The Setting up of a Computer Package

for the

Testing of Random Numbers

by

Patricia J. M. Atkinson B.Sc.

Computing Laboratory

University of St. Andrews

A thesis presented for the degree of Master of Science

(1971)

I hereby declare that this thesis has been composed by myself; that the work of which it is a record has been done by myself; and, that it has not been accepted in any previous application for any higher degree. This research concerning the testing of random numbers was undertaken from the beginning of October 1970, the date of my admission as a research student for the degree of Master of Science (M.Sc.).

ACKNOWLEDGEMENTS

I should like to record my gratitude to Professor Cole for his help and encouragement in all aspects of this work.

I am indebted to the Science Research Council for the award of a Research Scholarship 1970-1971.

My thanks are also due to Dr. Robert Erskine and his staff for their invaluable assistance and to Miss Janette Herd who typed this thesis.

CONTENTS

LIST OF DIAGRAMS AND TABLES

ABSTRACT

There are several statistical tests for evaluating the randomness of a set of numbers. However, no one test can be considered entirely sufficient. Therefore, the purpose of this work was to combine the most informative of these tests to produce a computer package for the testing of random numbers. Four random number generating functions were investigated using this package. They were found to yield very good, good, poor and bad random number sequences respectively. Hence, as no single test could give such an exact differentiation between the generators, this package is a more effective method for testing the randomness of a set of numbers.

CHAPTER I. INTRODUCTION

The interest in random numbers has increased considerably in recent years. This can be attributed to the advent of electronic computing devices and the number of problems in which they can be used.

There are a great variety of applications for random numbers. One of the most important is in the simulation of natural phenomena which is used in nuclear physics, queuing theory, traffic control, organisation of telephone systems and many others (J. Hammersley and D. Handscomb, J. Todd and O. Taussky). Another use concerns the subject of sampling where it is often better to consider every case in the particular population but frequently this is impractical due to time and cost. Hence the cases to be selected are picked at random. Random numbers are also of importance in solving numerical problems and in examining the effectiveness of algorithms and hypotheses.

In most of these applications, large samples are frequently required. As a result, there is the difficulty of obtaining a large set of random numbers quickly and easily. Many different methods have been proposed and used. These methods fall into two categories, the first of which involves the production of true random numbers and the second, the production of what are commonly called pseudo-random numbers. The first group concerns manual and physical methods such as the tossing of a dice for random numbers, electronic pulses, and the use of a table of random numbers. Pseudo-random numbers are deterministic and, therefore, are not in fact truly random. Usually each one is calculated, according to predefined rules, from the previous one. Pseudo-random numbers are mainly used in computing because, firstly, manual methods are time consuming and invariably, due to storing of tables of numbers, take up too much space, and secondly, physical methods are often not reproducible.

If pseudo-random numbers are used instead of true random numbers, then they must appear to be random and have the following characteristics: short and easy generating sequences, long periods, reproducibility and statistical acceptability. Even today, with modern techniques, it is desirable to have easy and short generating sequences because the work load on computers is for ever increasing. Long periods are essential for most problems otherwise the sequence will start repeating after a short time. This will bring in a bias and the numbers will not be random. The advantages of reproducing a sequence of numbers are in using the same data for comparisons and retesting in different ways.

This thesis is concerned with the statistical acceptability of pseudo-random numbers. No sufficient conditions exist which can be used to approve a pseudo-random number generator, since a statistical test can always be found which will not be satisfied by the generator. Pseudo-random numbers can still be used if they pass such statistical tests as are relevant to the problem under consideration. As a result, there are no set rules for the approval of a generator and the requirements change from problem to problem. Thus, in theory, one should apply a whole set of statistical tests (relevant to the application) on the generator to be used, and this should show its acceptability for the particular problem. However, in practice most generators have a general use as standard routines, therefore, they have to pass a number of standard tests. Empirical tests are applied to the numbers produced by a generator, whereas theoretical tests are applied to the actual generating function. The work reported here is the development of a system of computer subroutines for the empirical standard tests most frequently used in the statistical analysis of the randomness of a set of numbers. The system has been designed for workers of all disciplines

and made easy so that new procedures may be added. Every care has been taken to avoid fixing limitations, so that the system can be employed over as wide a range of problems as possible. However, a few concessions have had to be made in order to keep within the boundaries of the storage facilities of the computer. It is hoped that these will not restrict the user in the implementation of this package .
(A.B. Forsythe)

CHAPTER II. STATISTICAL THEORY

The set of empirical tests considered in this work fall primarily into two groups. The first one concerns those tests which use the Chi-squared, χ^2, distribution as a means of comparison. The second group contains those tests which use the Kolmogorov-Smirnov, KS, test for comparing the difference between the empirical and theoretical distributions of a set of numbers. It is assumed that each empirical test is applied to a sequence of real numbers which are uniformly distributed between zero and one. Some tests are designed more for integer-valued sequences. Therefore, the numbers are multiplied by a base, say D, which can vary, and the integer parts are then taken.

The Chi-squared Test

For the χ^2 test, it is assumed that there are n independent observations and that each observation falls into one of K classes. If p_i is the probability that each observation falls into class i, and x_i is the number of observations that fall into class i, then the statistic, V, where

$$V = \sum_{1 \le i \le k} \left\{ \frac{(x_i - np_i)^2}{np_i} \right\}$$

is called the "chi-square" statistic. This is the most important chi-squared statistic and is the sum of the squares of the differences between the observed values and the expected values in each class divided by a weighting factor, the expected frequency.

$$V = \sum_{1 \le i \le k} \left\{ \frac{x_i^2 + n^2 p_i^2 - 2np_i x_i}{np_i} \right\}$$

$$= \sum_{1 \le i \le k} \frac{x_i^2}{np_i} + n \sum_{1 \le i \le k} p_i - 2 \sum_{1 \le i \le k} x_i$$

but $\sum_i p_i = 1$ $\sum_i x_i = n$

Thus,

$$V = \sum_i \frac{x_i^2}{np_i} + n - 2n$$

$$= \frac{1}{n} \sum_i \frac{x_i^2}{p_i} - n$$

This is a better form of the χ^2 statistic which is easier for computation. Theoretical values may be obtained from the χ^2 distribution. These values vary according to the number of degrees of freedom. The degrees of freedom are the number of independent variables which in this case is one less than the number of categories, (k-1), as

$$x_1 + x_2 + \dots + x_k = n$$

thus,

$$x_1 = n - x_2 - x_3 - \dots - x_k$$

Thus the probability, P, that a random variable, distributed according to the χ^2 distribution and with the same number of degrees of freedom, (k-1), as V, is less than or equal to V can be calculated.

When V is either very small or very large in comparison with the theoretical value, it has to be viewed with suspicion as to having a significant departure from random behaviour. Usually if $P > 0.99$ or <0.01, the numbers are considered not sufficiently random and are rejected. If $0.95 < P \leq 0.99$ or $0.01 \leq P < 0.05$ the numbers are suspect of not being random. When P is between 0.90 and 0.95 or 0.05 and 0.1, the numbers are considered to be slightly suspect of not being random. Otherwise the numbers are considered to be sufficiently random.

The actual calculation of theoretical χ^2 values is difficult. Therefore, in most cases, approximations must be used which are only valid if n is large. As n becomes larger, a bias in the numbers to

be tested would be detected but local non-random behaviour would probably be smoothed out. Thus, several tests should be made for different values of n. In the majority of cases a sufficient guide to the value of n required is that it should be large enough so that the expected frequencies are greater than, or equal to five. This rule is followed for the choice of n in the subsequent tests unless otherwise stated.
(M. Fisz, B.W. Lindgren)

The Frequency Test

This test discerns whether the numbers are uniformly distributed with an acceptable probability. The number span, D, is divided into equal non-overlapping intervals. A tally is taken of the quantity of numbers in each interval over n observations and then the χ^2 test is applied. If there are y intervals then the number of degrees of freedom will be (y-1) and the probability, p_i, of a number falling into the ith category is $\frac{1}{y}$.

The Serial Test

The serial test is an extension of the frequency test. It investigates whether or not pairs of successive numbers are independently and uniformly distributed. The occurrences for every pair of numbers is counted over n pairs of observations and the χ^2 test applied. The number of categories is D^2 and the probability of a pair of numbers being in a particular category is $\frac{1}{D^2}$. This test can be generalised to test triples, quadruples, etcetera as well as pairs of numbers but for these cases D must be made smaller to reduce the number of intervals to a manageable size. In practice, triples, etcetera should be investigated using other tests such as the poker, maximum or sum tests. (I.J. Good)

The Poker Test

This test considers n groups of k successive integers and a count is kept of the number of distinct values in each set of k numbers. A distinct value of r signifies that in a group of k numbers, then, regardless of order, there are r different numbers. For example when k=5 there are five categories and a group of five numbers must belong to one of them. These categories are as follows:

Five different	=	abcde	all different
Four different	=	aabcd	one pair
Three different	=	aaabc	three the same and the remaining two different
		aabbc	two pairs and the remaining one different
Two different	=	aaabb	one pair and three the same
		aaaab	four the same
One different	=	aaaaa	all the same

The probability of a group of k numbers having r different is

$$p_r = \frac{d(d-1) \ldots\ldots (d-r+1)}{d^k} \left\{ {k \atop r} \right\}$$

where $\left\{ {k \atop r} \right\}$ are Stirling's numbers of the second kind. The χ^2 test can be applied using the above probabilities and the number of degrees of freedom is one less than the number of different categories. That is if k=4, then the number of different categories is four since

Four different	=	abcd
Three different	=	aabc
Two different	=	aaab
		aabb
One different	=	aaaa

Thus the number of degrees of freedom is 3.

Generally, because the probabilities are low when r=1 or 2 a few categories of low probability are combined before the χ^2 test is applied. (D.E. Knuth, Vol. 1, 1968; M. Abramowitz and I.A. Stegun)

The Runs Test

A run of length r is either a monotone increasing sequence or a monotone decreasing sequence of numbers. Thus a run up of length r is defined as

$$\ldots\ldots x_i > x_{i+1} < \ldots\ldots < x_{i+r-1} < x_{i+r} > x_{i+r+1} \ldots\ldots$$

and a run down of length r is defined as

$$\ldots\ldots x_i < x_{i+1} > x_{i+2} > \ldots\ldots > x_{i+r-1} > x_{i+r} < x_{i+r+1} \ldots\ldots$$

Since adjacent runs are not independent, a simple χ^2 test cannot be applied. If the statistic V, where

$$V = \frac{1}{n} \sum_{1 \le i, j \le 6} \{\text{COUNT}(i) - nb_i\}\{\text{COUNT}(j) - nb_j\}a_{ij} \quad *$$

is calculated, it can be used for calculations in the χ^2 test for n runs up or down with six degree of freedom. The mean values of the runs of exactly length r, where r is less than or equal to six, are the b_j coefficients which can be calculated comparatively easily. Calculation of the coefficients, a_{ij}, is more difficult and was effected using the inverted covariance matrix of the number of runs of exactly length r. Thus, an approximation for the a_{ij}'s has been calculated on the assumption that n is large, that is greater than or equal to 4,000 and the coefficients used are given below.

* a_{ij} and b_i are coefficients, constant for a particular value of n .

$$a_{ij} = \begin{bmatrix} 4529.4 & 9044.9 & 13568 & 18091 & 22615 & 27892 \\ 9044.9 & 18097 & 27139 & 36187 & 45234 & 55789 \\ 13568 & 27139 & 40721 & 54281 & 67852 & 83685 \\ 18091 & 36187 & 54281 & 72414 & 90470 & 111580 \\ 22615 & 45234 & 67852 & 90470 & 113262 & 139476 \\ 27892 & 55789 & 83685 & 111580 & 139476 & 172860 \end{bmatrix}$$

$$(b_1\ b_2\ b_3\ b_4\ b_5\ b_6) = (\tfrac{1}{6}\ \tfrac{5}{24}\ \tfrac{11}{120}\ \tfrac{19}{720}\ \tfrac{29}{5040}\ \tfrac{1}{840})$$

Each time there is a run of length i one is added to COUNT(i), but if $i \geq T$, where T is the maximum length of a run, then one is added to COUNT(T). In this particular test, the statistic and coefficients have been calculated for T=6. (D.E. Knuth, Vol. 2, 1969)

The Gap Tests

There are two kinds of gap test, the former being applied to real numbers and the latter to digits.

In the first test, if a and b are two real numbers with

$$o \leq a < b \leq 1$$

Then, if

$x_j, x_{j+1}, \ldots . x_{j+r-1}$ are not between a and b

but x_{j+r} is, this sequence of numbers is considered to be a gap of length r. Whenever a gap of length i is found one is added to COUNT(i) but if $i \geq t$, where t is the maximum length of a gap, then one is added to COUNT(t). The values of t and n, where n is the number of gaps to be found, are chosen so that each COUNT(i) is expected to be five or more.

If $p = \text{prob}\{a \leq x_i < b\}$

then $p_o = \text{prob}\{\text{run of length } 0\} = p$

$p_1 = \text{prob}\{\text{run of length } 1\} = (1-p)p$

$p_2 = \text{prob}\{\text{run of length } 2\} = (1-p)^2 p$

$\vdots$

$$p_{t-1} = \text{prob}\{\text{run of length } t-1\} = (1-p)^{t-1}p$$

$$p_t = \text{prob}\{\text{runs of length } \geq t\} = 1-p-p(1-p)-p(1-p)^2-\ldots-p(1-p)^{t-1} = (1-p)^t$$

Therefore with the above probabilities, the χ^2 test is performed with t degrees of freedom. When a=0 and b=0.5, the test is usually called "runs below the mean" and similarly, when a=0.5 and b=1.0 the test is known as "runs above the mean".

In the second test, a gap is defined as the distance to the next occurrence of a particular digit. Thus the probability of a gap of length r is

$$p_r = \{1 - \frac{1}{D}\}^{r-1}\frac{1}{D} \qquad r=1,2,\ldots\ldots\ldots$$

where D is the base of the number system or number span to be tested. Thus for this distribution the

$$\text{mean} = \sum_{r=1}^{\infty} r\,(1 - \frac{1}{D})^{r-1}\frac{1}{D} = D$$

and the

$$\text{variance} = \left[\sum_{r=1}^{\infty} r^2\,(1 - \frac{1}{D})^{r-1}\frac{1}{D}\right] - D^2 = D(D-1)$$

The length of each gap of a particular digit is noted and the mean $(\overline{x})$ and the variance (s^2) are calculated. If the number (n) of gaps is large, that is greater than fifty, and the numbers are random, then by the laws of probability, the mean and the variance have the following distributions.

$$\overline{x} \sim N(D, \frac{D(D-1)}{n})$$

$$s^2 \sim N(\frac{n-1}{n}D(D-1),\ 2\frac{(n-1)}{n^2}\ D^2(D-1)^2)$$

Thus, for the mean and variance of the sample not to differ significantly from the hypothetical values and as a result be sufficiently random then,

$$\left| \frac{\bar{x} - D}{\frac{\sqrt{D(D-1)}}{n}} \right| \leq t_\alpha$$

$$\left| \frac{s^2 - \frac{n-1}{n} D(D-1)}{\frac{\sqrt{2(n-1)} \quad D(D-1)}{n}} \right| \leq t_\alpha$$

where α = level of significance.

If $z \cap N(0,1)$ then

$$\text{prob}\{|z| \leq t_\alpha\} = 1-\alpha$$

t_α may be chosen for a one or two sided significance test (M. Fisz) from the tables for the normal distribution function.

These inequalities imply that the confidence limits for $\bar{x}$ and s^2 are

$$D - t_\alpha \frac{\sqrt{D(D-1)}}{n} \leq \bar{x} \leq D + t_\alpha \frac{\sqrt{D(D-1)}}{n}$$

and

$$\left[(n-1) - \sqrt{2(n-1)}\ t_\alpha\right] D(D-1) \leq n\ s^2 \leq \left[(n-1) + \sqrt{2(n-1)}\ t_\alpha\right] D(D-1)$$

The mean and the variance of the length of gaps should be calculated and tested for each digit. (F. Gruenberger and G. Jaffrey, B. Jansson 1966)

D^2 Test

This test considers the distribution of numbers over the unit square. It is assumed that the numbers lie between nought and one. Four consecutive numbers are taken as the coordinates of two points in the unit square. Then the square of the distance (d^2) between the two points $((x_1, y_1), (x_2, y_2))$ is calculated.

$$d^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$$

If the numbers are rectangularly distributed, then the distribution function of d^2 is as follows:

$$\text{prob}\{d^2 < B^2\} = \pi B^2 - 8\frac{B^3}{3} + B^4 \qquad \text{for } B^2 < 1.0$$

$$= \frac{1}{3} + (\pi-2)B^2 + 4(B^2-1)^{\frac{1}{2}} + \frac{8}{3}(B^2-1)^{3/2}$$

$$- \frac{B^4}{2} - 4B^2\sec^{-1}B \qquad \text{for } B^2 \geq 1.0$$

(B. Wilson)

The n results for d^2 are tabulated into twenty classes, 0.0 up to but not including 0.1, 0.1 to 0.2, ..., 1.9 to 2.0. Then a χ^2 test is performed with these results, the above probabilities and 19 degrees of freedom. (F. Gruenberger and G. Jaffrey)

Kolmogorov-Smirnov Test (KS)

Some random quantities can take an infinite number of values. When this occurs, the numbers are said to form a continuous function, say X. The distribution function of X is

$$F(x) = \text{prob}\{X < x\}$$

The empirical distribution function, Fn(x), of X is found from n independent observations X_1, X_2, ... ,X_n as follows:

$$Fn(x) = \frac{\text{number of } X_1, X_2, \ldots, X_n < x}{n}$$

Thus, as n gets larger, Fn(x) should be a better approximation to F(x).

The KS test is used when F(x) is continuous and it is based on the numerical difference between F(x) and Fn(x). A bad random number generator will give an empirical distribution function which is not sufficiently close to F(x). To measure the proximity of Fn(x) to F(x), the statistics

kn^+ and kn^- should be calculated, where

$$kn^+ = \text{maximum deviation between } Fn(x) \text{ and } F(x) \text{ when } Fn(x) > F(x)$$

$$= \sqrt{n} \max_{-\infty<x<\infty} \{Fn(x) - F(x)\}$$

$$kn^- = \text{maximum deviation between } Fn(x) \text{ and } F(x) \text{ when } Fn(x) < F(x)$$

$$= \sqrt{n} \max_{-\infty<x<\infty} \{F(x) - Fn(x)\}$$

The probability of having a larger value of kn^+ and of kn^- can either be calculated or looked up in a percentile table. The bounds of acceptability for the probabilities are the same as for the χ^2 test. Therefore, if the probability is too high or too low for either kn^+ or kn^- then the numbers are not considered to be sufficiently random.

The value of n should not be too high because local non-random behaviour could be smoothed out, but, on the other hand, it should not be too low as then, there would not be sufficient information. A reasonable value of n would appear to be in the region of 1,000.

If several calculations of kn^+ and kn^- are made on different parts of the random sequence, then a KS test can again be applied to these numbers. This gives a better idea of the behaviour of the original numbers, and any local non-random behaviour is not smoothed out. Similarly a KS test can be applied after a χ^2 test. If several χ^2 values have been calculated the KS test can be used to compare these values with the actual χ^2 distribution function. (W. Feller)

Maximum Test

This test considers the distribution of the maximum of t numbers. Hence the

$$\text{prob}\{\max(X_1, X_2, \dots, X_t) < x\} = \text{prob}\{X_1 < x, X_2 < x, \dots, X_t < x\}$$

$$= x.x.x.........x$$

$$= x^t$$

since the X_i's are independent and uniformly distributed. Thus t consecutive numbers are taken and the maximum, U_i, is found. This is repeated n times to give U_0, U_1, ... , U_{n-1}. The KS test is now applied to compare the empirical distribution function of U_0, ... , U_{n-1} with the following theoretical distribution function.

$$F(x) = x^t$$

Minimum Test

Similarly, the minimum test considers the distribution of the minimum, V_i, of t numbers. Therefore using the same procedure as above

$$\begin{aligned} \text{prob}\{\min(X_1, X_2, \ldots , X_t) < x\} &= 1 - \text{prob}\{\min(X_1, X_2, \ldots , X_t) \\ &\quad \geq x\} \\ &= 1 - \text{prob}\{X_1 \geq x, X_2 \geq x, \ldots, X_t \geq x\} \\ &= 1 - (1 - x)^t \end{aligned}$$

Thus, in this case the KS test is applied to the empirical distribution function of V_0, V_1, ... , V_{n-1} by comparing it with the distribution function given below:

$$F(x) = 1 - (1 - x)^t$$

Sum Test

This test considers the distribution of the sum of t numbers. Because the numbers are assumed to be uniformly distributed, their probability density function is

$$f_1(x) = \begin{cases} 1 & 0<x<1 \\ 0 & \text{otherwise} \end{cases}$$

If N_1, N_2, ... are independent and uniform distributed variables over the interval (0,1), then the sum, $N_1 + N_2 + ... + N_t$ is confined to the interval (0,t). Therefore, let $f_n(x)$ denote the probability density function of $N_1 + ... + N_t$, then by probability theory

$$f_{i+1}(x) = \int_{-\infty}^{\infty} f_1(x-t)\, f_i(t)\, dt$$

$$= \int_{x-1}^{x} f_i(t)\, dt$$

This implies that

$$f_2(x) = \begin{cases} x & 0<x<1 \\ x - 2(x-1) & 1<x<2 \end{cases}$$

$$f_3(x) = \begin{cases} \frac{x^2}{2} & 0<x<1 \\ \frac{1}{2}\left[x^2 - 3(x-1)^2\right] & 1<x<2 \\ \frac{1}{2}\left[x^2 - 3(x-1)^2 + 3(x-2)^2\right] & 2<x<3 \end{cases}$$

and in general

$$[1] \quad f_i(x) = \frac{1}{(i-1)!}\left[x^{i-1} - \binom{i}{1}(x-1)^{i-1} + \binom{i}{2}(x-2)^{i-1} - .....\right]$$

The summation continues so long as the arguments x, x-1, x-2, ... are positive. The mean and variance of this distribution are $\frac{i}{2}$ and $\frac{i}{12}$ respectively. Thus the density function of the standardised sum is

$$\sqrt{\frac{i}{12}}\; f_i\left(\frac{i}{2} + x\sqrt{\frac{i}{12}}\right)$$

and as i increases this rapidly approaches the normal density function

$$\frac{1}{\sqrt{2\pi}}\, e^{-\frac{x^2}{2}} \qquad \text{(H. Cramér)}$$

Therefore the sum of t numbers is calculated and this is repeated n times to give U_0, U_1, ..., U_{n-1}. Then the KS test is applied which compares the empirical distribution function of U_0, U_1, ... , U_{n-1} with the theoretical distribution function calculated from the above density function. The theoretical distribution varies according to the value of t. For low values of t (≤ 5), the compound uniform distribution, [1], should be used, but for the remaining values of t, the normal distribution is an adequate approximation and can be used. (D.Y. Downham and F. Roberts)

Some tests may appear to be more sensitive than others. The runs test seems to be the most sensitive, whereas the frequency and serial tests the least since they are satisfied by most generators. This does not imply that only the runs test should be applied since it does not test all the aspects of a sequence of numbers. Hence the package has been written to include all the previously mentioned tests.
(B. Jansson 1966, K.D. Tocher, A. Van Gelder, M.D. MacLaren and G. Marsaglia, S. Gorenstein)

CHAPTER III. GENERAL APPROACH OF THE PACKAGE

The tests mentioned in the previous chapter were all written in the form of subroutines which are called from the main program. They were divided into groups and, when working satisfactorily, these subroutines were stored in a library on disc. An overlay system was then implemented which is described later, in Chapter four.

