Y:=INT(Y);
   YY:=TRUNC(Y);
   IF KU[YY]=1 THEN
      GOTO 10;
   KU[YY]:=1;
   X[C]:=Y;
END;
END;
{ RANDOM NUMBER GENERATOR SECTION }
IF STAT='I' THEN
BEGIN
{ DETERMINE NUMBER IN SAMPLE }
IF CH='3' THEN
   NM:=TRUNC(S[3])
ELSE
   NM:=TRUNC(S[4]);
{ SAMPLE FROM UNIFORM OR NEG. EXPONENTIAL DISTR. }
IF (CH='1')OR(CH='3') THEN
BEGIN
FOR C:=1 TO NM DO
BEGIN
   { RANDOH - GENERATES RANDOM NUMBERS }
   RNDM(SEED,RNUM,AMT);
   Y:=RNUM;
   IF CH='1' THEN
   IF CH='3' THEN
      Y:=-L(N)(Y)*S[2],;
   X[C]:=Y;
END;
END;
{ SAMPLE FROM NORMAL DISTRIBUTION }
IF CH='2' THEN
BEGIN
FOR C:=1 TO NM DO
BEGIN
   SEE:=0;
   FOR YY:=1 TO 12 DO
BEGIN
   { RANDOH - GENERATES RANDOM NUMBERS }
   RNDM(SEED,RNUM,AMT);
   SEE:=SEE+RNUM;
END;
   X[C]:=S[2]+(SEE-6)*S[3];
END;
END;
END;
IF STAT='1' THEN
BEGIN
   IF CH='3' THEN
      ALL:=TRUNC(S[3])
   ELSE
      ALL:=TRUNC(S[4]);
END
ELSE
   ALL:=NM;
{ WRITE ARRAY TO FILE }
REWRITE(VECTORFILE);
FOR C:=1 TO ALL DO
BEGIN
   DATA:=X[C];
   WRITE(VECTORFILE,DATA);
END;
{ CLOSE FILE }
}
CLOSE(VECTORFILE):
{ PROMPT USER THAT CALCULATIONS ARE COMPLETE }
CLRSCR;WRITELN;
IF STAT='!' THEN
  WRITELN('Random Number generation complete !')
ELSE
  WRITELN('Random Sampling complete !');
{ GIVE OPTION TO REPEAT THIS SECTION }
ANS:='N';
WHILE (ANS='Y')AND(ANS='N') DO
BEGIN
  WRITELN;WRITELN;WRITELN('Would you like to repeat the \"WHICH\" section ? (Y/N)');
  READLN(ANS);
END;
{ RETURN TO MAIN MENU }
ASSIGN(VSTAT,'VSTAT.COM');
EXECUTE(VSTAT);
END(RAND).
PROGRAM PURGEV;
{ MAIN PROGRAM STORAGE SPECIFICATIONS }
LABEL 5;
TYPE
STRING2=STRING[2];
VAR
DRIVE:STRING2;
VIEW:INTEGER;
PPURGE:STRING[8];
ANS:CHAR;
OK:BOOLEAN;
VSTAT:FILE;
VECTORFILE:FILE OF REAL;

{ MAIN PROGRAM - VARIABLE DEFINITIONS }

DRIVE - Current data disk-drive.
PPURGE - Name of vector to be purged.
ANS - User's response to yes/no questions.
OK - True if file exists.
VSTAT - File containing main menu.
VECTORFILE - File name variable.

BEGIN
ANS:='Y';
WHILE ANS='Y' DO { LOOP TO REPEAT THIS SECTION }
BEGIN
CLSCR;
{ LOCATE VECTOR TO PURGE }
REPEAT
{ CHECK FOR RETURN W/O ENTRY }
PPURGE:='';
WHILE (LENGTH(PPURGE)<1)OR(LENGTH(PPURGE)>6) DO
BEGIN
WRITELN;WRITELN;WRITELN('What is the name of the vector you want to PURGE ? ');
READLN(PPURGE);
END;
{ ATTACH DISK-DRIVE TO NAME }
PPURGE:=DRIVE+PPURGE;
{ CHECK TO SEE VECTOR EXISTS }
ASSIGN(VECTORFILE,PPURGE);
{I++}RESET(VECTORFILE);{I++}
OK:=[IOresult=0];
CLSCR;
IF NOT OK THEN
{ PRINT ERROR MESSAGE }
BEGIN
ANS:='M';
WHILE (ANS='Y') AND (ANS='N') DO
BEGIN
WRITELN;WRITELN;WRITELN('VECTOR ',PPURGE,' DOES NOT EXIST !');
WRITELN;WRITELN;WRITELN('Would you like to try again to PURGE a vector ? (Y/N)');
READLN(ANS);
END;
IF ANS='N' THEN GOTO 5;
END;
UNTIL OK;
{ CLOSE FILE }
CLOSE(VECTORFILE);
{ DOUBLE-CHECK BEFORE PURGING }
ANS:='M';
WHILE (ANS='Y') AND (ANS='N') DO
BEGIN
CLSCR;WRITELN;WRITELN;WRITELN('Are you certain you want to ERASE vector ',PPURGE,' ? (Y/N)');
READLN(ANS);
END;
IF ANS='Y' THEN
{ ERASE DESIRED VECTOR }
ERASE(VECTORFILE)
ELSE
GOTO 5;
{ PURGE ANOTHER VECTOR ? }
ANS := 'N';
WHILE (ANS = 'Y') AND (ANS = 'N') DO
BEGIN
  CLRSCR;WRITELN;WRITELN;WRITELN('Would you like to PURGE another vector ? (Y/N)');
  READLN(ANS);
END;
END;
{ RETURN TO MAIN MENU }
S:ASSIGN(VSTAT,'VSTAT.COM');
EXECUTE(VSTAT);
END(PURGEV).