The numbers to be tested are read by default from an unformatted sequential file, but if required, the data can be read either under format control or obtained from a user supplied generator. If the user wishes to read the numbers under format, then this is done either by supplying the user's own format or using the format provided by the system. In most cases when a generator is used to produce numbers a starting value is required. Thus the facility to read in a starting value is available. In Chapter 5, a user is shown how to invoke these options, whereas in Chapter 4, the methods of implementation are described.

To use the KS test, the empirical distribution has to be compared with a theoretical distribution, thus, there is the facility for the user to provide his own distribution function. Also, the data control parameters may be read under a different format than that provided by the system. This facility is employed by supplying a user's format according to the rules given in Chapter 5. The way in which these options are implemented is described in Chapter 4 along with the method used to enlarge the package by adding more tests.

CHAPTER IV. PROGRAMMING TECHNIQUES

This chapter is split into three interrelated sections. It is intended to explain the main theory behind the programming of this package with the aid of the subroutines listed at the end of the thesis in Appendix 1.

The first section is concerned with the overlay system and deals with the setting up of the package. The second section concerns the subroutines themselves and includes how they are stored, how new ones are added together with some of their interesting features. The remaining section deals with the way in which various options have been made available to the user. In all sections the job control language is for an IBM 360/44 computer running under the 44PS system. For a different computer or system the job control statements would have to be changed according to the appropriate rules.

Overlay System

The main reason that an overlay system is used here is to save on space in the problem program area in main core store. All programs to be executed under system control must first be processed by the linkage editor. The linkage editor program converts assembler and compiler output modules into a form of one or more phases which are suitable for loading and execution. A phase is that portion of a program which can be loaded into main storage by a single LOAD call. The size and the particular subprograms of a phase are specified by the programmer with linkage editing control statements (diagram 1). A program may use several phases, the only limition being the size of the problem program area. Hence, in this way the maximum size of a phase is fixed. The phases are stored in the phase library on SDSABS by using the KEEP option in the RLNKEDT statement (diagram 3).

For this package a root phase overlay system is used. One of the phases is designated the root phase and this remains in the problem program area throughout the execution of the whole program (diagram 2). The other phases, that is the subordinate phases, are loaded into the problem program area when they are needed.

DIAGRAM 1

The following example illustrates the various linkage editing control statements that define the contents of certain phases and their origins.

```
PHASE      EXAMPLE1, PAR
INCLUDE    A1
INCLUDE    A2
PHASE      EXAMPLE2, PAR
INCLUDE    B1
PHASE      EXAMPLE3, PAR
INCLUDE    C1
```

where

EXAMPLE1, EXAMPLE2, EXAMPLE3 — are the phase names which can be up to eight alphameric characters in length with the first one being alphabetic.

PAR =

- ROOT — specifies the origin of the root phase.
- * — sets the origin of the subordinate phase to the first location following the most recently processed phase.
- 'phasename' — sets the origin of the current phase equal to the origin of the phase whose name is specified.

A1, A2, B1, C1 — are the subroutine names which are to be included in a particular phase.

DIAGRAM 2

The previous example could be written as follows

```
PHASE      EXAMPLE1, ROOT
INCLUDE    A1
INCLUDE    A2
PHASE      EXAMPLE2, *
INCLUDE    B1
PHASE      EXAMPLE3, EXAMPLE2
INCLUDE    C1
```

which would give the following overlay system.

ROOT A1	A2
EXAMPLE2 B1	EXAMPLE3 C1

Therefore a subordinate phase may overlay a previously loaded subordinate phase but, in general, it must not overlay the root phase.

The loading of phases is controlled by the main or calling program which is effected by the statement

CALL   LOAD ('phasename')

However, control returns to the next statement in the main or calling program. Once a phase has been loaded, any subprogram within that phase may be utilised using the statement

CALL   'subprogram name'

DIAGRAM 3

```
//   EXEC      RLNKEDT(KEEP,SYS002,NOAUTO)
     PHASE     APTEST,ROOT
     INCLUDE   APMAIN,R,2
     INCLUDE   NDTR,R,2
     INCLUDE   APTR,R,2
     INCLUDE   DATA1,R,2
     INCLUDE   DATA2,R,2
     INCLUDE   GETFMT.R,2
     INCLUDE   READ2,R,2
     INCLUDE   READ1,R,2
     INCLUDE   IBCOM#,R
     INCLUDE   FIOCS#,R
     INCLUDE   USEROPT,R
     INCLUDE   UNITAB#,R
     INCLUDE   LOAD,R
     INCLUDE   EXP,R
     INCLUDE   FRXPI#,R
     INCLUDE   FRXPR#,R
     INCLUDE   SQRT,R
     INCLUDE   ARCOS,R
     INCLUDE   DLOG,R
     INCLUDE   DEXP,R
     INCLUDE   DSQRT,R
     INCLUDE   ALOG,R
     INCLUDE   KLOCK,R
     INCLUDE   ATAN,R
     INCLUDE   AMAX1,R
     PHASE     APCHI,*
     INCLUDE   FREQ,R,2
     INCLUDE   POKER,R,2
     INCLUDE   SERIAL,R,2
     INCLUDE   APSN,R
     INCLUDE   GAP,R,2
     INCLUDE   DSQUR,R,2
     INCLUDE   RUNS,R,2
     INCLUDE   CDTR,R,2
     INCLUDE   DLGAM,R,2
     PHASE     APKMSM,APCHI
     INCLUDE   SUM,R,2
     INCLUDE   MAX,R,2
     INCLUDE   MIN,R,2
     INCLUDE   AP261,R,2
     INCLUDE   AP262,R,2
     INCLUDE   AP263,R,2
     INCLUDE   AP264,R,2
     INCLUDE   AP265,R,2
     INCLUDE   AP266,R,2
     INCLUDE   AP267,R,2
     INCLUDE   USER,R,2
     INCLUDE   KOLMO,R,2
     INCLUDE   SMIRN,R,2
     PHASE     APOTHR,APKMSM
     INCLUDE   GAPRD,R,2
```

DIAGRAM 4

APTEST (root phase)	
	APMAIN
	NDTR
	APTR
	DATA1
	:
	ATAN
	AMAX1

APCHI	APKMSM	APOTHR
FREQ	SUM	GAPRD
SERIAL	MIN	
POKER	MAX	
APSN	AP261	
	:	
GAP	AP267	
DSQUR	USER	
RUNS	KOLMO	
CDTR	SMIRN	
DLGAM		

The linkage editing control statements used in setting up the overlay system of this package are illustrated in diagram 3. The form of the overlay system is shown in diagram 4. NOAUTO is specified in the RLNKEDT statement. This causes automatic searching of the module library of names matching unresolved external references for the entire linkage editing job to be suppressed. For example, when a subprogram calls another subprogram, the module library is automatically searched for this called subprogram or external reference. The external references may be library or system subprograms, or subprograms provided by the user. When NOAUTO is specified,all external references must be included in the linkage editors control statements. Therefore, the library subprograms FIOCS, USEROPT, SQRT etcetera have been included in the linkage editing control statements which are shown in diagram 3 for this package. SYS002 is specified in the RLNEDT statement to inform the linkage editor on which unit the user supplied subprograms have been stored.

For further information on overlay systems refer to

IBM C28 - 6812

IBM C28 - 6813

Subroutines

The main program, which handles the loading of phases, formats, data etcetera together with all the user's subroutines requested in this package, are stored in a library on disc. Space was obtained by using the following statements

```
//'jobname'   JOB, ALLOC
//                 ALLOC   APLIB,191='SA45VI',350,30
//                 LABEL   360,99366,RECLEN=72
/*
/&
```

These statements create a library, called APLIB, on disc, SA45VI, through the disc drive, 191, with 350 blocks of storage and 30 entries into the directorial set. The LABEL statement specifies, firstly, that the block length is 360 bytes, secondly, when this data set may be deleted and finally, the size of a logical record.

To add a new member into the library, for example a subroutine called TEST, the following statements are used

```
//'jobname'   JOB, 'user's name'     'time of job'
//SYS000      ACCESS APLIB(TEST), 191='SA45VI', NEW
//            EXEC FORTRAN
             :
             subroutine's cards
             :
/*
/&
```

To delete the library or a subroutine from the library the following statements should be used

```
//     JOB, DELETE
//     ACCESS APLIB, 191='SA45VI'
//     DELETE APLIB/APLIB(TEST)
/*
/&
```

To condense or investigate the contents of the library, the card

```
//     DELETE  ________
```

in the previous JOB should be replaced by

```
//     CONDENSE APLIB
```

and

```
//     EXEC CLSDSREL
```

respectively. Thus a new subroutine may be added to the library at any time, the only restriction being that there is enough space and there are sufficient directorial entries. If either of these restrictions is found to be violated, the library may be deleted and then recreated with a larger number of blocks or directorial entries or both. This is implemented by increasing the appropriate parameters in the ALLOC statement.

Since this is an overlay system with more than one phase, then if any subroutines are changed or added into the library, the pointers for the phasing will probably be incorrect. Therefore, the whole program has to be relinkage edited with the changes. If any new subroutines are to be included in the library, extra linkage editing control statements must also be added before the linkage editing takes place. These control statements show how the phasing of the new subroutines is effected and include any additional library subprograms used.

At the moment, there are ten statistical tests. When a new test, which may include several subroutines, is to be added it must be stored in the library APLIB. Also, various statements must be changed in the main program (subroutine APMAIN) and several extra ones included. In Program 1, line 39 is

```
16    GO TO(20,30,40,50,60,70,80,90,92,94),K
```

where K is the code given to a test. This statement branches to the statement with label in position K within the brackets. That is, if K=3 the control is passed to statement 40. Any extra tests should be coded as eleven onwards and line 39 becomes

```
16    GO TO(20,30,40,50,60,70,80,90,92,94,IX,IY,...),K
```

The statements labelled IX,IY,... should be inserted before the statement labelled 180 as follows:-

```
IX      IF(FLAG.EQ.'phasenumber')GO TO IX+5
        CALL LOAD ('phasename') ——This loads the phase required
                                  for the test that has been requested.
IX+5    FLAG = 'phasenumber'
        CALL    'name of test'  ——This calls the test requested.
        GO TO 5
             :
IY           :
```

where IX is any integer, for numbering statements, which must be greater than 95 but not equal to either 180 or 190.

An index (FLAG) is given to each phase and these are as follows:

Phase APCHI	FLAG=1
Phase APKMSM	FLAG=2
Phase APOTHR	FLAG=3

If any more phases are to be added, then their flag numbers must be greater than three. A test is in a particular phase, thus if the FLAG is equal to that phasenumber, then the phase is already loaded into store. Otherwise the phase still has to be loaded and once that has been accomplished, the FLAG is set to that phasenumber. Thus the FLAG shows which phase is in store.

To apply the KS test, a statistical test calls the subroutine KOLMO. This subroutine was initially an IBM scientifc subroutine, but it has been adapted for use in this package. KOLMO compares an empirical distribution with a theoretical distribution by using the respective distribution functions. The theoretical distribution functions provided by KOLMO are from the following distributions.

Normal

Exponential

Cauchy

Uniform

Distribution for the maximum of t uniform random numbers

Distribution for the minimum of t uniform random numbers

Distribution for the sum of 2 uniform random numbers

Distribution for the sum of 3 uniform random numbers

Distribution for the sum of 4 uniform random numbers

Distribution for the sum of 5 uniform random numbers

Distribution for the sum of 6 uniform random numbers

User supplied

The same procedure for relinkage editing, when adding more theoretical distributions, is followed as before. This will put the subroutines in the package permanently. The statements that must be altered and added for the additional subroutines are as follows:-

In Program 3 line 184 is

```
GO TO(30,32,36,38,42,44,46,48),ISIN
```

where ISIN is the index for the particular subroutine requested in a test. Any new subroutines must be indexed by 9.0 onwards and line 184 becomes

```
GO TO(30,32,36,38,42,44,46,48,IW,IZ,...),ISIN
```

The following statements should be included immediately preceeding the statement labelled 48.

```
IW      CALL AP269(X(J),Y,IER)
        GO TO 50
IZ      CALL AP2610(X(J),Y,IER)
        GO TO 50
        .
        .
        .
48      IER=1
        CALL USER(X(J),Y,IER)
50      .
        .
        .
```

The number of statements depends on the number of subroutines to be added. The USER subroutine is a dummy subroutine to enable the user to supply his own subroutine, which is described in a later section under USER step. Then, if a user wishes to apply a test in this package to a set of numbers, in most cases, only two data cards have to be supplied per test. The first card is used for changing the system's formats for reading. The second card contains the parameters for the test which is to be applied. These are: the number of trials, how the numbers are obtained, etcetera and are described in the next chapter. If the user wishes to obtain his numbers from a generator, a third card may be added which will provide an integer starting value for the sequence. However, the numbers may be read from cards and these must then be included after the second data card. If the format provided by the system is not adequate, the user may supply an appropriate one. By default the numbers to be tested are read from an unformatted file on disc, with 82 numbers per record.

APMAIN (Program 1)

APMAIN reads the first card and selects the format to read the next card. Having read the latter card,it uses these parameters to load the correct phase into store and then calls the requested test. Control is always passed back to APMAIN at the end of a test after which the next set of cards for the succeeding test are read.

Subroutine Poker (Program 9)

In applying the POKER test, M numbers are considered at one time. These will be obtained from the subroutine, DATA1, and stored in an array, A, of dimension, M. This array of numbers is then sorted into descending order using a temporary array.

$$A(1) \geq A(2) \geq \ldots\ldots \geq A(M-1) \geq A(M)$$

To ascertain how many different numbers are present in this group, A(M) is compared with A(M-1). If they are not equal then there must be at least two different numbers in this group.

Otherwise they are equal as

$$A(M) \leq A(M-1)$$

This procedure is repeated until A(1) is compared with A(2) and a count is kept for the total of different numbers, say r. Then one is added to COUNT(r) and another M numbers are considered. The above process is repeated the required number of times (N). The χ^2 statistic is then calculated in the usual way with the probabilities as given in Chapter 2. These probabilities are calculated using the subroutine APSN to obtain Stirling's numbers for the value M. The subroutine CDTR, which is an IBM scientific subroutine used for calculating the probability of a worse value of the χ^2 statistic, is then called. Using this probability, the randomness of the numbers which have been tested may be deduced. The subroutine APTR does this automatically according to the rules given in Chapter 2 for the χ^2 statistic.

Subroutine FREQ (Program 7)

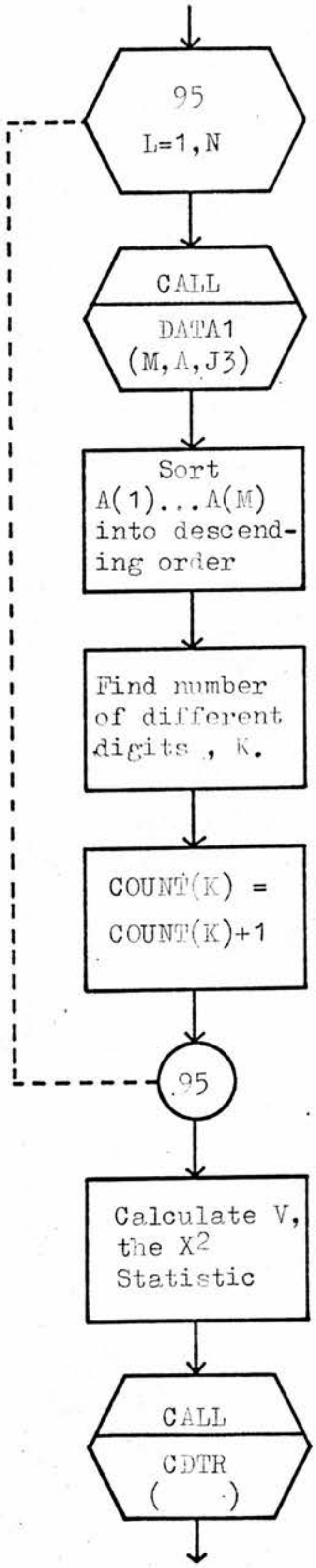
In the subroutine, FREQ, used to apply the frequency test, one number, which is obtained from the subroutine DATA1, is considered at a time. A count is kept of the number of occurrences of each integer over, say n observations. The χ^2 statistic is then calculated with the probabilities as given in Chapter 2. Then the subroutine CDTR is called for calculating the probabilities of a worse value of the χ^2 statistic. The same procedure is followed as for the poker test.

Subroutine SERIAL (Program 8)

The subroutine SERIAL, used in applying the serial test, is similar to the subroutine FREQ. The main difference is that the numbers are considered in pairs. Again a count is kept of the number

DIAGRAM (5)

FLOW DIAGRAM
for the
POKER TEST



of occurrences of each pair of integers and the same procedure is followed as before.

Subroutine MAX (Program 14)

When using the subroutine KOLMO, an array of numbers is passed to it for comparison with a given distribution function. Therefore, for the maximum test, if T numbers are required in each trial the first of these numbers is assumed to be the maximum and it is put into an element of an array, V, say V(1). A second number, U, is obtained and is compared with V(1). If U is greater than V(1) then V(1) is assigned the value of U, otherwise V(1) remains unchanged. This process is repeated until all the remaining (T-2) numbers have been compared with V(1) and thus the maximum value of the T numbers is in V(1). Other sets of T numbers are considered until the value of V(N), where N is the number of trials in the test, is found. The subroutine KOLMO is then called to compare the empirical and theoretical distributions for the array V using the KS test. The subroutine KOLMO calculates the probability of a statistic, which has the given theoretical distribution, of having a larger value than the largest value of the difference between the empirical and theoretical distribution function. Thus, using this probability, the randomness of the numbers tested can be deduced. The subroutine APTR does this automatically according to the rules given in Chapter 2 on the KS test.

The subroutine MIN, for applying the minimum test, and the subroutine SUM, for applying the sum test, are similar in application to that of the maximum test. The differences can be easily seen from Program 15 and Program 16 in Appendix 1 at the end of this thesis.

DIAGRAM (6)

FLOW DIAGRAM
for the
SERIAL TEST

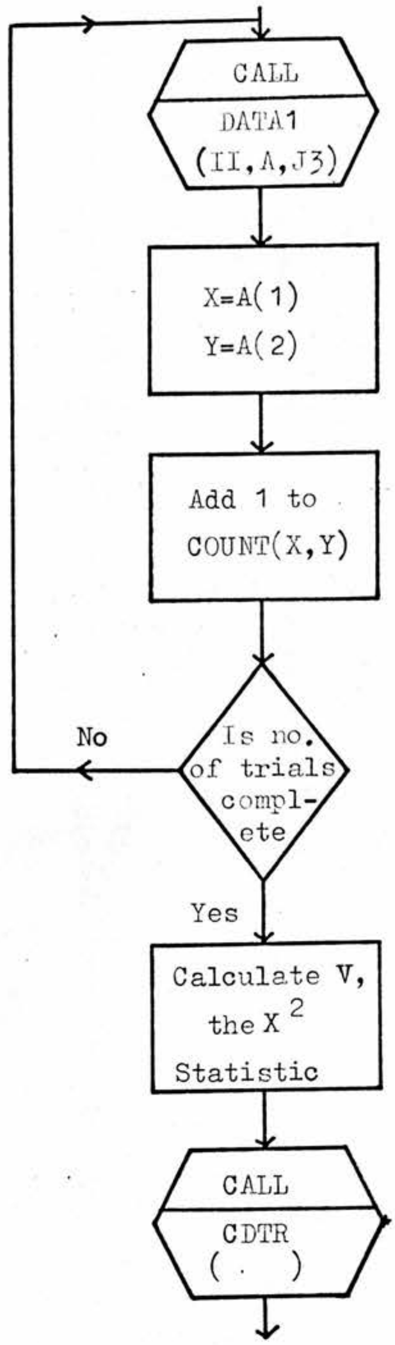
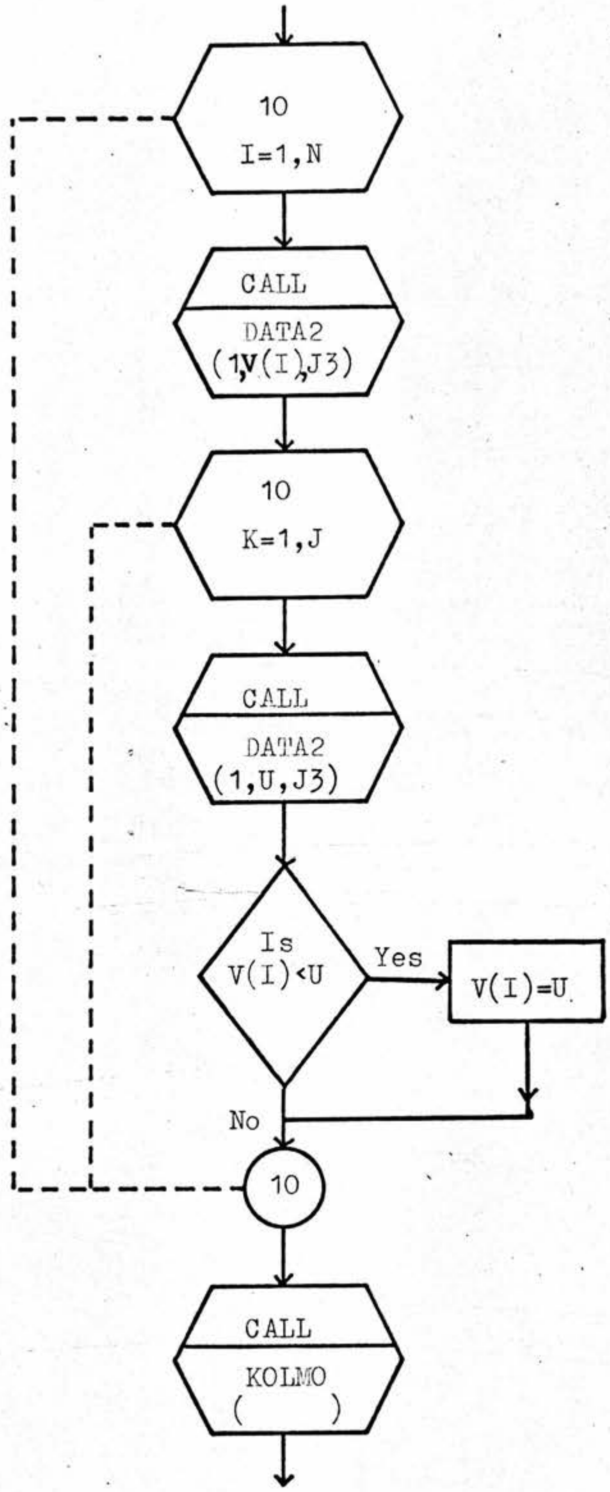


DIAGRAM (7)

FLOW DIAGRAM
for the
MAXIMUM TEST



Subroutine DSQUR (Program 12)

The D^2 test deals with two pairs of numbers at each trial. These numbers are considered to be the coordinates of two points in the unit square and thus the squared distance, D2, between them is calculated for each trial. Every value of D2 can fall into one of the twenty catergories, 0.0 to 0.1 up to 1.9 to 2.0. Thus, a count is kept for the number of times a value of D2 falls into each category. The χ^2 statistic is then calculated in the usual way with the probabilities as given in Chapter 2. The same procedure is then followed as for the poker test.

Subroutine RUNS (Program 11)

When applying the runs test, each number is compared with the previous one until a run occurs according to the definition given in Chapter 2. A count is kept of the number of occurrences of each run length, say r, where r can vary from one up to and including six. If any run is of length greater than six, one is added to the count for runs of length six. A slightly different χ^2 statistic is used and this statistic and the corresponding probabilities required are also given in Chapter 2. Once the χ^2 statistic is calculated the procedure is the same as that for the poker test.

Subroutine GAP (Program 10)

In applying the gap test, a gap is calculated according to the rules given in Chapter 2. When a gap of length R is found the COUNT(R) is increased by one. If any gap is of length greater than T, where T is the maximum length of a gap, then the counter for the gap of length T is increased by one. The χ^2 statistic is calculated and the remainder of the subroutine GAP is similar to that of the subroutine POKER. Again the probabilities and the statistic used are given in Chapter 2 under the gap test.

DIAGRAM (8)

FLOW DIAGRAM
for the
D^2 TEST

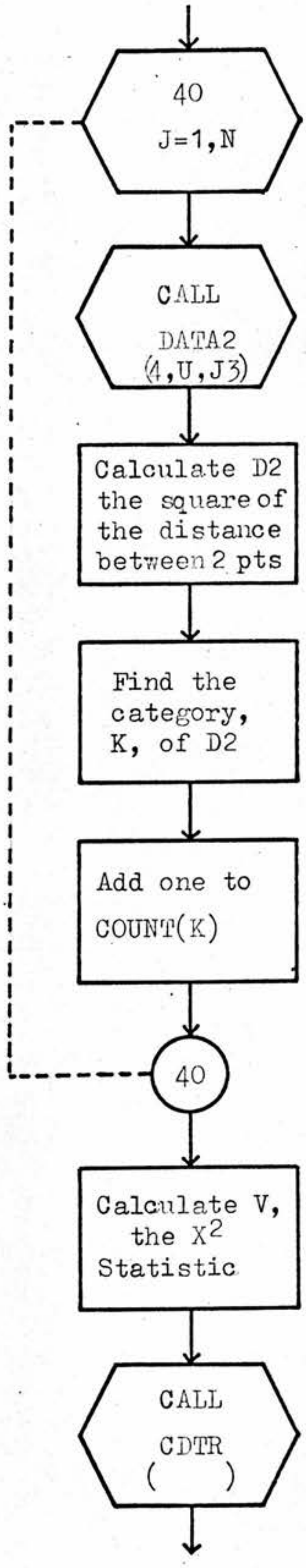


DIAGRAM (9)

FLOW DIAGRAM
for the
RUNS TEST

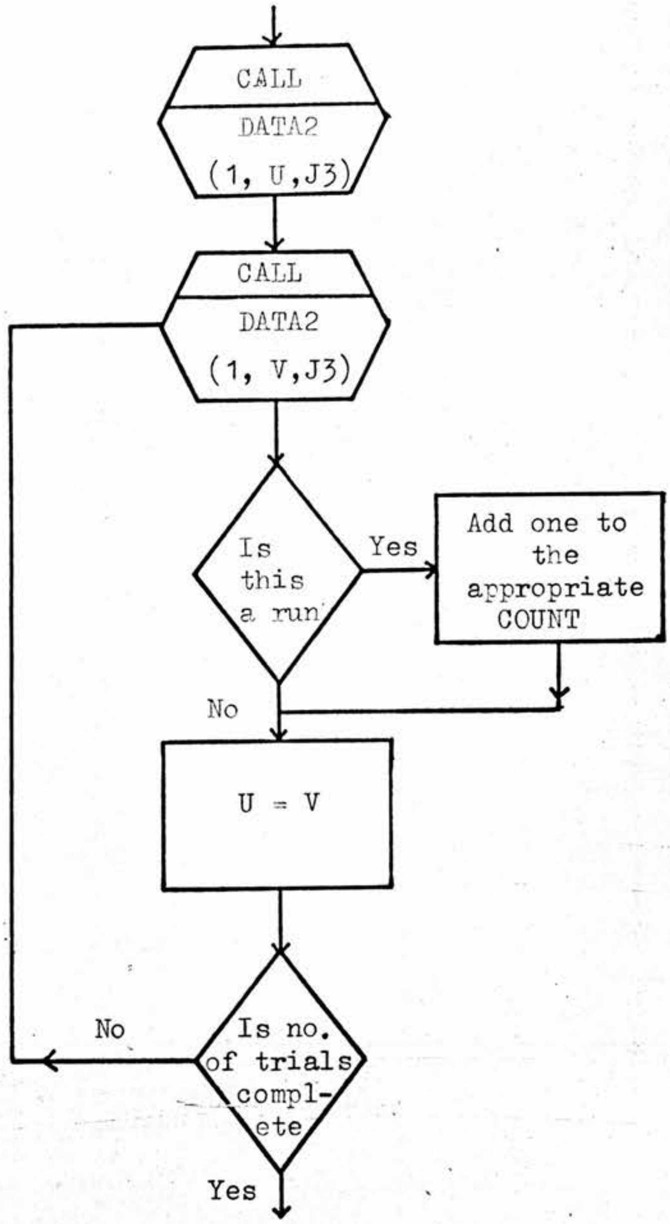
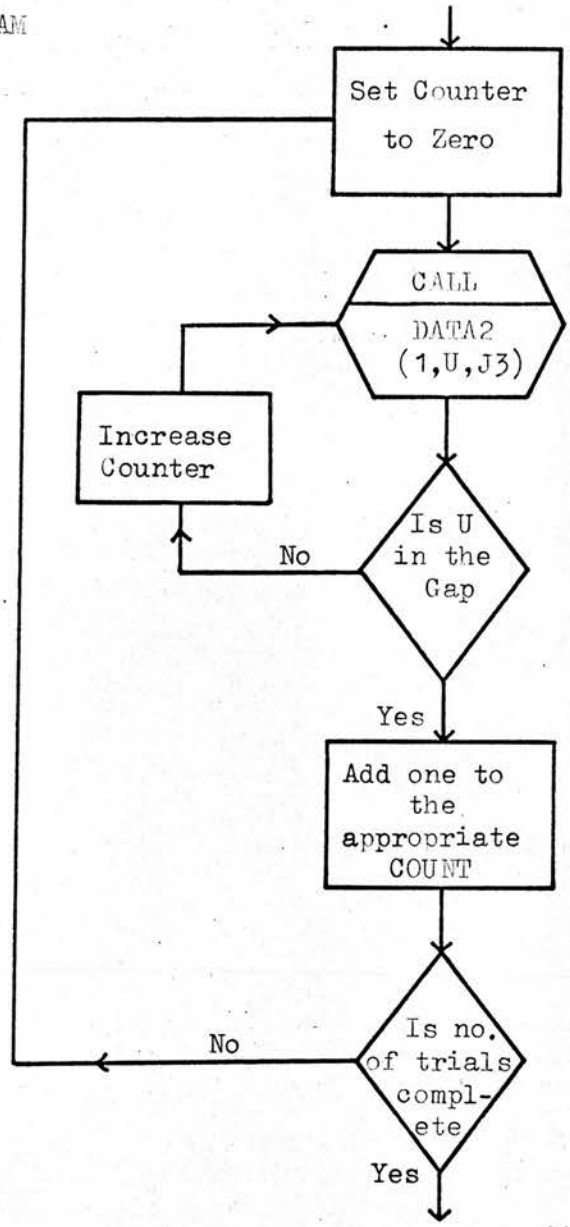


DIAGRAM (10)

FLOW DIAGRAM
for the
GAP TEST



Subroutine GAPRD (Program 13)

The subroutine GAPRD calculates the gap lengths for each digit according to the rules given in Chapter 2 for the gap test for random digits. An array is kept which contains the position at which each digit last occurred. An occurrence of a number must end a gap for that number. Thus, when a digit occurs the position is noted and the gap length from whence it last occurred can be calculated. An accumulating sum and a sum of squares are kept for the gap lengths of each digit. Every occurring gap length is added appropriately into the accumulating totals for that particular digit and the counters are adjusted accordingly. Thus, for each digit, it is relatively easy to perform significance tests on the variance and the mean of these gap lengths according to the rules given in Chapter 2.

Subroutine AP261 (Program 17)

This subroutine provides the distribution function for the maximum of U uniform random numbers. Thus, it calculates the probability, Y, of a variable V, with distribution function

$$F(V) = V^u$$

being less than X.

That is

$$Y = \text{prob}\{V < X\} = X^u$$

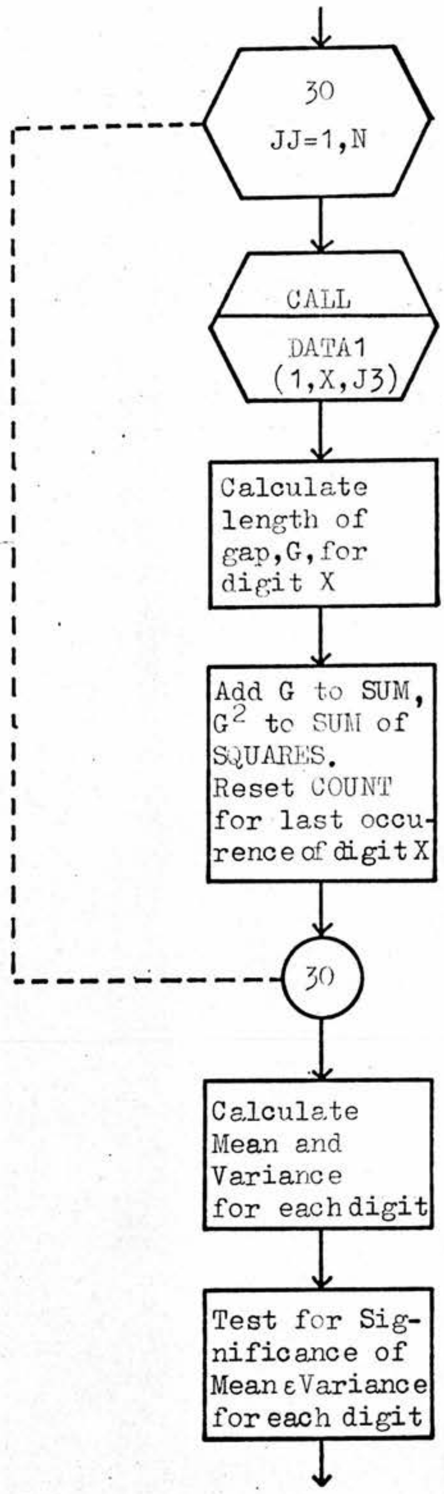
This theoretical distribution function is used for comparison in the maximum test and U is the total of numbers in each trial.

Subroutine AP262 (Program 18)

This subroutine provides the distribution function for the minimum of U uniform random numbers. Thus it calculates the probability, Y, of

DIAGRAM (11)

FLOW DIAGRAM
for the
Subroutine
GAPRD



a variable V, with distribution function

$$F(V) = 1 - (1-V)^u$$

being less than X.

That is

$$Y = \text{prob}\{V < X\} = 1 - (1-X)^u$$

This theoretical distribution function is used for comparison in the minimum test and u is the total of numbers in each trial. The remaining subroutines AP263, AP264, ...,AP267 are similar to the above two subroutines except they are used for comparison in the sum test. Thus, using the formulae given in Chapter 2, these subroutines provide the distribution functions that are given below.

Subroutine AP263 (Program 19)

Distribution function of the sum of 2 uniform random numbers.

$$F(x) = \begin{cases} \frac{X^2}{2} & 0<x<1 \\ 1 - \frac{(2-X)^2}{2} & 1<x<2 \end{cases}$$

Subroutine AP264 (Program 20)

Distribution function of the sum of 3 uniform random numbers.

$$F(x) = \begin{cases} \frac{X^3}{6} & 0<x<1 \\ \frac{1}{2}(\frac{X^3}{6} - (X-1)^3) & 1<x<2 \\ \frac{1}{2}(\frac{X^3}{6} - (X-1)^3 + (X-2)^3) & 2<x<3 \end{cases}$$

Subroutine AP265 (Program 21)

Distribution function of the sum of 4 uniform random numbers.

$$F(x) = \begin{cases} \frac{X^4}{24} & 0<x<1 \\ \frac{1}{6}(\frac{X^4}{4} - (X-1)^4) & 1<x<2 \\ \frac{1}{6}(\frac{X^4}{4} - (X-1)^4 + \frac{3}{2}(X-2)^4) & 2<x<3 \\ \frac{1}{6}(\frac{X^4}{4} - (X-1)^4 + \frac{3}{2}(X-2)^4 - (X-3)^4) & 3<x<4 \end{cases}$$

Subroutine AP266 (Program 22)

Distribution function of the sum of 5 uniform random numbers.

$$F(x) = \begin{cases} \frac{1}{24}(\frac{x^5}{5}) & 0<x<1 \\ \frac{1}{24}(\frac{x^5}{5} - (X-1)^5) & 1<x<2 \\ \frac{1}{24}(\frac{x^5}{5} - (X-1)^5 + 2(X-2)^5) & 2<x<3 \\ \frac{1}{24}(\frac{x^5}{5} - (X-1)^5 + 2(X-2)^5 - 2(X-3)^5) & 3<x<4 \\ \frac{1}{24}(\frac{x^5}{5} - (X-1)^5 + 2(X-2)^5 - 2(X-3)^5 + (X-4)^5) & 4<x<5 \end{cases}$$

Subroutine AP267 (Program 23)

Distribution function of the sum of 6 uniform random numbers.

$$F(x) = \begin{cases} \frac{1}{120}(\frac{x^6}{6}) & 0<x<1 \\ \frac{1}{120}(\frac{X^6}{6} - (X-1)^6) & 1<x<2 \\ \frac{1}{120}(\frac{X^6}{6} - (X-1)^6 + \frac{5}{2}(X-2)^6) & 2<x<3 \\ \frac{1}{120}(\frac{X^6}{6} - (X-1)^6 + \frac{5}{2}(X-2)^6 - \frac{10}{3}(X-3)^6) & 3<x<4 \\ \frac{1}{120}(\frac{X^6}{6} - (X-1)^6 + \frac{5}{2}(X-2)^6 - \frac{10}{3}(X-3)^6 + \frac{5}{2}(X-4)^6) & 4<x<5 \\ \frac{1}{120}(\frac{X^6}{6} - (X-1)^6 + \frac{5}{2}(X-2)^6 - \frac{10}{3}(X-3)^6 + \frac{5}{2}(X-4)^6 - (X-5)^6) & 5<x<6 \end{cases}$$

Subroutine GETFMT (Program 29)

Subroutine GETFMT compares an input and a blank array. Should they agree, then the input array is also blank, and another array is returned with the standard format. The standard format used depends on the value of the parameter SWITCH, Program 29, and the formats available are given in Chapter 5 under the section called DATA. Otherwise a further array is returned with the input format which is assumed to be correct. The dimension of the input array is eight. Therefore, as the format is read under A4 format, the maximum number of characters in the format is thirty-two. This includes any brackets and all format symbols. Thus, it is undesirable to have blanks in the format statement since they are counted as characters.

Subroutines DATA1 and DATA2 (Programs 27 and 28)

Depending on the data control parameters, subroutine DATA1 and DATA2 obtain numbers to be tested from a file on unit 3, from cards or from a user supplied generator. In the first two cases a check is made to see if this is the first data call from this test. Should this be true, the numbers are read from the file or cards into any array, one record or card at a time. Each record must contain 82 unformatted numbers. On the other hand, the cards have no fixed amount of numbers so long as they do not vary in content within a test. Counters are kept as to how many numbers in a record or card have been used and as to whether a new record or card must be read in. Evey kth number can be used by increasing the appropriate counter by K, AA(4), instead of one. Finally if the numbers are to be produced by a generator, a READ subroutine is called. This subroutine will be supplied by the user and the method and rules to follow are described at the end of this chapter and in Chapter 5 under Read step.

Subroutine APSN (Program 30)

Stirling's numbers of the second kind, $\left\{{K \atop L}\right\}$ are

K \ L	0	1	2	3	4	5	6
0	1	0	0	0	0	0	0........
1	0	1	0	0	0	0	0
2	0	1	1	0	0	0	0
3	0	1	3	1	0	0	0
4	0	1	7	6	1	0	0
5	0	1	15	25	10	1	0........
	⋮						

(Knuth Vol. 1)

Given that $\left\{{0 \atop 0}\right\} = 1$

then for all K > 0 Stirling's numbers can be calculated using the relationship

$$\left\{{K \atop L}\right\} = \left\{{K-1 \atop L-1}\right\} + L*\left\{{K-1 \atop L}\right\}$$

Each row in the above table can be calculated from the previous one. Hence, the numbers in row(i) are stored in array IS. Then each number in row(i+1) is calculated using the above recurrence relationship and put into an array JTEMP. When all the numbers of row(i+1) have been calculated, they are stored in the array IS and overwrite the previous values given by row(i). Thus space is saved by keeping only two rows of the above table in store at any one time.

The dimension of each of the arrays IS and JTEMP is twenty and therefore, the maximum value of K is also twenty. Hence, if the user wishes to calculate Stirling's numbers for K > 20 the dimensions of IS and JTEMP must be increased.

Subroutine APTR (Program 31)

The subroutine, APTR, puts into words the meanings deduced from the probabilities which are calculated by the statistical tests using the rules given in Chapter 2 for the KS and χ^2 tests.

A listing of all these subroutines is given at the end of this thesis in Appendix 1. They are listed in the order of their program numbers.

User Step

The user is able to supply his own theoretical distribution function for comparison purposes in the subroutine KOLMO. This is carried out by replacing the dummy subroutine, USER, in APLIB with the user's own subroutine in addition to the automatic deletion and linkage editing of the phases. Thus, every time this option is used the subroutine, USER, is changed and therefore one particular subroutine is not a permanent feature in the package.

The above option is implemented by 'copying' the relevant control statements into a file on disc, SA45V1. Thus the library AP was created and into the member A1, the linkage editing control statements were 'copied', using a system utilities program. These control statements are shown in diagram 12. A further member of the library, AP, is USER which contains the job control statements for deleting and linkage editing. These job control statements are listed in diagram 13.

To invoke the user step the following cards are inserted by the user after the JOB card.

```
//SYSRDR     ACCESS      AP(USER), 191 = 'SA45V1'
/*
  source statements for function USER
/*
:
```

DIAGRAM 12

```
PHASE  APTEXX,ROOT
INCLUDE  APMAIN,R,2
INCLUDE  NDTR,R,2
INCLUDE  APTR,R,2
INCLUDE  DATA1,R,2
INCLUDE  DATA2,R,2
INCLUDE READ1,R,2
INCLUDE READ2,R,2
INCLUDE  GETFMT,R,2
INCLUDE  IBCOM#,R
INCLUDE  FIOCS#,R
INCLUDE  USEROPT,R
INCLUDE  UNITAB#,R
INCLUDE  LOAD,R
INCLUDE  EXP,R
INCLUDE  FRXPI#,R
INCLUDE  FRXPR#,R
INCLUDE  SQRT,R
INCLUDE  ARCOS,R
INCLUDE  DLOG,R
INCLUDE  DEXP,R
INCLUDE  DSQRT,R
INCLUDE  ALOG,R
INCLUDE  KLOCK,R
INCLUDE  ATAN,R
INCLUDE  AMAXI,R
PHASE  APCXX,*
INCLUDE FREQ,R,2
INCLUDE POKER,R,2
INCLUDE SERIAL,R,2
INCLUDE APSN,R,2
INCLUDE GAP,R,2
INCLUDE DSQUR,R,2
INCLUDE RUNS,R,2
INCLUDE CDTR,R,2
INCLUDE DLGAM,R,2
PHASE  APKMXX,APCXX
INCLUDE  SUM,R,2
INCLUDE  MAX,R,2
INCLUDE  MIN,R,2
INCLUDE  AP261,R,2
INCLUDE  AP262,R,2
INCLUDE  AP263,R,2
INLCUDE  AP264,R,2
INCLUDE  AP265,R,2
INCLUDE  AP266,R,2
INCLUDE  AP267,R,2
INCLUDE  USER,R,2
INCLUDE  KOLMO,R,2
INCLUDE  SMIRN,R,2
PHASE  APOTXX,APKMXX
INCLUDE  GAPRD,R,2
/*
```

DIAGRAM 13

```
//SYSLST  ACCESS  IGN
//  ACCESS  APLIB,191='SA45V1'
//  DELETE  APLIB(USER)
//  CONDENSE  APLIB
//SYS000  ACCESS  APLIB(USER),191='SA45V1',NEW
//USER  EXEC  FORTRAN
//  ACCESS  SDSABS
//  DELETE  SDSABS(APTEXX)
//  DELETE  SDSABS(APTEXY)
//  DELETE  SDSABS(APCXX)
//  DELETE  SDSABS(APCXY)
//  DELETE  SDSABS(APKMXX)
//  DELETE  SDSABS(APKMXY)
//  DELETE  SDSABS(APOTXX)
//  DELETE  SDSABS(APOTXY) ______________________ (a)
//  CONDENSE  SDSABS
//SYS002  ACCESS  APLIB,191='SA45V1'
//SYSIPT  ACCESS  AP(A1),191='SA45V1'
//APTEXX  EXEC  RLNKEDT(KEEP,SYS002,NOAUTO)
//  RENAME  SDSABS(APTEXX,APTEXY)
//  RENAME  SDSABS(APCXX,APCXY)
//  RENAME  SDSABS(APKMXX,APKMXY)
//  RENAME  SDSABS(APOTXX,APOTXY) ______________ (b)
//  DELETE  SDSABS(APTEST)
//  DELETE  SDSABS(APCHI)
//  DELETE  SDSABS(APKMSM)
//  DELETE  SDSABS(APOTHR) ______________________ (c)
//  RENAME  SDSABS(APTEXY,APTEST)
//  RENAME  SDSABS(APCXY,APCHI)
//  RENAME  SDSABS(APKMXY,APKMSM)
//  RENAME  SDSABS(APOTXY,APOTHR) ______________ (d)
//  RESET  SYSLST
//  RESET  SYS000
//  RESET  SYSIPT
//  RESET  SYSRDR
```

These cards cause SYSRDR to read and execute the job control statements in the member USER of AP, as shown in diagram 13. Since there is no need for the user to see these jobs control statements, they are not printed when they are executed due to the inclusion of the statement

```
//SYSLST      ACCESS      IGN
```

The statement

```
//USER      EXEC      FORTRAN
```

causes the new subroutine to be compiled and is then stored in APLIB under the name USER.

It is important that the phases APTEST, APCHI, APKMSM and APOTHR of this package are not deleted. If the compilation of the new subroutine fails, all these phases would be removed from SDSABS. Therefore, the deleting and linkage editing is carried out by first, using dummy names (APTEXX, APKMXX, APCHXX, APOTXX), then renaming under new dummy names (APTEXY, etcetera), followed by deleting and renaming under the real phase names (APTEST, etcetera). Thus, if the compilation of the new function fails the linkage editor proceeds but also fails and therefore the first renaming step causes the job to abort. This is because the first renaming step will be out of sequence and an attempt to rename the phase, APTEXX, will fail as it will not have been created by the linkage editor. As a result, the original phases are not deleted.

The statement

```
//SYSIPT      ACCESS      AP(A1), 191 = 'SA45V1'
```

in diagram 13 causes the input to come from the file A1 which contains the linkage editing control statements. These control statements followed by

```
//APTEXX      EXEC      RLNEDT(            )
```

form the phases called APTEXX, APKMXX, APCHXX and APOTXX in SDSABS.

If any new subroutine has been added permanently to the package and the user option is to be used, the extra linkage editing control statements for this subroutine must be added into A1. These control statements will vary depending on the number of additional phases or, alternatively, to which phase the new subroutine has been added. For example, suppose a new subroutine, say TRY, is permanently added to phase APOTHR, then A1 must be enlarged with the statement

```
INCLUDE     TRY, R, 2
```

which is inserted after the statement

```
INCLUDE     GAPRD, R, 2
```

Similarly, if a new phase is permanently added to the package, then A1 must be enlarged with the following statements,

```
PHASE       'dummy name of new phase',      APOTXX
INCLUDE     C, R, 2
INCLUDE     D, R, 2
```

which are inserted after the last statement of phase APOTXX. These set the origin of this particular phase equal to the origin of the phase, APOTXX. However, the origin of a new phase need not be this and can be changed by altering the parameters in the PHASE statement according to the rules given in the previous section concerning the phasing system (diagram 1).

Furthermore, if a new phase is added, extra statements have to be inserted in the member USER of AP; they are

```
a)  // DELETE     'dummy name 1 of new phase'
    // DELETE     'dummy name 2 of new phase'
b)  // RENAME     SDSABS('dummy name 1 of new phase','dummy name 2
                              of new phase')
         :
c)  // DELETE     SDSABS('new phase name')
         :
```

d) // RENAME SDSABS('dummy name 2 of new phase', 'new phase
.
. name')

which are included in the appropriate places as shown in diagram 13.

Read Step

A user is able to insert his own subroutine into the package to obtain data. This subroutine will not be permanently in the package and will frequently be written in the form of a generator. Therefore, this option has been devised. A new member of AP was created, called READ1. This member, shown in diagram 14, is very similar to the member USER of AP. The main difference is that the statement

```
// USER      EXEC      FORTRAN
```

is replaced by

```
// EXEC      ASSEMBLE(LINK,NODECK)
```

which will compile an Assembler subroutine. The read step is invoked, set up and used in virtually the same way as the user step. However, the READ1 subroutine must be written in Assembler language. The rules for adding new subroutines and phases are exactly the same as those of the user step.

There are two read options, READ1 and READ2, the difference being that READ1 supplies integer data and READ2 supplies real data. Thus, there is also the member, READ2, in the library AP. This is the same as READ1 with the exception that whenever READ1 occurs in diagram 14, it is replaced by READ2.

Data Set Reference Numbers

The data set reference numbers used in this package are

IPRINT which is set to 6 for writing under format control.

IRED which is set to 3 for reading and writing from an unformatted file on unit 3.

DIAGRAM 14

```
//SYSLST  ACCESS  IGN
//  ACCESS  APLIB,191='SA45V1'
//  DELETE  APLIB(READ1)
//  CONDENSE  APLIB
//SYS000  ACCESS  APLIB(READ1),191='SA45V1',NEW
//  EXEC  ASSEMBLE(LINK,NODECK)
//  ACCESS  SDSABS
//  DELETE  SDSABS(APTEXX)
//  DELETE  SDSABS(APTEXY)
//  DELETE  SDSABS(APCXX)
//  DELETE  SDSABS(APCXY)
//  DELETE  SDSABS(APKMXX)
//  DELETE  SDSABS(APKMXY)
//  DELETE  SDSABS(APOTXX)
//  DELETE  SDSABS(APOTXY)
//  CONDENSE SDSABS
//SYS002  ACCESS  APLIB,191='SA45V1'
//SYSIPT  ACCESS  AP(AI),191='SA45V1'
//APTEXX  EXEC  RLNKEDT(KEEP,SYS002,NOAUTO)
//  RENAME  SDSABS(APTEXX,APTEXY)
//  RENAME  SDSABS(APCXX,APCXY)
//  RENAME  SDSABS(APKMXX,APKMXY)
//  RENAME  SDSABS(APOTXX,APOTXY)
//  DELETE  SDSABS(APTEST)
//  DELETE  SDSABS(APCHI)
//  DELETE  SDSABS(APKMSM)
//  DELETE  SDSABS(APOTHR)
//  RENAME  SDSABS(APTEXY,APTEST)
//  RENAME  SDSABS(APCXY,APCHI)
//  RENAME  SDSABS(APKMXY,APKMSM)
//  RENAME  SDSABS(APOTXY,APOTHR)
//  RESET  SYSLST
//  RESET  SYS000
//  RESER  SYSIPT
//  RESET  SYSRDR
```

JRED which is set to 5 for reading cards under format control. These parameters are all placed in Common and are assigned the above values at the beginning of APMAIN, Program 1. Thus, if the user wishes to alter any of these parameters the appropriate statements in APMAIN must be altered and the phases deleted and linkage edited.

Writing of Extra Subroutines

All the data control parameters, FI, AA, NUM and FMT2, together with a starting value for a generated sequence, if one is requred, ISTART, and the data set reference numbers, IPRINT, IRED and JRED, are passed into Common area. (Most of these parameters are described more fully in Chapter 5). Therefore, if the user wishes to incorporatre these parameters in one of his subroutines he must include the following statement.

```
COMMON      FI(5), AA(9), NUM, FMT2(8), ISTART, IPRINT, IRED, JRED
```

The variable names, given in the above statement, must not be duplicated in the subroutines but they can be assigned different names by using an equivalence statement. If the subroutine is written in assembler language the COM statement must be inserted instead of the COMMON statement. This statement and its applications are described in IBM C28-6811-1.

Also, certain parameters must be passed to and from the subroutines provided by the user, USER, READ1 and READ2, as these parameters are not in Common.

The USER subroutine has three parameters:- USER(X, Y, IER)

where

X — is the input scalar from the subroutine KOLMO

Y — is the output scalar to the subroutine KOLMO. This is the probability that Z is less than X if Z has the distribution function given by this subroutine.

IER --- is the error code which is non-zero if there is an error in the input or output parameters (as shown in program 24). Otherwise there is no error and the error code is set equal to zero. When the USER subroutine is used, the error code must be set equal to zero on entry to that subroutine otherwise an error will be recorded.

The READ1 subroutine has three parameters:- READ1 (M,K, LL) where

M --- is the number of integers required in each data call from a statistical test.

K --- is the output vector of M dimensions which returns the requested data.

LL --- is an index which is either zero, if this is the first data call from the test, or one in all other cases.

Subroutine READ2 has the same parameters as the subroutine READ1, except the output vector, V, is of real numbers:- READ2 (M,V,LL)

Most statistical tests require one number per data call. However the D^2 test requires four numbers per data call and the poker test requires M, where M can be between one and twenty. Therefore, when a READ subroutine is written, it must provide the facility to obtain a variable amount of numbers for each data call.

When writing a new statistical subroutine, the user must adhere to the rules on the input parameters that are given in Chapter 5.

Formats

The number of data control parameters is fixed due to the dimensions of the variables FI and AA which are five and nine respectively. Therefore, if the number of parameters has to be increased, then the dimensions of FI and AA would also have to be increased accordingly in all Common statements. However these parameters can be read with a different format than that provided by the system, taking into account that FI is integer and AA is real. The standard format is

(4I4, I10, 5F5.1, 2F9.4, 2F5.1)

Thus, using the subroutine GETFMT, if the first half of the first card of input parameters is blank, then the standard format is used. However, if the first half of this card is not blank, that is, it contains a format statement, this format is read in and used to read the succeeding card which contains the data control parameters. The format statement must include brackets and format symbols as shown above.

The user may wish to read the numbers to be tested from cards. These should be placed in the deck immediately after the card containing the data control parameters for the test. Again, there is a format provided by the system for reading these cards:

(5E14.7)

On the other hand the user may wish to provide his own format to read the cards. In this case, the last half of the first card of input parameters is considered and the same proceedure is adopted as above. The only difference is that the format required is read into Common area, FMT2, and used later to read the data cards. If the cards are to be read with a user's format, the parameter, NUM, must be assigned a value. This parameter is the number of numbers on each data card. The value of NUM is placed between the two formats, described above, on the first input parmaeter card and this is described in more detail in the next Chapter. (Cress et al 1968)

By varying the parameter AA(8), the numbers to be tested may be obtained from various sources. The sources, associated with the values of AA(8), are described in the next Chapter.

The reader may wonder why the user step has not been extended to include the adding of new tests and subprograms which would probably include several subroutines. One of the reasons is that there would always be a limitation to the number of subroutines that could be added. This is because this step depends on the number of dummy subroutines available in the package. Since, for the subroutines to be incorporated automatically, the relevant linkage editing control statements must already be in a data set, of library AP. Another reason is that the subroutines would not be in the package permanently. Hence, with more than one user, it would be difficult to ensure the contents of a particular dummy subroutine. Therefore, the procedure described previously for adding subroutines permanently to the package must be followed.

CHAPTER V. USER'S MANUAL

The package is run with numbers obtained from a file on disc using the following cards:

```
// 'jobname'  JOB ,'users name'  'time of job in minutes'
// SYS003  ACCESS  APOLLO, 192 = 'JHPSV3'
// bEXEC  APTEST  (b implies blank column)
                 :
            sets of data cards
                 :
/*
/&
```

where the second statement accesses a data set called APOLLO on unit 3, on disc JHPSV3 with disc drive 192. Unit 3 must always be used for a file in this package (see data set reference numbers in Chapter 4). The names for the data set, the disc and the disc drive are those which were employed for this work. These should be replaced by the parameters of the user's own file which contains the numbers to be tested. This ACCESS statement must be altered if the file is on a tape. If the numbers to be tested are read from cards, there is no ACCESS card. These cards are included for each test and are placed directly after the input parameter cards for that test. Also, when cards are used, some data control parameters have to be changed. However, if the numbers to be tested are produced by a generator, the READ step is used and instead of the ACCESS card, a set of Fortran and control cards to invoke this option is included. (see READ step later) Also, certain parameters must be changed. If an integer starting value is required for the generator, it must be read in on a separate card. This card is placed immediately after the last input parameter card of a test.

The third statement

```
// EXEC  APTEST
```

starts the execution of the package since APTEST is the name of the root phase. (see Chapter 4)

For further information on control cards see IBM C28 - 6812 and
IBM C28 - 6813.

If the numbers to be tested are to be stored in a file on disc, the following statements should be used.

```
// 'jobname'  JOB ,'user's name'  'time of job'
// SYS003  ALLOC  APOLLO, 192 = 'JHPSV3', 1000, FMT
//         LABEL  360, 'time this data set expires'
//         EXEC  FORTRAN
                 :
      Fortran program for producing the numbers
                 :
/*
//         EXEC  RLOADER
/*
             :
      Data cards
             :
/*
/&
```

The second and third cards cause the data set, APOLLO to be assigned 1000 blocks of storage with each block containing 360 bytes. Since the numbers are written unformatted into the data set, they each take up 4 bytes, and thus, due to buffering, the quantity of numbers in each block is fixed at 82. Hence, both of the DATA subroutines have been written for 82 numbers per block.

<u>The Numbers to be Tested</u>

In general, the numbers to be tested are stored in a sequential unformatted file on unit 3. This file may be on a disc or a tape. If it is on a tape, some of the aforementioned job control statements concerning the allocation and accession of the data in a file on disc, must be altered slightly. (IBM C28 - 6812, IBM C28 - 6813) The numbers must be stored with 82 per record. These numbers may be multiplied by the parameter AA(1) before use but this depends on the range of numbers required in a test. (see later for definition of AA(1))

The numbers to be tested may be punched on cards and read under format control. The total of numbers on a card is not fixed but must be constant within a statistical test. Again, the numbers may be multiplied by AA(1), before use, but this depends on the range of numbers required in a test.

Finally, the user may provide his own way of obtaining data by writing a subroutine. This subject is described under the Read step in Chapters 4 and 5.

Format Codes

For all the three format codes I, F and E, any blanks within a field are interpreted as zeros. A field is the number of columns on a card, that is read positions, alloted to one variable. For the I format code which has general form Iw, where w is the length of the field, the integer variable should be right justified in that field. Thus if the digit 2 is read using the I4 format, its value would be assigned as follows:-

b b b 2 → 2
b b 2 b → 20
b 2 b b → 200
2 b b b → 2000

where b denotes a blank column

Therefore, the format 4I4 specifies that the first variable, say FI(1), is read from the first four columns, the second variable, say FI(2), from the next four columns, the third variable, say FI(3), from the third four columns and the final variable, say FI(4), from the columns numbered 13 - 16. Thus, to assign the values 1, 10, 2 and 5 to the above variables respectively, the data card must be punched in the following way.

b	b	b	1	b	b	1	0	b	b	b	2	b	b	b	5
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

The F format code, which is for real numbers, is of general form,

F w.d

where w is the length of the field and d is the number of places after the decimal point, both of these parameters are integer variables. Thus F5.1 indicates a field of length five with one place after the decimal point. It should be noted that the decimal point can be included anywhere in the field and it then overrides the position indicated in the format statement. Therefore by punching the following on the data card:-

(1) ...... b 1 0 . 0 .... (2) ...... 1 0 . 0 b

27 28 29 30 31 27 28 29 30 31

the variable, say AA(1), is assigned the value 10.0. Case (1) is equivalent to the format F5.1 and case (2) is equivalent to F5.2 but would still be accepted under the F5.1 format. There are similar rules for the E format code which has the general form

E w.d

(IBM C28 - 6515 - 7)

Thus, care must be taken in punching data cards to ensure that the values are in the correct columns.

DATA

A set of data cards must be included for each test. These cards provide the input parameters in addition to any further information as was described above.

The first card in a set contains the parameters IN, NUM and JN in that order. They are all integer variables with IN and JN arrays of dimension eight.

IN —— This variable indicates which format is to be used to read the next card. If IN contains a format this format is used to read the next card otherwise the standard format is used.

NUM —— This variable is the total of numbers on a card if the numbers to be tested are read from cards. It must have a constant value within a test but may be altered for different tests.

JN —— This variable indicates which format is to be used to read the cards containing the numbers to be tested. A format will always be assigned for this purpose even though the numbers may not be read from cards. If JN contains a format, this format is used, otherwise the standard format is taken.

The values of IN and JN are read with A4 format whereas the variable, NUM, is read with I4 format. Thus IN and JN each take up 32 columns of a card and they are separated by the 4 columns containing the value of NUM. This implies that 32 characters may be read into IN and JN but not more. If a format is to be read into either IN or JN, it must contain all the format symbols including brackets and commas but not the word FORMAT. The first bracket of IN must be in column one of the card and the first bracket of JN must be in column 37 of this card. If these formats, on being used in a read or write statement, are made incorrect by the omission of a necessary bracket or symbol, an error will occur. Also, if these formats contain more than 32 characters an error will occur as the closing bracket will be missing. Thus, care must be taken to keep within these restrictions when writing such formats.

On the other hand, if IN is blank, the systems format is used to read the succeeding card. Similarly, if JN is blank, the same procedure is adopted. The formats provided by the system for such cases are listed below.

IN — (4I4, I10, 5F5.1, 2F9.4, 2F5.1)

NUM — 5

JN — (5E14.7)

However, if either IN or JN contains a format statement, this is read in and used to read the appropriate cards.

The second card in a set contains the parameters FI and AA, where FI is an integer array of dimension five and AA is a real array of dimension nine. When a new format is written to read these variables, the user must remember that they are defined as integer and real respectively. The meanings given to these parameters are listed below.

FI(1) The test requested.

Each statistical test is coded so that the main program can call the correct test.

The codes are as follows:-

0	End of run	6	Gap Test for random digits
1	Frequency Test	7	Runs Test up and down
2	Serial Test	8	Maximum Test
3	Poker Test	9	Minimum Test
4	Gap Test	10	Sum Test
5	D^2 Test		

This parameter must be assigned a value for all tests, since by default FI(1)=0 and the run will be terminated.

FI(2) This is the number of times a test is repeated on a run. This parameter must also be assigned a value for all tests since by default, FI(2)=0 and no test will be made.

FI(3) This is the sum of numbers requested in each data call by a test. This parameter is only of importance in the Poker test and must be assigned a value since FI(3) is also the total of numbers to be considered at each trial. Furthermore, FI(3) must be less than or equal to twenty.

FI(4) This parameter is dependent upon the test being applied.

Sum, Maximum and Minimum Tests

For these statistical tests, which use the KS test, this parameter is a code which shows the distribution function to be used for comparison purposes. In this case FI(4) must be assigned a value and the codes are listed below.

1 Normal distribution function

2 Exponential distribution function

3 Cauchy distribution function

4 Uniform distribution function

5
- Maximum of T numbers distribution function
- Minimum of T numbers distribution function
- Sum of 2 uniform random variables distribution function
- Sum of 3 uniform random variables distribution function
- Sum of 4 uniform random variables distribution function
- Sum of 5 uniform random variables distribution function
- Sum of 6 uniform random variables distribution function
- User supplied distribution function

If one of the distribution functions with code 5 is used, a further parameter, AA(5) must be specified. Also, if a subroutine containing a distribution function is permanently added to this package for using in these statistical tests, it should have code 5.

Gap Test

For the gap test this parameter is the maximum length of a gap. FI(4) must be assigned a value which is chosen in conjunction with the parameters FI(5) and the probabilities given in Chapter 2. This gives an expected number of occurrences of each gap length which must be greater than or equal to five.

Runs Test

For this test FI(4) must be assigned a value since this parameter shows whether the runs are to be made up or down (as described in Chapter 2).

$$FI(4) = \begin{cases} 0 & \text{Runs up} \\ 1 & \text{Runs down} \end{cases}$$

Thus, by default, if FI(4) is blank or zero, this test will be performed for runs up.

FI(5) This is the number of trials for each test which must be included for all tests. Otherwise by default FI(5)=0 and no proper test will occur. In the gap test this is the number of gaps to be found.

AA(1) This is the base of the number system for the numbers that are to be used in the statistical test. AA(1) is used in calculating the probabilities for the χ^2 test as well as for bringing the numbers to be tested into the correct range. If the numbers are not in the correct range and if AA(8)= 0.0 or 2.0 (described later) the numbers are multiplied by AA(1) to bring them into the correct range. If the numbers are in the correct range and if AA(8)=1.0 or 3.0 the numbers are left unchanged. In both cases AA(1) must be specified to ensure a base is present if one is required.

If the numbers are not in the correct range and their base does not bring them into the required range, then the numbers should be adjusted. It should be remembered that AA(1) is used in the χ^2 test according to the rules given in Chapter 2. By default if AA(1) is blank or zero it is reassigned the value 1.0, that is AA(1)=1.0.

AA(2)=1.0 implies that the file of numbers to be tested is rewound. The user cannot assume that the file will be at its load point (beginning) whenever a new run is applied.

≠1.0 implies that the file of numbers to be tested is not rewound. By default AA(2)=0.0 and the file is not rewound.

AA(3) This gives the position of the record from which the numbers to be tested are read. If it is blank or zero, then by default the next record is used.

AA(4) This informs the data subroutines which numbers are to be used. That is, whether every number or every Kth number from each record is tested. AA(K) must have a value of less than or equal to 82 and if it is blank or zero, then by default every number is tested.

AA(5) This parameter is dependent upon the test being applied.

Sum, Maximum and Minimum Tests

When these statistical tests are used and FI(4)=5, this parameter is also a code. This code enables the user to specify which distribution function is to be used for comparison purposes. A list of the codes is given below.

1.0 Distribution function for the Maximum of T numbers at a time

2.0 Distribution function for the Minimum of T numbers at a time

3.0 Distribution function for the Sum of 2 uniform random numbers

4.0 Distribution function for the Sum of 3 uniform random numbers

5.0 Distribution function for the Sum of 4 uniform random numbers

6.0 Distribution function for the Sum of 5 uniform random numbers

7.0 Distribution function for the Sum of 6 uniform random numbers

8.0 User supplied

If any extra distribution functions are added permanently to the package, the value for this parameter should be greater than 8.0 and AA(6) and AA(7) may be defined as required.

Gap Test

When the gap test for random numbers is applied, this parameter is the lowest end of the gap as defined in Chapter 2.

Gap Test for random digits

When this test is applied AA(5) is the significance level for this test which, by default, has a value of 0.05.

AA(6) This parameter is dependent on the test being applied.

Maximum, Minimum and Sum Tests

When these statistical tests are applied, AA(6) is an extra parameter. Its meaning depends on the subroutine which is to be used for comparison purposes.

If

FI(4) = 1 AA(6) is the mean of the normal distribution to be considered.

2 AA(6) is the mean of the exponential distribution to be considered.

FI(4) = 3 AA(6) is the median of the Cauchy distribution to be used.

FI(4) = 4 AA(6) is the left end point of the uniform distribution to be used.

FI(4) = 5 and AA(5)=1.0 AA(6) is the total of numbers (T) considered in each trial for the maximum test.

AA(5)=2.0 AA(6) is the total of numbers (T) considered in each trial for the minimum test.

AA(5)=8.0 AA(6) is user specified.

When AA(5)=3.0, 4.0, ... 7.0 then AA(6) need not be defined as the subroutines requested by these codes require no extra parameters.

Gap Test

When the gap test for random numbers is applied, this parameter is the highest end of the gap.

Gap Test for random digits

When this test is applied, AA(6) is the value of the variable with a normal (0,1) distribution for the above level of significance, AA(5). By default this is 1.96 which is for application to a two-sided test as described in Chapter 2.

AA(7) This parameter is dependent on the test being applied.

Maximum, Minimum and Sum Tests

When these tests are applied, AA(7) is an extra parameter which depends on the distribution function which is to be used for comparison purposes.

For

FI(4) = 1 then AA(7) is the standard deviation of the normal distribution to be considered and should be positive.

FI(4) = 2 then AA(7) is the standard deviation of the exponential distribution to be considered and should be positive.

FI(4) = 3 then AA(6) minus AA(7) specifies the first quartile of the Cauchy distribution to be considered, again it should be positive.

FI(4) = 4 then AA(7) is the right end point of the uniform distribution and should be greater than AA(6).

FI(4) = 5 and AA(5)=8.0 then AA(7) is user specified otherwise when AA(5) < 8.0 then AA(7) is not defined since these distributions require no extra parameters.

Gap Test for random digits

When this test is applied, AA(7) is the value of a variable with a normal (0,1) distribution for the above level of significance, AA(5). By default this is 1.625 which is for application to a one sided test as described in Chapter 2.

AA(8) This parameter shows how the numbers to be tested are obtained.

0.0 The numbers are read from an unformated sequential file on unit 3 and there are 82 numbers per record. Each number is multiplied by AA(1) before use.

1.0 The numbers are read from an unformatted sequential file on unit 3 and there are again 82 numbers per record. The numbers are used without alteration.

2.0 The numbers are read from cards under format control and NUM is the total of numbers per card. Each number is multiplied by AA(1) before use.

3.0 The numbers are read from cards under format control and NUM is the total of numbers per card. The numbers are used without alteration.

4.0 Optional.

The user may decide how the numbers are obtained by supplying his own subroutines for READ1 and READ2.

By default, when AA(8) is blank, the data is assumed to come from a sequential unformatted file.

AA(9) This parameter shows whether a starting value, for a generator, is required to be read in.

AA(9) $\begin{cases} = 0.0 & \text{no number is read in} \\ \neq 0.0 & \text{a number is read in} \end{cases}$

Thus, if AA(9) is blank or zero, by default no number is read in.

If AA(8) >= 4.0 then AA(9) can be ≠ 0.0 but

If AA(8) < 4.0 then AA(9) must be equal to 0.0.

If a starting value has to be read in, then it must be on the third card in a set and is read in under the I10 format. If the numbers to be tested are read from cards, then these cards should be included immediately after the second data card. To terminate a run, one blank card followed by a card with FI(1) equal to zero should be inserted at the end of all the data cards.

If a user writes a new test and includes it permanently in the package, according to the rules given in Chapter 4, then only four of the above parameters must have the same meanings. These are FI(1), AA(2), AA(3) and AA(9) as they are all used in the main program, APMAIN. However, if the DATA subroutines are called by the user's test, then the parameters AA(1), AA(4) and AA(8) must also have the same meanings as given above. Furthermore, when the KS test is used in a subroutine the parameters AA(5), AA(6) and AA(7) must have the meanings defined above.

If any parameters are left blank, then these parameters will be taken as zero. In some cases, as explained previously, there is a default option whenever the parameter is zero. However, in the remaining parameters, if they are zero and are required, an error in calculations will be caused. In most cases, the value of FI(5) should be large enough to ensure that the expected frequencies are greater than or equal to five. Thus, it is advisable to follow this rule unless stated otherwise.

The parameters required by each particular statistical test are indicated below.

Frequency Test

The following parameters must be specified.

FI(1) = 1

FI(2) = The number of times that this test is repeated.

FI(5) = The number of trials in each test which must be greater than 5 x AA(1).

AA(1) — This parameter must be specified and it is used if the integers are not in the range of numbers required. The maximum value of AA(1) is 1000 as the maximum number of different integers or categories is 1000. If the numbers to be tested are uniform random variables between zero and one, then AA(1) must not be 0.0 or 1.0 since then, the integers obtained would be all zero.

AA(2), AA(3), AA(4), AA(8) — These may be specified but all have a default option, as described previously. Thus, if they are all zero (or blank) the file of numbers is not rewound, the next record is read and all numbers are used, each number being read unformatted. If the numbers to be tested are from a file, the user cannot assume that the file is set at its load point. Therefore, when the first test is applied in a run, the file should be rewound and then moved to the particular record required.

AA(9) — This parameter must be specified if an integer starting value is required.

Serial Test

The parameters to be specified for the serial test are the same as those for the frequency test except that

FI(1) = 2

and the maximum value AA(1) is 50.

Poker Test

The parameters to be specified for the poker test are the same as those for the frequency test except that

$$FI(1) = 3$$

and the parameter FI(3) must also be specified. The parameter supplies the total of numbers considered in each trial and has a maximum value of twenty.

Gap Test for real numbers

The parameters to be specified for the gap test are the same as those for the frequency test except that

$$FI(1) = 4$$

and the parameters FI(4), AA(5) and AA(6) must be specified. These parameters give the end points of the gap length considered in this test and the maximum length of a gap which must be less than 20. It should be remembered that as a gap of length zero is possible, then the number of categories is one more than the maximum length of a gap. In addition, the parameter FI(5) has a slightly different meaning in that it is the number of gaps to be found before the test terminates.

D^2 Test

The parameters to be specified for the D^2 test are the same as those for the frequency test except that

$$FI(1) = 5$$

Gap Test for digits

The parameters to be specified for the gap test for random digits are the same as those for the frequency test except that

$$FI(1) = 6$$

and the parameters AA(5), AA(6) and AA(7) must also be specified.

These parameters provide the significance level of this test and the appropriate normal values for this significance level. The maximum value of AA(1) is 1000.

Runs Test

The parameters to be specified for the runs test are the same as those of the frequency test except that

FI(1) = 7

Also, FI(4) must be specified if runs down are required and FI(5) should be greater than or equal to 4,000.

Maximum Test

The parameters to be specified for the maximum test are the same as those for the frequency test except that

FI(1) = 8

and the parameters FI(4), AA(5), AA(6) and AA(7) must also be specified according to the rules given previously. These parameters indicate the theoretical distribution which is to be compared with the empirical distribution. The maximum value of FI(5) is 5,000.

The Sum Test and the Minimum Test have the same parameters as the maximum test except that

FI(1) = 9 for the minimum test

and

FI(1) = 10 for the sum test

In the statistical tests the maximum values indicated above are due to the limitation of storage space. Thus, if more core store is available, these maximum values can be raised by increasing the appropriate dimensions in the relevant subroutines.

USER and READ Step

To invoke the user option, that is, for the user to supply his own subroutine with a distribution function in the maximum, minimum and sum tests, the following cards must be inserted after the JOB card.

```
//SYSRDR    ACCESS  AP(USER),191='SA45V1'
/*
 Fortran
 Subroutine Cards
/*
:
```

This operation was described in Chapter 4, the resultant effect being that this user's subroutine is compiled and stored under the name USER in the library APLIB. Thus, if the parameters are set correctly, the above subroutine can be employed in a test. The relevant parameters should be set as follows.

FI(4) = 5   AA(5) = 8.0

The same control cards are used to invoke the READ1 and READ2 options. The only differences being that AP(USER) is replaced by AP(READ1) or AP(READ2) and that these subroutines must be written in Assembler language. The subroutines READ1 and READ2 are present in order that the user may supply his own generator to produce numbers. These subroutines are usually written in Assembler language and the rules for writing them are described in the last Chapter. (see Writing of Subroutines) The relevant parameters, which must be set for the READ1 and READ2 options, are:-

AA(8) = 4.0   AA(9) = [1.0 if a starting value is required.
                                   0.0 otherwise.

The only difference between the READ1 and READ2 subroutines is that they are called from different tests with READ1 providing integer numbers and READ2 real numbers. For the USER and READ subroutines provided by the user, the variables AA(6) and AA(7) can be used for input of any extra parameters which may be required.

Error Codes

Most statistical tests have an associated error code, IER. For those tests which use the KS test and thus the subroutine KOLMO, that is the maximum, minimum and sum tests, the error code is as follows:-

IER — is non-zero if the input parameters violate the conventions of the subroutine KOLMO. (For these conventions see Program 3)

— is set to zero if no violations occur.

For the remaining tests that use the χ^2 test, which is applied using the subroutine CDTR, the error code is as follows:-

IER =
- 0 No Error
- -1 Input parameter is invalid
- +1 Output parameter is invalid

(For further information see Program 2)

Example

An example of the data cards employed for a run of two statistical tests is given in Diagram 15.

The first data card is blank which means that the system formats are used to read any subsequent cards for this test. Thus, the second card is read with the following format:-

4I4, I10, 5F5.1, 2F9.4, 2F5.1

These parameters imply that the poker test should be applied. This test is repeated twice with 100 trials in each test and with five numbers in each trial. Since the poker test is being executed the numbers to be tested are integers and as AA(1) = 10.0 and AA(8) = 0.0 they are in the range zero to nine. The numbers are obtained from a file which is rewound and, by default, the next record in the file (in this case it is the first record) is read and every number is used.

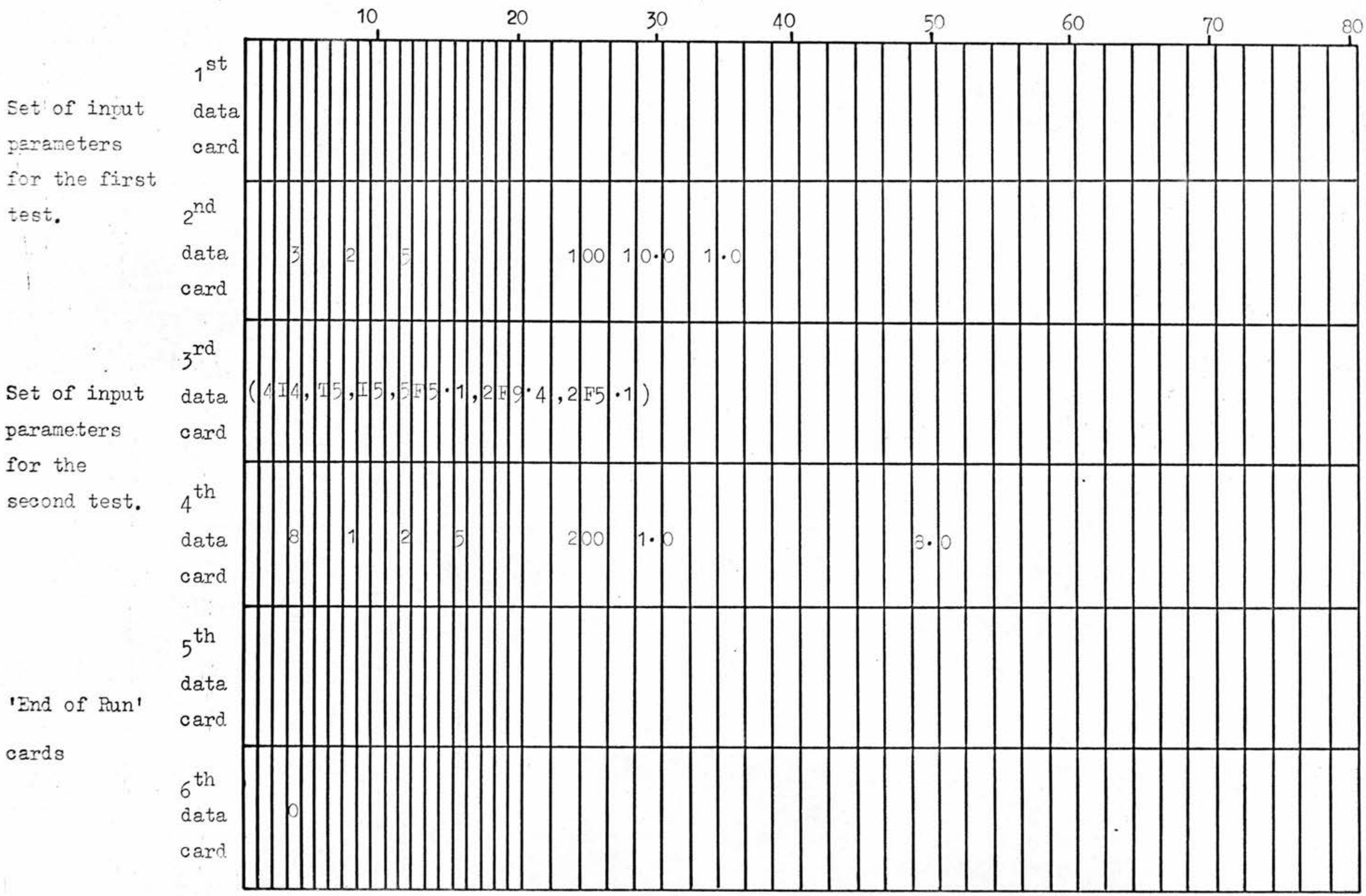
Since the third data card contains the format:-

(4I4, T5, I5, 5F5.1, 2F9.4, 2F5.1)

the fourth card is read with this format. The parameters on the fourth card imply that the second test to be applied is the maximum test. This test is repeated once with 200 trials and two numbers in each trial. The maximum test uses the KS test and as FI(4) = 5 and AA(5) = 8.0 the theoretical distribution used for comparison purposes, is user supplied. Since AA(1) = 1.0 and AA(8) = 0.0 the numbers to be tested are between zero and one and are stored in a file which is not rewound. The next record is read and every number is used.

The last two data cards ends this run as FI(1) is set to the value zero.

DIAGRAM (15)



Set of data cards for a run with two tests

CHAPTER VI. APPLICATIONS.

To demonstrate the applicability of this package for testing numbers, several runs of statistical tests were applied to four different generators.

1. RNDMIN

The first set of tests is concerned with a set of numbers produced by an IBM random number generator which has been adapted for use on the installation in St. Andrews. The method for obtaining data is of the power residue type and takes the residues of the successive powers of a given number. For example

$$y^n \pmod{m} \qquad \text{for } n = 1, 2, \ldots\ldots$$

(IBM GC 20-8011-0)

The numbers obtained from this generator are between zero and one and they were stored in an unformatted file on disc. All the statistical tests that have previously been described were applied and the results are given on the succeeding pages. These results are then summarized in Table 1.

```
THE FREQUENCY TEST
THE NUMBER OF TRAILS IS   1000  REPEATED      5 TIMES

****************************************************************************************************

THE NUMBER IN EACH CATEGORY IS
    98.00    109.00     95.00    112.00    111.00     85.00     93.00     89.00     88.00    120.00
THE CHI-SQUARED VALUE IS 0.1313989E 02  WITH  9.00  DEGREES OF FREEDOM.
THE PROBILITY OF A WORSE VALUE OF CHI-SQUARE IS 0.1563777
THE ERROR CODE IS     0
THIS DATA IS CONSIDERED TO BE RANDOM

****************************************************************************************************

THE NUMBER IN EACH CATEGORY IS
   106.00     88.00     99.00    108.00    116.00     81.00    109.00     97.00    101.00     95.00
THE CHI-SQUARED VALUE IS 0.9779785E 01  WITH  9.00  DEGREES OF FREEDOM.
THE PROBILITY OF A WORSE VALUE OF CHI-SQUARE IS 0.3686053
THE ERROR CODE IS     0
THIS DATA IS CONSIDERED TO BE RANDOM

****************************************************************************************************

THE NUMBER IN EACH CATEGORY IS
    89.00     98.00     97.00    120.00    107.00    103.00     88.00    113.00     90.00     95.00
THE CHI-SQUARED VALUE IS 0.1029980E 02  WITH  9.00  DEGREES OF FREEDOM.
THE PROBILITY OF A WORSE VALUE OF CHI-SQUARE IS 0.3267640
THE ERROR CODE IS     0
THIS DATA IS CONSIDERED TO BE RANDOM

****************************************************************************************************

THE NUMBER IN EACH CATEGORY IS
    90.00     94.00    108.00    100.00    107.00    116.00     90.00    101.00    103.00     91.00
```

```
THE CHI-SQUARED VALUE IS 0.6959961E 01  WITH  9.00  DEGREES OF FREEDOM.
THE PROBILITY OF A WORSE VALUE OF CHI-SQUARE IS 0.6412882
THE ERROR CODE IS      0
THIS DATA IS CONSIDERED TO BE RANDOM

**********************************************************************************************************


THE NUMBER IN EACH CATEGORY IS
   104.00     92.00    108.00    119.00    100.00     90.00    106.00    105.00     79.00     97.00
THE CHI-SQUARED VALUE IS 0.1115991E 02  WITH  9.00  DEGREES OF FREEDOM.
THE PROBILITY OF A WORSE VALUE OF CHI-SQUARE IS 0.2649069
THE ERROR CODE IS      0
THIS DATA IS CONSIDERED TO BE RANDOM

**********************************************************************************************************


THE SERIAL TEST
THE NUMBER OF TRAILS IS   1000REPEATED      4   TIMES

**********************************************************************************************************
THE NUMBER IN EACH CATEGORY IS
    7.0    5.0    7.0    8.0   13.0    8.0    8.0    9.0    8.0    8.0   11.0   11.0    5.0   10.0    8.0
   12.0   19.0   14.0   10.0   11.0   12.0   13.0    9.0   12.0   12.0   14.0   10.0   13.0    6.0   13.0
   13.0   12.0    9.0    8.0   11.0    9.0   13.0    6.0   11.0   10.0    7.0    6.0   12.0   12.0   12.0
    8.0   14.0   12.0    4.0   15.0   12.0   14.0    4.0   11.0   11.0    8.0   13.0    9.0   12.0    9.0
   10.0   10.0    6.0   16.0   13.0   10.0    7.0   10.0   11.0    8.0    9.0   11.0   13.0   16.0   13.0
   14.0    8.0   11.0    7.0    6.0    5.0   11.0    8.0   15.0    6.0    6.0   11.0    9.0   12.0   11.0
    6.0   10.0   12.0    9.0    9.0    8.0   10.0   11.0    4.0    6.0
THE CHI-SQUARED VALUE IS 0.8929980E 02  WITH 99.00  DEGREES OF FREEDOM
THE PROBABILITY OF A WORSE VALUE OF CHI-SQUARED IS  0.7471
WITH ERROR CODE  0
THIS DATA IS CONSIDERED TO BE RANDOM

**********************************************************************************************************


THE NUMBER IN EACH CATEGORY IS
```

```
  11.0   7.0   6.0  12.0   5.0   8.0  18.0  12.0   9.0  12.0  14.0   7.0   8.0   6.0  12.0
  11.0  13.0  14.0   8.0   8.0  12.0   7.0  13.0   8.0  14.0   8.0   6.0   5.0  16.0  11.0
  12.0   9.0  15.0  10.0  13.0   9.0  11.0  13.0   9.0   6.0   8.0   9.0   9.0  11.0   9.0
   9.0   9.0   8.0   4.0   9.0  12.0  13.0  12.0   7.0  10.0   7.0  14.0   5.0  11.0  10.0
   7.0   6.0   8.0   8.0  11.0  10.0  11.0  15.0  14.0   9.0  11.0  11.0   8.0   6.0   6.0
  13.0   5.0   9.0  11.0   5.0  13.0  13.0   8.0  12.0  18.0   5.0  12.0  12.0   8.0  13.0
  10.0  10.0  12.0   9.0  14.0  12.0  11.0  11.0  10.0  10.0
THE CHI-SQUARED VALUE IS 0.8929980E 02  WITH 99.00  DEGREES OF FREEDOM
THE PROBABILITY OF A WORSE VALUE OF CHI-SQUARED IS  0.7471
WITH ERROR CODE  0
THIS DATA IS CONSIDERED TO BE RANDOM

*******************************************************************************************************

THE NUMBER IN EACH CATEGORY IS
   7.0  11.0   9.0   6.0   9.0   8.0   6.0  10.0   6.0  10.0   6.0   8.0  12.0  10.0  10.0
   8.0  16.0  11.0   7.0  11.0   8.0  14.0   6.0  13.0  16.0  10.0  11.0  12.0  12.0  16.0
  14.0  16.0  11.0   9.0  10.0  11.0   8.0  11.0  17.0  13.0   7.0   9.0   7.0   7.0  13.0
  14.0   6.0  15.0   9.0   7.0   6.0  12.0   8.0  12.0  14.0  15.0   7.0   8.0   9.0  13.0
  13.0  11.0   9.0   9.0  11.0   8.0   8.0   7.0   7.0  11.0  13.0  19.0   6.0  11.0   6.0
   8.0   8.0  12.0   8.0  12.0   8.0   8.0  11.0  10.0   9.0   8.0   5.0  15.0   7.0  16.0
   6.0  10.0   8.0  10.0  12.0  10.0   6.0  13.0   6.0   9.0
THE CHI-SQUARED VALUE IS 0.9569995E 02  WITH 99.00  DEGREES OF FREEDOM
THE PROBABILITY OF A WORSE VALUE OF CHI-SQUARED IS  0.5752
WITH ERROR CODE  0
THIS DATA IS CONSIDERED TO BE RANDOM

*******************************************************************************************************

THE NUMBER IN EACH CATEGORY IS
   7.0   8.0  12.0   7.0  10.0  12.0   7.0  14.0   7.0  12.0   9.0  13.0  12.0  12.0   5.0
   7.0   7.0   7.0  13.0   7.0   6.0  13.0   7.0   8.0   9.0  11.0   9.0   9.0   9.0  14.0
  14.0   9.0   8.0   6.0  16.0   6.0   6.0   8.0  11.0   8.0   7.0   5.0  14.0  15.0  13.0
   8.0   9.0  15.0  16.0   9.0   8.0  15.0  10.0  13.0  10.0   9.0  11.0   3.0   8.0   5.0
  10.0   7.0  12.0   8.0  10.0   7.0   9.0   8.0  12.0  15.0   8.0  14.0  11.0  11.0   9.0
```

```
   9.0  10.0  11.0   8.0  11.0  11.0  10.0  11.0  13.0  12.0  13.0  11.0  12.0   9.0  10.0
  11.0  14.0  14.0  11.0  11.0   8.0  10.0   7.0   9.0  16.0
THE CHI-SQUARED VALUE IS 0.8229980E 02  WITH 99.00  DEGREES OF FREEDOM
THE PROBABILITY OF A WORSE VALUE OF CHI-SQUARED IS  0.8874
WITH ERROR CODE  0
THIS DATA IS CONSIDERED TO BE RANDOM

***************************************************************************************************


THE POKER TEST
THE NUMBER OF TRIALS IS   1000 TAKEN      5  AT A TIME AND REPEATED      2   TIMES

***************************************************************************************************


THE NUMBER IN EACH CATEGORY IS
   0.0     10.00  189.00  528.00  273.00
STIRGLING'S NUMBERS FOR THE VALUE  5  ARE
    1    15    25    10     1
THE CHI-SQUARED VALUE IS 0.5404541E 01  WITH  3.00  DEGREES OF FREEDOM
THE PROBABILITY OF A WORSE VALUE OF CHI-SQUARED IS  0.1445
WITH ERROR CODE  0
THIS DATA IS CONSIDERED TO BE RANDOM

***************************************************************************************************

THE NUMBER IN EACH CATEGORY IS
   0.0      7.00  178.00  511.00  304.00
STIRGLING'S NUMBERS FOR THE VALUE  5  ARE
    1    15    25    10     1
THE CHI-SQUARED VALUE IS 0.3331299E 01  WITH  3.00  DEGREES OF FREEDOM
THE PROBABILITY OF A WORSE VALUE OF CHI-SQUARED IS  0.3433
WITH ERROR CODE  0
THIS DATA IS CONSIDERED TO BE RANDOM

***************************************************************************************************
```

```
THE GAP TEST
THE NUMBER OF TRIALS IS   1000 REPEATED       3  TIMES
THE GAP IS BETWEEN 0.0 AND 0.50

*****************************************************************************************************


THE NUMBER IN EACH CATEGORY IS
 267.00  118.00   67.00   27.00   18.00    6.00    5.00    6.00
 486.00
THE CHI-SQUARED VALUE IS 0.5051758E 01  WITH  8.0  DEGREES OF FREEDOM
THE PROBABILITY OF A WORSE VALUE OF CHI-SQUARE IS 0.7520301E 00 WITH ERROR CODE  0
THIS DATA IS CONSIDERED TO BE RANDOM

*****************************************************************************************************


THE NUMBER IN EACH CATEGORY IS
 261.00  116.00   69.00   28.00   17.00    7.00    1.00    2.00
 499.00
THE CHI-SQUARED VALUE IS 0.5445801E 01  WITH  8.0  DEGREES OF FREEDOM
THE PROBABILITY OF A WORSE VALUE OF CHI-SQUARE IS 0.7090371E 00 WITH ERROR CODE  0
THIS DATA IS CONSIDERED TO BE RANDOM

*****************************************************************************************************


THE NUMBER IN EACH CATEGORY IS
 239.00  112.00   57.00   32.00   12.00    7.00   12.00    7.00
 522.00
THE CHI-SQUARED VALUE IS 0.2345190E 02  WITH  8.0  DEGREES OF FREEDOM
THE PROBABILITY OF A WORSE VALUE OF CHI-SQUARE IS 0.2830148E-02 WITH ERROR CODE  0
THIS DATA CANNOT BE CONSIDERED RANDOM
```

```
*****************************************************************************************************

THE D SQUARED TEST
THE NUMBER OF TRAILS IS   1000 REPEATED      1  TIMES

*****************************************************************************************************

THE NUMBER IN EACH CATEGORY IS
   221.00     175.00     159.00     105.00      87.00
    73.00      67.00      36.00      27.00      22.00
    14.00       7.00       3.00       3.00       1.00
     0.0        0.0        0.0        0.0        0.0
THE CHI-SQUARE VALUE IS 0.1039453E 02   WITH 19.0  DEGREES OF FREEDOM
THE PROBABILITY OF A WORSE VALUE OF CHI-SQUARE IS 0.9425329E 00 WITH ERROR CODE  0
THIS DATA IS  SLIGHTY SUSPECT OF NOT BEING RANDOM

*****************************************************************************************************

THE GAP TEST  FOR RANDOM DIGITS
THE NUMBER OF TRAILS IS    2000 THE TEST IS AT0.0500  LEVEL OF SIGNIFICANCE

*****************************************************************************************************

    1995.0     1997.0     1983.0     1986.0     1989.0
    1978.0     1999.0     1996.0     1990.0     2000.0
   43041.0    34327.0    37837.0    31360.0    32111.0
   40406.0    39997.0    34938.0    36372.0    46210.0
    182    216    189    219    230
    194    189    200    205    176
FOR DIGIT  1 THE MEAN IS 10.96  WHICH IS NOT SIGNIFICANTLY DIFFERENT FROM THE EXPECTED MEAN
```

```
AT 0.5000000E-01 LEVEL OF SIGNIFICANCE.
THE VARIANCE FAILS THE TEST WITH VALUE 0.1163337E 03

****************************************************************************************************


FOR DIGIT  2 THE MEAN IS  9.25  WHICH IS NOT SIGNIFICANTLY DIFFERENT FROM THE EXPECTED MEAN
AT 0.5000000E-01 LEVEL OF SIGNIFICANCE.
THE VARIANCE IS 73.444 WHICH IS NOT SIGNIFICANT

****************************************************************************************************


FOR DIGIT  3 THE MEAN IS 10.49  WHICH IS NOT SIGNIFICANTLY DIFFERENT FROM THE EXPECTED MEAN
AT 0.5000000E-01 LEVEL OF SIGNIFICANCE.
THE VARIANCE IS 90.112 WHICH IS NOT SIGNIFICANT

****************************************************************************************************


FOR DIGIT  4 THE MEAN IS  9.07  WHICH IS NOT SIGNIFICANTLY DIFFERENT FROM THE EXPECTED MEAN
AT 0.5000000E-01 LEVEL OF SIGNIFICANCE.
THE VARIANCE IS 60.959 WHICH IS NOT SIGNIFICANT

****************************************************************************************************


FOR DIGIT    5 THE MEAN FAILS THE TEST OF SIGNIFICANCE WITH VALUE 0.8647825E 01
THE VARIANCE IS 64.828 WHICH IS NOT SIGNIFICANT

****************************************************************************************************


FOR DIGIT  6 THE MEAN IS 10.20  WHICH IS NOT SIGNIFICANTLY DIFFERENT FROM THE EXPECTED MEAN
AT 0.5000000E-01 LEVEL OF SIGNIFICANCE.
```

```
THE VARIANCE IS104.322 WHICH IS NOT SIGNIFICANT

*************************************************************************************************************************

FOR DIGIT  7 THE MEAN IS 10.58  WHICH IS NOT SIGNIFICANTLY DIFFERENT FROM THE EXPECTED MEAN
AT 0.5000000E-01 LEVEL OF SIGNIFICANCE.
THE VARIANCE IS 99.757 WHICH IS NOT SIGNIFICANT

*************************************************************************************************************************

FOR DIGIT  8 THE MEAN IS  9.98  WHICH IS NOT SIGNIFICANTLY DIFFERENT FROM THE EXPECTED MEAN
AT 0.5000000E-01 LEVEL OF SIGNIFICANCE.
THE VARIANCE IS 75.090 WHICH IS NOT SIGNIFICANT

*************************************************************************************************************************

FOR DIGIT  9 THE MEAN IS  9.71  WHICH IS NOT SIGNIFICANTLY DIFFERENT FROM THE EXPECTED MEAN
AT 0.5000000E-01 LEVEL OF SIGNIFICANCE.
THE VARIANCE IS 83.192 WHICH IS NOT SIGNIFICANT

*************************************************************************************************************************

FOR DIGIT 10 THE MEAN IS 11.36  WHICH IS NOT SIGNIFICANTLY DIFFERENT FROM THE EXPECTED MEAN
AT 0.5000000E-01 LEVEL OF SIGNIFICANCE.
THE VARIANCE FAILS THE TEST WITH VALUE 0.1334244E 03

*************************************************************************************************************************

THE RUNS TEST
THE NUMBER OF TRAILS IS    4000 REPEATED     2 TIMES
```

```
RUNS UP

**************************************************************************************************************

   693.00    801.00    376.00     99.00     30.00      5.00
THE CHI-SQUARED VALUE IS 0.5529838E 01  WITH  6.00  DEGREES OF FREEDOM
THE PROBABILITY OF A WORSE VALUE OF CHI-SQUARE IS  0.4779
WITH ERROR CODE  0
THIS DATA IS CONSIDERED TO BE RANDOM

**************************************************************************************************************

   675.00    812.00    380.00    104.00     23.00      5.00
THE CHI-SQUARED VALUE IS 0.2037051E 01  WITH  6.00  DEGREES OF FREEDOM
THE PROBABILITY OF A WORSE VALUE OF CHI-SQUARE IS  0.9163
WITH ERROR CODE  0
THIS DATA IS  SLIGHTY SUSPECT OF NCT BEING RANDOM

**************************************************************************************************************

THE RUNS TEST
THE NUMBER OF TRAILS IS    4000 REPEATED     2 TIMES
RUNS DOWN

**************************************************************************************************************

   686.00    787.00    390.00    108.00     20.00      6.00
THE CHI-SQUARED VALUE IS 0.6790687E 01  WITH  6.00  DEGREES OF FREEDOM
THE PROBABILITY OF A WORSE VALUE OF CHI-SQUARE IS  0.3406
WITH ERROR CODE  0
THIS DATA IS CONSIDERED TO BE RANDOM
```

```
****************************************************************************************************

   677.00     815.00     382.00      98.00      26.00       4.00
THE CHI-SQUARED VALUE IS 0.2517718E 01  WITH  6.00  DEGREES OF FREEDOM
THE PROBABILITY OF A WORSE VALUE OF CHI-SQUARE IS  0.8665
WITH ERROR CODE   0
THIS DATA IS CONSIDERED TO BE RANDOM

****************************************************************************************************


THE MAXIMUN TEST
THE NUMBER OF TRAILS IS    300 REPEATED       3  TIMES.THE NUMBER IN EACH TRAIL IS       3
THE PROBABILITY OF THE STATISTIC BEING GREATER THAN OR EQUAL TO 0.1128179E 01
IF THE HYPOTHESIS IS TRUE IS 0.1567805E 00 WITH ERROR CODE   0
THIS DATA IS CONSIDERED TO BE RANDOM

****************************************************************************************************


THE PROBABILITY OF THE STATISTIC BEING GREATER THAN OR EQUAL TO 0.5811216E 00
IF THE HYPOTHESIS IS TRUE IS 0.8882495E 00 WITH ERROR CODE   0
THIS DATA IS CONSIDERED TO BE RANDOM

****************************************************************************************************


THE PROBABILITY OF THE STATISTIC BEING GREATER THAN OR EQUAL TO 0.6015473E 00
IF THE HYPOTHESIS IS TRUE IS 0.8622274E 00 WITH ERROR CODE   0
THIS DATA IS CONSIDERED TO BE RANDOM

****************************************************************************************************
```

```
THE MINIMUM TEST
THE NUMBER OF TRAILS IS    400 REPEATED      4  TIMES.THE NUMBER IN EACH TRAIL IS      4

***************************************************************************************************************

THE PROBABILITY OF THE STATISTIC BEING GREATER THAN OR EQUAL TO 0.6234658E 00
IF THE HYPOTHESIS IS TRUE IS 0.8317724E 00 WITH ERROR CODE  0
THIS DATA IS CONSIDERED TO BE RANDOM

***************************************************************************************************************

THE PROBABILITY OF THE STATISTIC BEING GREATER THAN OR EQUAL TO 0.1040580E 01
IF THE HYPOTHESIS IS TRUE IS 0.2290134E 00 WITH ERROR CODE  0
THIS DATA IS CONSIDERED TO BE RANDOM

***************************************************************************************************************

THE PROBABILITY OF THE STATISTIC BEING GREATER THAN OR EQUAL TO 0.8521736E 00
IF THE HYPOTHESIS IS TRUE IS 0.4620239E 00 WITH ERROR CODE  0
THIS DATA IS CONSIDERED TO BE RANDOM

***************************************************************************************************************

THE PROBABILITY OF THE STATISTIC BEING GREATER THAN OR EQUAL TO 0.5027854E 00
IF THE HYPOTHESIS IS TRUE IS 0.9621357E 00 WITH ERROR CODE  0
THIS DATA IS SUSPECT OF NOT BEING RANDOM

***************************************************************************************************************
```

```
THE SUM TEST
THE NUMBER OF TRAILS IS    500 REPEATED       2  TIMES.THE NUMBER IN EACH TRAIL IS      6

*****************************************************************************************************************

THE PROBABILITY OF THE STATISTIC BEING GREATER THAN OR EQUAL TO 0.1009038E 01
IF THE HYPOTHESIS IS TRUE IS 0.2604373E 00 WITH ERROR CODE  0
THIS DATA IS CONSIDERED TO BE RANDOM

*****************************************************************************************************************

THE PROBABILITY OF THE STATISTIC BEING GREATER THAN OR EQUAL TO 0.6206383E 00
IF THE HYPOTHESIS IS TRUE IS 0.8358340E 00 WITH ERROR CODE  0
THIS DATA IS CONSIDERED TO BE RANDOM

*****************************************************************************************************************
```

TABLE 1

TEST NUMBERS

Generator	Starting value	1	2	3	4	5	6	7 UP	7 DOWN	8	9	10
RNDMIN	971463237	AAAAA	AAAA	AA	bm AAA	B	A	AB	AA	AAA	AAAC	AA
	963214512	ABA	CA	AA	am BAA	AA	A	ABA	CAA	AAA	AD	BAA
	883296567	AA	ACA	AAAB	bm AA	CDA	A	AACA	A	CBA	ABBA	ABB
	79236598	AA	AAAA	BA	bm CA	CAA	A	AACA	AA	AAA	AAAA	AAA

where the letters

'A' signifies random

'B' signifies slightly suspect of not being random

'C' signifies suspect of not being random

'D' signifies not random

and the letters

'am' signifies above the mean

'bm' signifies below the mean

The assigned 'test numbers' are the same as those described in Chapter 5 under the section called DATA.

From the above table, it is evident that the numbers produced by this generator pass the tests well and are hence random. However, if an application concerned a particular attribute then further testing on this generator about this attribute would be advisable.

2. KLRAND

The second application illustrated in this work concerns a multiplicative congruential pseudo-random number generator which is attributed to D. H. Lehmer and is of the form:

$$Y_{i+1} = AY_i \pmod{p}$$

(J. Whiltlesey)

This generator has also been adapted for use on the installation in St. Andrews. The numbers obtained from this generator are integers between 1 and $2^{31} - 1$. However, for the purposes of testing, they were multiplied by a real constant to bring them into the range zero to one. The numbers were stored on a disc and all the previously mentioned statistical tests were applied. A summary of the results is given in Table 2.

TABLE 2

TEST NUMBERS

Generator	Starting Value	1	2	3	4	5	6	7 UP	7 DOWN	8	9	10
KLRAND	932165487	AA	CCA	AAAA	bm AA	AAA	A	AAAA	A	AAA	AABB	AAA
	563987231	ACA	DA	AA	am AAD	AA	A	BAA	AAA	ACAA	AA	AAAA
	241376785	DAACA	BAAA	AA	bm AAC	A	A	BA	AC	AAA	BACB	AA
	793216548	AA	CAAB	AA	bm AA	AAB	A	AAAA	AA	AAA	AAB	AAC

From the above table, it can be seen that this generator gives better results for some of the tests than the previous generator but for others it is not as good. However, it seems to pass the tests satisfactorily although there are one or two bad results. Thus for general purposes, this generator appears adequate. Again, for tests concerning particular attributes, this generator should be investigated more thoroughly.

3. The SINE Function

In general the SINE function is not considered to give very satisfactory random numbers due to its cycling property. For this reason it was chosen as an illustration example since the statistical tests must be able to pick out bad random number sequences as well as good ones.

After each sine value has been calculated it is stored as a random number. This value is then multiplied by a hundred and the resultant residue modulo ninety is used as the next argument for the sine function. The numbers obtained from this generator are between zero and one. The statistical tests are all applied to these numbers and the results are summarized in Table 3.

TABLE 3

		TEST NUMBERS										
Generator	Starting Value	1	2	3	4	5	6	7 UP	7 DOWN	8	9	10
SIN	20.0	DD	DDDD	DD	bm DD	DDD	D	DDDD	DD	DDD	DDDD	DDD
	35.0	DDDDD	DDDD	DD	bm DDD	D	D	DD	DD	DDD	DDDD	DD
	50.0	DD	DDD	DDDD	bm DD	DDD	D	DDDD	D	DDD	DDDD	DDD
	70.0	DDD	DD	DD	am DD	DD	D	DDD	DDD	DDDD	DD	DDDD

From Table 3 it can be seen that in all tests this generator gives very bad results. Thus, this form of the sine generator cannot be considered as a random number generator.

This generator may be altered to give slightly better results but on the whole the sine function would not provide a satisfactory random number generator

4. The Fibonacci Sequence

Although Fibonacci sequences are fairly easy to generate, they are considered to give bad random numbers because of short periods. For this reason, a Fibonacci generator was devised and used as an example in this work. In general two numbers X_n and X_{n-1} are considered at a time and the generator takes the form

$$X_{n+1} = (X_n + X_{n-1}) \pmod{m}$$

In order that the generator may be used in this example, both X_n and X_{n-1} are double precis ion real numbers. After they have been added together they are stored in a double register and the middle four bytes extracted. Before this resulting number can be used, it has to be normalised and then adjusted to lie between zero and one. This is implemented by overwriting the exponent part of that number with the appropriate exponent to bring it into the required range. The number obtained is a single precission real number and is stored in a file on disc. The remaining numbers are obtained by repeating the above process for X_{n+2}, X_{n+3}, etcetera. The results from applying all the statistical tests in the package to these numbers are summarized in Table 4.

It was found with test 4, the gap test, that the quantity of numbers in the file had to be increased greatly, or alternatively, the number of gaps to be found reduced drastically to obtain results. Due to the time available and storage facilities, the latter course was adopted. However, more extensive testing could be carried out in this region especially with the second starting value.

TABLE 4

TEST NUMBERS

Generator	Starting Value	1	2	3	4	5	6	7 UP	7 DOWN	8	9	10
Fibonacci	354321.98 799632	DDD	DD	DD	bm DDD	DD	D	DDD	DDD	DDDD	DD	DDDD
	96.3214578 55671	DD	DDD	DDDD	bm AB	DDD	C	DDDD	D	DDD	DDDD	DDD
	3543.2198 799632	DD	DDDD	DD	bm DD	DDD	D	DDDD	DD	DDD	DDDD	DDD
	123.86547 321883	DDDDD	DDDD	DD	am DDD	D	D	DD	DDD	DDD	DDDD	DD

From the above table it can be seen that in most tests, this generator gives very bad results. Thus overall this Fibonacci generator does not give satisfactory random number sequences. There are other Fibonacci generators that give better overall results, however, those tests which are concerned with periodicity still give rise to unsatisfactory random behaviour. Thus, for most cases, the Fibonacci generators would be best avoided in providing random numbers.

CHAPTER VII. FUTURE DEVELOPMENTS.

Addition of more tests

It is hoped that in the future more tests may be added to this package. These may include further empirical tests but should also include some theoretical ones. Theoretical tests cover a wide range of study, for example, in the study of the periods of a generated sequence (E. Bofinger et al 1961, S. Yamada 1961), in serial correlation (M. Greenberger 1961), in autocorrelation (B. Jannson 1964) and in the study of the moments of a sequence (D. Teichroew 1965). Another useful theoretical test is the Fourier analysis, not only of the generator itself, but of the autocorrelation function. This analysis may give more insight into hidden cycling. (E.S. Page 1967, R. Conveyou et al 1967)

Improvement of Package

This package could have been improved, if more time had been available, by the addition of certain extra techniques. These include the facility for the user to define his own dimensions.

Another technique, for use in the saving of storage space, involves the incorporation of as many variables as possible into common areas. Thus arrays and indices in one test may use the same storage space as that used in another test. A further way to save on space is to adopt the usage of half words wherever possible.

It may be found with a wide range of users that restricting the length of a record is impractical. In this case the input data subroutines must be adapted.

The purpose of this thesis has been to provide a package for the testing of random number sequences. However, although it was in no way intended to perfect a new generator or to improve existing ones during this work, it became evident that such a course of study would offer interesting possibilities.

APPENDIX I

LISTING OF THE SUBROUTINES IN THIS PACKAGE

TABLE OF CONTENTS

```
C                              PRCGRAM(1)
C
C.....................................................................
C
C        PURPOSE
C        MAIN CALLING PRCGRAM -- APMAIN
C
C.....................................................................
C
      INTEGER T1,T2,D1,D2
      T1=KLOCK(D1,D2)
      INTEGER FI,FLAG,FMT2
      INTEGER IN(8),JN(8),FMT(8)
      COMMCN FI(5),AA(9),NUM,FMT2(8),ISTART
      COMMON IPRINT,IRED,JRED
      INTEGER*4 W/'&'/
      IRED=3
      JRED=5
      IPRINT=6
      FLAG=0
    5 READ(JRED,1C)IN,NUM,JN
   1C FORMAT(8A4,I4,8A4)
      SWITCH=0.0
      CALL GETFMT(IN,SWITCH,FMT)
      SWITCH=1.C
      CALL GETFMT(JN,SWITCH,FMT2)
      READ(JRED,FMT)FI,AA
      IF(FI(1).EQ.0)GC TC 18C
      IF(AA(9).EQ.0.0)GC TC 12
      READ(JRED,11)ISTART
   11 FORMAT(I10)
   12 K=FI(1)
      IF(AA(2)-1)14,15,14
   15 REWIND IREC
   14 JREC=AA(3)-1
```

```
      IF(AA(3).EQ.0.0.OR.AA(3).EQ.1.0)GO TO 16
      DO 13 IREC=1,JREC
      READ(IRED)
   13 CONTINUE
   16 GO TO(20,30,40,50,60,70,80,90,92,94),K
   20 IF(FLAG.EQ.1)GO TO 25
      CALL LOAD('APCHI')
   25 FLAG=1
      CALL FREQ
      GO TO 5
   30 IF(FLAG.EQ.1)GO TO 35
      CALL LOAD('APCHI')
   35 FLAG=1
      CALL SERIAL
      GO TO 5
   40 IF(FLAG.EQ.1)GO TO 45
      CALL LOAD('APCHI')
   45 FLAG=1
      CALL POKER
      GO TO 5
   50 IF(FLAG.EQ.1)GO TO 55
      CALL LOAD('APCHI')
   55 FLAG=1
      CALL GAP
      GO TO 5
   60 IF(FLAG.EQ.1)GO TO 65
      CALL LOAD('APCHI')
   65 FLAG=1
      CALL DSQUR
      GO TO 5
   70 IF(FLAG.EQ.3)GO TO 75
      CALL LOAD('APOTHR')
   75 FLAG=3
      CALL GAPRD
      GO TO 5
```

```
 80 IF(FLAG.EQ.1)GC TC 85
    CALL LOAD('APCHI')
 85 FLAG=1
    CALL RUNS
    GO TC 5
 90 IF(FLAG.EQ.2)GC TC 91
    CALL LOAD('APKMSM')
 91 FLAG=2
    CALL MAX
    GO TC 5
 92 IF(FLAG.EQ.2)GC TC 93
    CALL LOAD('APKMSM')
 93 FLAG=2
    CALL MIN
    GO TC 5
 94 IF(FLAG.EQ.2)GC TC 95
    CALL LOAD('APKMSM')
 95 FLAG=2
    CALL SUM
     GC TC 5
180 WRITE(IPRINT,185)W
185 FORMAT(' 'A1)
    T2=KLOCK(D1,D2)
    T1=(T2-T1)/50
    WRITE(IPRINT,190)T1
190 FORMAT(' THE TIME TAKEN IN SECCNDS IS' I8)
    STCP
    END
```

PROGRAM(2)

```
C                                                                       CDTR  10
C     ..................................................................CDTR  20
C                                                                       CDTR  30
C        SUBROUTINE CDTR                                                CDTR  40
C                                                                       CDTR  50
C        PURPOSE                                                        CDTR  60
C           COMPUTES P(X) = PROBABILITY THAT THE RANDOM VARIABLE U,     CDTR  70
C           DISTRIBUTED ACCORDING TO THE CHI-SQUARE DISTRIBUTION WITH G CDTR  80
C           DEGREES OF FREEDOM, IS LESS THAN OR EQUAL TO X.  F(G,X), THECDTR  90
C           ORDINATE OF THE CHI-SQUARE DENSITY AT X, IS ALSO COMPUTED.  CDTR 100
C                                                                       CDTR 110
C        USAGE                                                          CDTR 120
C           CALL CDTR(X,G,P,D,IER)                                      CDTR 130
C                                                                       CDTR 140
C        DESCRIPTION OF PARAMETERS                                      CDTR 150
C           X   - INPUT SCALAR FOR WHICH P(X) IS COMPUTED.              CDTR 160
C           G   - NUMBER OF DEGREES OF FREEDOM OF THE CHI-SQUARE        CDTR 170
C                 DISTRIBUTION.  G IS A CONTINUOUS PARAMETER.           CDTR 180
C           P   - OUTPUT PROBABILITY.                                   CDTR 190
C           D   - OUTPUT DENSITY.                                       CDTR 200
C           IER - RESULTANT ERROR CODE WHERE                            CDTR 210
C               IER= 0 --- NO ERROR                                     CDTR 220
C               IER=-1 --- AN INPUT PARAMETER IS INVALID.  X IS LESS    CDTR 230
C                          THAN 0.0, OR G IS LESS THAN 0.5 OR GREATER   CDTR 240
C                          THAN 2*10**(+5).  P AND D ARE SET TO -1.E75. CDTR 250
C               IER=+1 --- INVALID OUTPUT.  P IS LESS THAN ZERO OR      CDTR 260
C                          GREATER THAN ONE, OR SERIES FOR T1 (SEE      CDTR 270
C                          MATHEMATICAL DESCRIPTION) HAS FAILED TO      CDTR 280
C                          CONVERGE.  P IS SET TO 1.E75.                CDTR 290
C                                                                       CDTR 300
C        REMARKS                                                        CDTR 310
C           SEE MATHEMATICAL DESCRIPTION.                               CDTR 320
C                                                                       CDTR 330
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED                  CDTR 340
```

```
C           DLGAM
C           NDTR
C
C        METHOD
C           REFER TO R.E. BARGMANN AND S.P. GHOSH, STATISTICAL
C           DISTRIBUTION PROGRAMS FOR A COMPUTER LANGUAGE,
C           IBM RESEARCH REPORT RC-1094, 1963.
C
C     ..................................................................
C
      SUBROUTINE CDTR(X,G,P,D,IER)
      DOUBLE PRECISION XX,DLXX,X2,DLX2,GG,G2,DLT3,THETA,THP1,
     1GLG2,DD,T11,SER,CC,XI,FAC,TLOG,TERM,GTH,A2,A,B,C,DT2,DT3,THPI
C
C        TEST FOR VALID INPUT DATA
C
      IF(G-(.5-1.E-5)) 590,10,10
   10 IF(G-2.E+5) 20,20,590
   20 IF(X) 590,30,30
C
C        TEST FOR X NEAR 0.0
C
   30 IF(X-1.E-8) 40,40,80
   40 P=0.0
      IF(G-2.) 50,60,70
   50 D=1.E75
      GO TO 610
   60 D=0.5
      GO TO 610
   70 D=0.0
      GO TO 610
C
C        TEST FOR X GREATER THAN 1.E+6
C
   80 IF(X-1.E+6) 100,100,90
```

```
   90 D=0.0                                                             CDTR 700
      P=1.0                                                             CDTR 710
      GO TO 610                                                         CDTR 720
C                                                                       CDTR 730
C        SET PROGRAM PARAMETERS                                         CDTR 740
C                                                                       CDTR 750
  100 XX=DBLE(X)                                                        CDTR 760
      DLXX=DLOG(XX)                                                     CDTR 770
      X2=XX/2.D0                                                        CDTR 780
      DLX2=DLOG(X2)                                                     CDTR 790
      GG=DBLE(G)                                                        CDTR 800
      G2=GG/2.D0                                                        CDTR 810
C                                                                       CDTR 820
C        COMPUTE ORDINATE                                               CDTR 830
C                                                                       CDTR 840
      CALL DLGAM(G2,GLG2,IOK)                                           CDTR 850
      DD=(G2-1.D0)*DLXX-X2-G2*.6931471805599453 -GLG2                   CDTR 860
      IF(DD-1.68D02) 110,110,120                                        CDTR 870
  110 IF(DD+1.68D02) 130,130,140                                        CDTR 880
  120 D=1.E75                                                           CDTR 890
      GO TO 150                                                         CDTR 900
  130 D=0.0                                                             CDTR 910
      GO TO 150                                                         CDTR 920
  140 DD=DEXP(DD)                                                       CDTR 930
      D=SNGL(DD)                                                        CDTR 940
C                                                                       CDTR 950
C        TEST FOR G GREATER THAN 1000.0                                 CDTR 960
C        TEST FOR X GREATER THAN 2000.0                                 CDTR 970
C                                                                       CDTR 980
  150 IF(G-1000.) 160,160,180                                           CDTR 990
  160 IF(X-2000.) 190,190,170                                           CDTR1000
  170 P=1.0                                                             CDTR1010
      GO TO 610                                                         CDTR1020
  180 A=DLOG(XX/GG)/3.D0                                                CDTR1030
      A=DEXP(A)                                                         CDTR1040
```

```
      B=2.D0/(9.D0*GG)                                                  CDTR1050
      C=(A-1.D0+B)/DSQRT(B)                                             CDTR1060
      SC=SNGL(C)                                                        CDTR1070
      CALL NDTR(SC,P,DUMMY)                                             CDTR1080
      GO TO 490                                                         CDTR1090
C                                                                       CDTR1100
C        COMPUTE THETA                                                  CDTR1110
C                                                                       CDTR1120
  190 K= IDINT(G2)                                                      CDTR1130
      THETA=G2-DFLOAT(K)                                                CDTR1140
      IF(THETA-1.D-8) 200,200,210                                       CDTR1150
  200 THETA=0.D0                                                        CDTR1160
  210 THP1=THETA+1.D0                                                   CDTR1170
C                                                                       CDTR1180
C        SELECT METHOD OF COMPUTING T1                                  CDTR1190
C                                                                       CDTR1200
      IF(THETA) 230,230,220                                             CDTR1210
  220 IF(XX-10.D0) 260,260,320                                          CDTR1220
C                                                                       CDTR1230
C        COMPUTE T1 FOR THETA EQUALS 0.0                                CDTR1240
C                                                                       CDTR1250
  230 IF(X2-1.68D02) 250,240,240                                        CDTR1260
  240 T1=1.0                                                            CDTR1270
      GO TO 400                                                         CDTR1280
  250 T11=1.D0-DEXP(-X2)                                                CDTR1290
      T1=SNGL(T11)                                                      CDTR1300
      GO TO 400                                                         CDTR1310
C                                                                       CDTR1320
C        COMPUTE T1 FOR THETA GREATER THAN 0.0 AND                      CDTR1330
C        X LESS THAN OR EQUAL TO 10.0                                   CDTR1340
C                                                                       CDTR1350
  260 SER=X2*(1.D0/THP1 -X2/(THP1+1.D0))                                CDTR1360
      J=+1                                                              CDTR1370
      CC=DFLOAT(J)                                                      CDTR1380
      DO 270 IT1=3,30                                                   CDTR1390
```

```
      XI=DFLOAT(IT1)                                                    CDTR1400
      CALL DLGAM(XI,FAC,ICK)                                            CDTR1410
      TLOG= XI*DLX2-FAC-DLOG(XI+THETA)                                  CDTR1420
      TERM=DEXP(TLOG)                                                   CDTR1430
      TERM=DSIGN(TERM,CC)                                               CDTR1440
      SER=SER+TERM                                                      CDTR1450
      CC=-CC                                                            CDTR1460
      IF(DABS(TERM)-1.D-9) 280,270,270                                  CDTR1470
  270 CONTINUE                                                          CDTR1480
      GO TO 600                                                         CDTR1490
  280 IF(SER) 600,600,290                                               CDTR1500
  290 CALL DLGAM(THP1,GTH,IOK)                                          CDTR1510
      TLOG=THETA*DLX2+DLOG(SER)-GTH                                     CDTR1520
      IF(TLOG+1.68D02) 300,300,310                                      CDTR1530
  300 T1=0.0                                                            CDTR1540
      GO TO 400                                                         CDTR1550
  310 T11=DEXP(TLOG)                                                    CDTR1560
      T1=SNGL(T11)                                                      CDTR1570
      GO TO 400                                                         CDTR1580
C                                                                       CDTR1590
C        COMPUTE T1 FOR THETA GREATER THAN 0.0 AND                      CDTR1600
C        X GREATER THAN 10.0 AND LESS THAN 2000.0                       CDTR1610
C                                                                       CDTR1620
  320 A2=0.D0                                                           CDTR1630
      DO 340 I=1,25                                                     CDTR1640
      XI=DFLOAT(I)                                                      CDTR1650
      CALL DLGAM(THP1,GTH,IOK)                                          CDTR1660
      T11=-(13.D0*XX)/XI +THP1*DLOG(13.D0*XX/XI) -GTH-DLOG(XI)          CDTR1670
      IF(T11+1.68D02) 340,340,330                                       CDTR1680
  330 T11=DEXP(T11)                                                     CDTR1690
      A2=A2+T11                                                         CDTR1700
  340 CONTINUE                                                          CDTR1710
      A=1.01282051+THETA/156.D0-XX/312.D0                               CDTR1720
      B=DABS(A)                                                         CDTR1730
      C= -X2+THP1*DLX2+DLOG(B)-GTH-3.951243718581427                    CDTR1740
```

```
      IF(C+1.68D02) 370,370,350                                         CDTR1750
  350 IF (A) 360,370,380                                                CDTR1760
  360 C=-DEXP(C)                                                        CDTR1770
      GO TO 390                                                         CDTR1780
  370 C=0.D0                                                            CDTR1790
      GO TO 390                                                         CDTR1800
  380 C=DEXP(C)                                                         CDTR1810
  390 C=A2+C                                                            CDTR1820
      T11=1.D0-C                                                        CDTR1830
      T1=SNGL(T11)                                                      CDTR1840
C                                                                       CDTR1850
C        SELECT PROPER EXPRESSION FOR P                                 CDTR1860
C                                                                       CDTR1870
  400 IF(G-2.) 420,410,410                                              CDTR1880
  410 IF(G-4.) 450,460,460                                              CDTR1890
C                                                                       CDTR1900
C        COMPUTE P FOR G GREATER THAN ZERO AND LESS THAN 2.0            CDTR1910
C                                                                       CDTR1920
  420 CALL DLGAM(THP1,GTH,ICK)                                          CDTR1930
      DT2=THETA*DLXX-X2-THP1*.6931471805599453 -GTH                     CDTR1940
      IF(DT2+1.68D02) 430,430,440                                       CDTR1950
  430 P=T1                                                              CDTR1960
      GO TO 490                                                         CDTR1970
  440 DT2=DEXP(DT2)                                                     CDTR1980
      T2=SNGL(DT2)                                                      CDTR1990
      P=T1+T2+T2                                                        CDTR2000
      GO TO 490                                                         CDTR2010
C                                                                       CDTR2020
C        COMPUTE P FOR G GREATER THAN OR EQUAL TO 2.0                   CDTR2030
C        AND LESS THAN 4.0                                              CDTR2040
C                                                                       CDTR2050
  450 P=T1                                                              CDTR2060
      GO TO 490                                                         CDTR2070
C                                                                       CDTR2080
C        COMPUTE P FOR G GREATER THAN OR EQUAL TO 4.0                   CDTR2090
```

```
C          AND LESS THAN OR EQUAL TO 1000.0
C
  460 DT3=0.D0
      DO 480 I3=2,K
      THPI=DFLOAT(I3)+THETA
      CALL DLGAM(THPI,GTH,IOK)
      DLT3=THPI*DLX2-DLXX-X2-GTH
      IF(DLT3+1.68D02) 480,480,470
  470 DT3=DT3+DEXP(DLT3)
  480 CONTINUE
      T3=SNGL(DT3)
      P=T1-T3-T3
C
C          SET ERROR INDICATOR
C
  490 IF(P) 500,520,520
  500 IF(ABS(P)-1.E-7) 510,510,600
  510 P=0.0
      GO TO 610
  520 IF(1.-P) 530,550,550
  530 IF(ABS(1.-P)-1.E-7) 540,540,600
  540 P=1.0
      GO TO 610
  550 IF(P-1.E-8) 560,560,570
  560 P=0.0
      GO TO 610
  570 IF((1.0-P)-1.E-8) 580,580,610
  580 P=1.0
      GO TO 610
  590 IER=-1
      D=-1.E75
      P=-1.E75
      GO TO 620
  600 IER=+1
      P= 1.E75
```

```
      GO TO 620                                                         CDTR2450
  610 IER=0                                                             CDTR2460
  620 RETURN                                                            CDTR2470
      END                                                               CDTR2480
```

PROGRAM(3)

```
C
C     ..................................................................
C
C        SUBROUTINE KOLMO
C
C        PURPOSE
C           TESTS THE DIFFERENCE BETWEEN EMPIRICAL AND THEORETICAL
C           DISTRIBUTIONS  USING THE KOLMOGOROV-SMIRNOV TEST
C
C        USAGE
C        CALL KOLMO(X,Z,PROB,IER)
C
C        DESCRIPTION OF PARAMETERS
C           X    - INPUT VECTOR OF N INDEPENDENT OBSERVATIONS.  ON
C                  RETURN FROM KOLMO, X HAS BEEN SORTED INTO A
C                  MONOTONIC NON-DECREASING SEQUENCE.
C           Z    - OUTPUT VARIABLE CONTAINING THE GREATEST VALUE WITH
C                  RESPECT TO X OF  SQRT(N)*ABS(FN(X)-F(X)) WHERE
C                  F(X) IS A  THEORETICAL DISTRIBUTION FUNCTION AND
C                  FN(X) AN EMPIRICAL DISTRIBUTION FUNCTION.
C           PROB - OUTPUT VARIABLE CONTAINING THE PROBABILITY OF
C                  THE STATISTIC BEING GREATER THAN OR EQUAL TO Z IF
C                  THE HYPOTHESIS THAT X IS FROM THE DENSITY UNDER
C                  CONSIDERATION IS TRUE.  E.G., PROB = 0.05 IMPLIES
C                  THAT ONE CAN REJECT THE NULL HYPOTHESIS THAT THE SET
C                  X IS FROM THE DENSITY UNDER CONSIDERATION WITH 5 PER
C                  CENT PROBABILITY OF BEING INCORRECT.  PROB = 1. -
C                  SMIRN(Z).
C           IER  - ERROR INDICATOR WHICH IS NON-ZERO IF S VIOLATES ABOVE
C                  CONVENTIONS.  ON RETURN NO TEST HAS BEEN MADE, AND X
C                  AND Y HAVE BEEN SORTED INTO MONOTONIC NON-DECREASING
C                  SEQUENCES.  IER IS SET TO ZERO ON ENTRY TO KOLMO.
C                  IER IS CURRENTLY SET TO ONE IF THE USER-SUPPLIED PDF
C                  IS REQUESTED FOR TESTING.  THIS SHOULD BE CHANGED
```

```
C              (SEE REMARKS) WHEN SOME PDF IS SUPPLIED BY THE USER.
C
C       EXTRA PARAMETERS USED IN KOLMO.
C          N     - NUMBER OF OBSERVATIONS IN X
C          IFCOD- A CODE DENOTING THE PARTICULAR THEORETICAL
C                 PROBABILITY DISTRIBUTION FUNCTION BEING CONSIDERED.
C                 = 1---F(X) IS THE NORMAL PDF.
C                 = 2---F(X) IS THE EXPONENTIAL PDF.
C                 = 3---F(X) IS THE CAUCHY PDF.
C                 = 4---F(X) IS THE UNIFORM PDF.
C                 =5 -- SIN IS ASSIGNED A VALUE,
C          SIN=1.0 -- F(X) IS THE MAXIMUM OF U UNIFORM NUMBERS
C          SIN=2.0 -- F(X) IS THE MINIMUM OF U UNIFORM NUMBERS
C          SIN=3.0 -- F(X) IS THE SUM OF 2 UNIFORM NUMBERS.
C          SIN=4.0 -- F(X) IS THE SUM OF 3 UNIFORM NUMBERS.
C          SIN=5.0 -- F-X) IS THE SUM OF 4 UNIFORM NUMBERS.
C          SIN=6.0 -- F-X) IS THE SUM OF 5 UNIFORM NUMBERS.
C          SIN=7.0 -- F-X) IS THE SUM OF 6 UNIFORM NUMBERS.
C          SIN=8.0 -- F-X) IS USER SUPPLIED.
C          U     - WHEN IFCOD IS 1 OR 2, U IS THE MEAN OF THE DENSITY
C                 GIVEN ABOVE.
C                 WHEN IFCOD IS 3, U IS THE MEDIAN OF THE CAUCHY
C                 DENSITY.
C                 WHEN IFCOD IS 4, U IS THE LEFT ENDPOINT OF THE
C                 UNIFORM DENSITY.
C                 WHEN IFCOD IS 5 AND SIN=1.0, U IS THE TOTAL OF
C                 NUMBERS CONSIDERED AT EACH TRAIL FOR THE MAXIMUM
C                 TEST.
C                 WHEN IFCOD IS 5 AND SIN=2.0, U IS THE TOTAL OF
C                 NUMBERS CONSIDERED AT EACH TRAIL FOR THE MINIMUM
C                 TEST.
C          S     - WHEN IFCOD IS 1 OR 2, S IS THE STANDARD DEVIATION OF
C                 DENSITY GIVEN ABOVE, AND SHOULD BE POSITIVE.
C                 WHEN IFCOD IS 3, U - S SPECIFIES THE FIRST QUARTILE
C                 OF THE CAUCHY DENSITY.  S SHOULD BE NON-ZERO.
```

```
C                IF IFCCD IS 4, S IS THE RIGHT ENDPOINT OF THE UNIFORM
C                DENSITY.  S SHOULD BE GREATER THAN U.
C                IF IFCOD IS 5 AND SIN=8.0, S IS USER SPECIFIED
C
C
C        REMARKS
C           N SHOULD BE GREATER THAN OR EQUAL TO 100.  (SEE THE
C           MATHEMATICAL DESCRIPTION GIVEN FOR THE PROGRAM SMIRN,
C           CONCERNING ASYMPTOTIC FORMULAE)  ALSO, PROBABILITY LEVELS
C           DETERMINED BY THIS PROGRAM WILL NOT BE CORRECT IF THE
C           SAME SAMPLES ARE USED TO ESTIMATE PARAMETERS FOR THE
C           CONTINUOUS DISTRIBUTIONS WHICH ARE USED IN THIS TEST.
C           (SEE THE MATHEMATICAL DESCRIPTION FOR THIS PROGRAM)
C           F(X) SHOULD BE A CONTINUOUS FUNCTION.
C           ANY USER SUPPLIED CUMULATIVE PROBABILITY DISTRIBUTION
C           FUNCTION SHOULD BE CODED BEGINNING WITH STATEMENT 26 BELOW,
C           AND SHOULD RETURN TO STATEMENT 27.
C
C           DOUBLE PRECISION USAGE---IT IS DOUBTFUL THAT THE USER WILL
C           WISH TO PERFORM THIS TEST USING DOUBLE PRECISION ACCURACY.
C           IF ONE WISHES TO COMMUNICATE WITH KOLMO IN A DOUBLE
C           PRECISION PROGRAM, HE SHOULD CALL THE FORTRAN SUPPLIED
C           PROGRAM SNGL(X) PRIOR TO CALLING KOLMO, AND CALL THE
C           FORTRAN SUPPLIED PROGRAM DBLE(X) AFTER EXITING FROM KOLMO.
C           (NOTE THAT SUBROUTINE SMIRN DOES HAVE DOUBLE PRECISION
C           CAPABILITY AS SUPPLIED BY THIS PACKAGE.)
C
C
C        SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
C           SMIRN, NDTR, AND ANY USER SUPPLIED SUBROUTINES REQUIRED.
C
C        METHOD
C           FOR REFERENCE, SEE (1) W. FELLER--ON THE KOLMOGOROV-SMIRNOV
C           LIMIT THEOREMS FOR EMPIRICAL DISTRIBUTIONS--
C           ANNALS OF MATH. STAT., 19, 1948.  177-189,
```

```
C         (2) N. SMIRNOV--TABLE FOR ESTIMATING THE GOODNESS OF FIT
C         OF EMPIRICAL DISTRIBUTIONS--ANNALS OF MATH. STAT., 19,
C         1948.  279-281.
C         (3) R. VON MISES--MATHEMATICAL THEORY OF PROBABILITY AND
C         STATISTICS--ACADEMIC PRESS, NEW YORK, 1964.  490-493,
C         (4) B.V. GNEDENKO--THE THEORY OF PROBABILITY--CHELSEA
C         PUBLISHING COMPANY, NEW YORK, 1962.  384-401.
C
C     ...............................................................
C
      SUBROUTINE KOLMO(X,Z,PROB,IER)
      INTEGER FI,FMT2
      COMMON FI(5),AA(9),NUM,FMT2(8)
      EQUIVALENCE (N,FI(5)), (SIN,AA(5)),(IFCOD ,FI(4)),(U,AA(6)),(S,AA(
     17))
      DIMENSION X(1)
C
C        NON DECREASING ORDERING OF X(I)'S  (DUBY METHOD)
C
      IER=0
      DO 5 I=2,N
      IF(X(I)-X(I-1))1,5,5
    1 TEMP=X(I)
      IM=I-1
      DO 3 J=1,IM
      L=I-J
      IF(TEMP-X(L))2,4,4
    2 X(L+1)=X(L)
    3 CONTINUE
      X(1)=TEMP
      GO TO 5
    4 X(L+1)=TEMP
    5 CONTINUE
C
C         COMPUTES MAXIMUM DEVIATION DN IN ABSOLUTE VALUE BETWEEN
```

```
C             EMPIRICAL AND THECRETICAL DISTRIBUTIONS
C
      NM1=N-1
      XN=N
      DN=0.0
      FS=0.0
      IL=1
    6 DO 7  I=IL,NM1
      J=I
      IF(X(J)-X(J+1))9,7,9
    7 CONTINUE
    8 J=N
    9 IL=J+1
      FJ=FS
      FS=FLOAT(J)/XN
      IF(IFCOD-2)10,13,17
   10 IF(S)11,11,12
   11 IER=1
      GO TO 29
   12 Z =(X(J)-U)/S
      CALL NDTR(Z,Y,D)
      GO TO 27
   13 IF(S)11,11,14
   14 Z=(X(J)-U)/S+1.0
      IF(Z)15,15,16
   15 Y=0.0
      GO TO 27
   16 Y=1.-EXP(-Z)
      GO TO 27
   17 IF(IFCOD-4)18,20,26
   18 IF(S)19,11,19
   19 Y=ATAN((X(J)-U)/S)*0.3183099+0.5
      GO TO 27
   20 IF(S-U)11,11,21
   21 IF(X(J)-U)22,22,23
```

```
   22 Y=C.0
      GO TO 27
   23 IF(X(J)-S)25,25,24
   24 Y=1.0
      GO TO 27
   25 Y=(X(J)-U)/(S-U)
      GO TO 27
   26 ISIN=SIN
      GO TO(30,32,36,38,42,44,46,48),ISIN
      IER=1
      GO TO 50
   30 CALL AP261(X(J),Y,IER)
      GO TO 50
   32 CALL AP262(X(J),Y,IER)
      GO TO 50
   36 CALL AP263(X(J),Y,IER)
      GO TO 50
   38 CALL AP264(X(J),Y,IER)
      GO TO 50
   42 CALL AP265(X(J),Y,IER)
      GO TO 50
   44 CALL AP266(X(J),Y,IER)
      GO TO 50
   46 CALL AP267(X(J),Y,IER)
      GO TO 50
   48 IER=1
      CALL USER(X(J),Y,IER)
   50 IF(IER.EQ.1)GO TO 29
   27 EI=ABS(Y-FJ)
      ES=ABS(Y-FS)
      DN=AMAX1(DN,EI,ES)
      IF(IL-N)6,8,28
C
C          COMPUTES Z=DN*SQRT(N)  AND  PROBABILITY
C
```

```
28 Z=DN*SQRT(XN)
   CALL SMIRN(Z,PROB)
   PRCB=1.0-PRCB
29 RETURN
   END
```

```
C                                 PROGRAM(4)
C                                                                      NDTR
C.......................................................................NDTR
C                                                                      NDTR
C        SUBROUTINE NDTR                                               NDTR
C                                                                      NDTR
C        PURPOSE                                                       NDTR
C           COMPUTES Y = P(X) = PROBABILITY THAT THE RANDOM VARIABLE  U,NDTR
C           DISTRIBUTED NORMALLY(0,1), IS LESS THAN OR EQUAL TO X.     NDTR
C           F(X), THE ORDINATE OF THE NORMAL DENSITY AT X, IS ALSO     NDTR
C           COMPUTED.                                                  NDTR
C                                                                      NDTR
C        USAGE                                                         NDTR
C           CALL NDTR(X,P,D)                                           NDTR
C                                                                      NDTR
C        DESCRIPTION OF PARAMETERS                                     NDTR
C           X--INPUT SCALAR FOR WHICH P(X) IS COMPUTED.                NDTR
C           P--OUTPUT PROBABILITY.                                     NDTR
C           D--OUTPUT DENSITY.                                         NDTR
C                                                                      NDTR
C        REMARKS                                                       NDTR
C           MAXIMUM ERROR IS 0.0000007.                                NDTR
C                                                                      NDTR
C        SUBROUTINES AND SUBPROGRAMS REQUIRED                          NDTR
C           NONE                                                       NDTR
C                                                                      NDTR
C        METHOD                                                        NDTR
C           BASED ON APPROXIMATIONS IN C. HASTINGS, APPROXIMATIONS FOR NDTR
C           DIGITAL COMPUTERS, PRINCETON UNIV. PRESS, PRINCETON, N.J., NDTR
C           1955.  SEE EQUATION 26.2.17, HANDBOOK OF MATHEMATICAL      NDTR
C           FUNCTIONS, ABRAMOWITZ AND STEGUN, DOVER PUBLICATIONS, INC., NDTR
C           NEW YORK.                                                  NDTR
C                                                                      NDTR
C.......................................................................NDTR
C                                                                      NDTR
```

```
C                                                                      NDTR 360
      SUBROUTINE NDTR(X,P,D)                                           NDTR 350
      AX=ABS(X)                                                        NDTR 370
      T=1.0/(1.0+.2316419*AX)                                          NDTR 380
      D=C.3989423*EXP(-X*X/2.0)                                        NDTR 390
      P = 1.0 - D*T*((((1.330274*T - 1.821256)*T + 1.781478)*T -       NDTR 400
     1  C.3565638)*T + 0.3193815)                                      NDTR 410
      IF(X)1,2,2                                                       NDTR 420
    1 P=1.0-P                                                          NDTR 430
    2 RETURN                                                           NDTR 440
      END                                                              NDTR 450
```

```
                         PROGRAM(5)

..................................................................

   SUBROUTINE DLGAM

   PURPOSE
      COMPUTES THE DOUBLE PRECISION NATURAL LOGARITHM OF THE
      GAMMA FUNCTION OF A GIVEN DOUBLE PRECISION ARGUMENT.

   USAGE
      CALL DLGAM(XX,DLNG,IER)

   DESCRIPTION OF PARAMETERS
      XX    - THE DOUBLE PRECISION ARGUMENT FOR THE LOG GAMMA
              FUNCTION.
      DLNG  - THE RESULTANT DOUBLE PRECISION LOG GAMMA FUNCTION
              VALUE.
      IER   - RESULTANT ERROR CODE WHERE
              IER= 0----NO ERROR.
              IER=-1----XX IS WITHIN 10**(-9) OF BEING ZERO OR XX
                        IS NEGATIVE.  DLNG IS SET TO -1.OD75.
              IER=+1----XX IS GREATER THAN 10**70. DLNG IS SET TO
                        +1.OD75.

   REMARKS
      NONE

   SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
      NONE

   METHOD
      THE EULER-MCLAURIN EXPANSION TO THE SEVENTH DERIVATIVE TERM
      IS USED, AS GIVEN BY M. ABRAMOWITZ AND I.A. STEGUN,
      'HANDBOOK OF MATHEMATICAL FUNCTIONS', U. S. DEPARTMENT OF
```

```
C         COMMERCE, NATIONAL BUREAU OF STANDARDS APPLIED MATHEMATICS
C         SERIES, 1966, EQUATION 6.1.41.

   ...................................................................

      SUBROUTINE DLGAM(XX,DLNG,IER)
      DOUBLE PRECISION XX,ZZ,TERM,RZ2,DLNG
      IER=0
      ZZ=XX
      IF(XX-1.D10) 2,2,1
    1 IF(XX-1.D70) 8,9,9

         SEE IF XX IS NEAR ZERO OR NEGATIVE

    2 IF(XX-1.D-9) 3,3,4
    3 IER=-1
      DLNG=-1.D75
      GO TO 10
C
         XX GREATER THAN ZERO AND LESS THAN OR EQUAL TO 1.D+10

    4 TERM=1.D0
    5 IF(ZZ-18.D0) 6,6,7
    6 TERM=TERM*ZZ
      ZZ=ZZ+1.D0
      GO TO 5
    7 RZ2=1.D0/ZZ**2
      DLNG =(ZZ-0.5D0)*DLOG(ZZ)-ZZ +0.9189385332046727 -DLOG(TERM)+
     1(1.D0/ZZ)*(.8333333333333333D-1 -(RZ2*(.2777777777777777D-2 +(RZ2*
     2(.7936507936507936D-3 -(RZ2*(.5952380952380952D-3)))))))
      GO TO 10

         XX GREATER THAN 1.D+10 AND LESS THAN 1.D+70

    8 DLNG=ZZ*(DLOG(ZZ)-1.D0)
```

```
      GO TO 10                                                          DLGA 700
C                                                                       DLGA 710
C        XX GREATER THAN OR EQUAL TO 1.D+70                             DLGA 720
C                                                                       DLGA 730
    9 IER=+1                                                            DLGA 740
      DLNG=1.D75                                                        DLGA 750
   10 RETURN                                                            DLGA 760
      END                                                               DLGA 770
```

PROGRAM(6)

```
.................................................................

   SUBROUTINE SMIRN

   PURPOSE
      COMPUTES VALUES OF THE LIMITING DISTRIBUTION FUNCTION FOR
      THE KOLMOGOROV-SMIRNOV STATISTIC.

   USAGE
      CALL SMIRN(X,Y)

   DESCRIPTION OF PARAMETERS
      X     - THE ARGUMENT OF THE SMIRN FUNCTION
      Y     - THE RESULTANT SMIRN FUNCTION VALUE

   REMARKS
      Y IS SET TO ZERO IF X IS NOT GREATER THAN 0.27, AND IS SET
      TO ONE IF X IS NOT LESS THAN 3.1.  ACCURACY TESTS WERE MADE
      REFERRING TO THE TABLE GIVEN IN THE REFERENCE BELOW.
      TWO ARGUMENTS, X= 0.62, AND X = 1.87 GAVE RESULTS WHICH
      DIFFER FROM THE SMIRNOV TABLES BY 2.9 AND 1.9 IN THE 5TH
      DECIMAL PLACE.  ALL OTHER RESULTS SHOWED SMALLER ERRORS,
      AND ERROR SPECIFICATIONS ARE GIVEN IN THE ACCURACY TABLES
      IN THIS MANUAL.  IN DOUBLE PRECISION MODE, THESE SAME
      ARGUMENTS RESULTED IN DIFFERENCES FROM TABLED VALUES BY 3
      AND 2 IN THE 5TH  DECIMAL PLACE.  IT IS NOTED IN
      LINDGREN (REFERENCE BELOW) THAT FOR HIGH SIGNIFICANCE LEVELS
      (SAY, .01 AND .05) ASYMPTOTIC FORMULAS GIVE VALUES WHICH ARE
      TOO HIGH ( BY 1.5 PER CENT WHEN N = 80).  THAT IS, AT HIGH
      SIGNIFICANCE LEVELS, THE HYPOTHESIS OF NO DIFFERENCE WILL BE
      REJECTED TOO SELDOM USING ASYMPTOTIC FORMULAS.

   SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED
```

```
C        NONE                                                           SMIR 350
C                                                                       SMIR 360
C     METHOD                                                            SMIR 370
C        THE METHOD IS DESCRIBED BY W. FELLER-ON THE KOLMOGOROV-        SMIR 380
C        SMIRNOV LIMIT THEOREMS FOR EMPIRICAL DISTRIBUTIONS- ANNALS     SMIR 390
C        OF MATH. STAT., 19, 1948, 177-189, BY N. SMIRNOV--TABLE        SMIR 400
C        FOR ESTIMATING THE GOODNESS OF FIT OF EMPIRICAL                SMIR 410
C        DISTRIBUTIONS- ANNALS OF MATH. STAT., 19, 1948, 279-281,       SMIR 420
C        AND GIVEN IN LINDGREN, STATISTICAL THEORY, THE MACMILLAN       SMIR 430
C        COMPANY, N. Y., 1962.                                          SMIR 440
C                                                                       SMIR 450
C     ..................................................................SMIR 460
C                                                                       SMIR 470
C     DOUBLE PRECISION X,Q1,Q2,Q4,Q8,Y                                  SMIR 490
C                                                                       SMIR 500
C        IF A DOUBLE PRECISION VERSION OF THIS ROUTINE IS DESIRED, THE CSMIR 510
C        IN COLUMN ONE OF THE DOUBLE PRECISION CARD ABOVE SHOULD BE     SMIR 520
C        REMOVED, AND THE C IN COLUMN ONE OF THE STATEMENTS NUMBERED    SMIR 530
C        C   3, C   5, AND C   8 SHOULD BE REMOVED, AND THESE CARDS     SMIR 540
C        SHOULD REPLACE THE STATEMENTS NUMBERED 3, 5, AND 8,            SMIR 550
C        RESPECTIVELY.  ALL ROUTINES CALLED BY THIS ROUTINE MUST ALSO   SMIR 560
C        PROVIDE DOUBLE PRECISION ARGUMENTS TO THIS ROUTINE.            SMIR 570
C                                                                       SMIR 580
C     ..................................................................SMIR 590
C                                                                       SMIR 600
      SUBROUTINE SMIRN(X,Y)                                             SMIR 480
      IF(X-.27)1,1,2                                                    SMIR 610
    1 Y=0.0                                                             SMIR 620
      GO TO 9                                                           SMIR 630
    2 IF(X-1.0)3,6,6                                                    SMIR 640
    3 Q1=EXP(-1.233701/X**2)                                            SMIR 650
C   3 Q1=DEXP(-1.2337005501361700/X**2)                                 SMIR 660
      Q2=Q1*Q1                                                          SMIR 670
      Q4=Q2*Q2                                                          SMIR 680
      Q8=Q4*Q4                                                          SMIR 690
```

```
      IF(Q8-1.0E-25)4,5,5                                          SMIR 700
    4 Q8=0.0                                                       SMIR 710
    5 Y=(2.506628/X)*Q1*(1.0+Q8*(1.0+Q8*Q8))                       SMIR 720
    5 Y=(2.5066282746310001/X)*Q1*(1.0D0+Q8*(1.0D0+Q8*Q8))         SMIR 730
      GO TO 9                                                      SMIR 740
    6 IF(X-3.1)8,7,7                                               SMIR 750
    7 Y=1.0                                                        SMIR 760
      GO TO 9                                                      SMIR 770
    8 Q1=EXP(-2.0*X*X)                                             SMIR 780
    8 Q1=DEXP(-2.0D0*X*X)                                          SMIR 790
      Q2=Q1*Q1                                                     SMIR 800
      Q4=Q2*Q2                                                     SMIR 810
      Q8=Q4*Q4                                                     SMIR 820
      Y=1.0-2.0*(Q1-Q4+Q8*(Q1-Q8))                                 SMIR 830
    9 RETURN                                                       SMIR 840
      END                                                          SMIR 850
```

```
C                               PROGRAM(7)
C
C.............................................................................
C
C         SUBROUTINE FREQ
C
C         PURPOSE
C         APPLIES THE FREQUENCY TEST.
C
C         USAGE
C         CALL FREQ

      SUBROUTINE FREQ
      INTEGER FI,FMT2
      COMMON FI(5),AA(9),NUM,FMT2(8),ISTART,IPRINT
      EQUIVALENCE (N,FI(5)),(M,FI(2)),(D,AA(1)),(E,AA(4))
      INTEGER Y,R
      LOGICAL*1 LINE(105)/105*'*'/
      DIMENSION COUNT(1000)
      WRITE(IPRINT,4)N,M
    4 FORMAT('0THE FREQUENCY TEST '/' THE NUMBER OF TRAILS IS',I7,T34,'R
     1EPEATED',I7,T50,'TIMES')
      WRITE(IPRINT,6)LINE
    6 FORMAT('0',105A1///)
    9 R=D
      LL=0
      DO 100 II=1,M
      DO 10 I=1,R
   10 COUNT(I)=0.0
      DO 40 I=1,N
      CALL DATA1(1,Y,LL)
      LL=1
      IF(Y.EQ.0)GO TO 35
      COUNT(Y)=COUNT(Y)+1.0
      GO TO 40
```

```
35 COUNT(R)=COUNT(R)+1.0
40 CONTINUE
   WRITE(IPRINT,43)
43 FORMAT(' THE NUMBER IN EACH CATEGORY IS')
   WRITE(IPRINT, 45)(COUNT(J),J=1,R)
45 FORMAT(10F10.2)
   P=1/D
   V1=0.0
   DO 50 K=1,R
50 V1=V1+(COUNT(K)**2)/P
   V=V1/N-N
   A=R-1
   WRITE(IPRINT,60)V,A
60 FORMAT(' THE CHI-SQUARED VALUE IS'E14.7,T42,'WITH'F6.2,T54,'DEGREE
  1S OF FREEDOM.')
   CALL CDTR(V,A,P1,D1,IER)
   P2=1-P1
   WRITE(IPRINT,85)P2,IER
85 FORMAT(' THE PROBILITY OF A WORSE VALUE OF CHI-SQUARE IS'F10.7/' T
  1HE ERROR CODE IS'I6)
   CALL APTR(P2)
   WRITE(IPRINT,90)LINE
90 FORMAT('0',105A1//)
100 CONTINUE
   RETURN
   END
```

```
C                                  PROGRAM(8)
C
C.......................................................................
C
C        SUBROUTINE SERIAL
C
C        PURPOSE
C        APPLIES THE SERIAL TEST.
C
C        USAGE
C        CALL SERIAL
C
      SUBROUTINE SERIAL
      INTEGER FI,FMT2
      COMMON FI(5),AA(9),NUM,FMT2(8),ISTART,IPRINT
      EQUIVALENCE (N,FI(5)),(M,FI(2)),(D,AA(1)),(G,AA(4))
      INTEGER X,Y,A,B
      DIMENSION A(5),COUNT(50,50)
      LOGICAL*1 LINE(105)/105*'*'/
      WRITE(IPRINT,5)N,M
    5 FORMAT('OTHE SERIAL TEST '/' THE NUMBER CF TRAILS IS'I7,T32,'REPEA
     1TED',I7,T5C,'TIMES')
      WRITE(IPRINT,1C)LINE
   1C FORMAT('0',12CA1///)
   13 J3=0
      L=1
      B=D
      II=2
      DO 1CC JJ=1,M
      DO 12 I=1,B
      DO 12 J=1,B
   12 COUNT(I,J)=C.0
      I=C
   15 CALL DATA1(II,A,J3)
      J3=1
```

```
      X=A(1)+1
      Y=A(2)+1
      COUNT(X,Y)=COUNT(X,Y)+1
      I=I+1
      IF(I.LE.N)GO TO 15
      WRITE(IPRINT,20)
   20 FORMAT(' THE NUMBER IN EACH CATEGORY IS')
      WRITE(IPRINT,25)((COUNT(I,J),I=1,B),J=1,B)
   25 FORMAT(' '15F6.1)
      P=1/(D*D)
      V1=0
      DO 30 K=1,B
      DO 30   L=1,B
   30 V1=V1+(COUNT(K,L)**2)/P
      V=V1/N-N
      C=D*D-1
      WRITE(IPRINT,40)V,C
   40 FORMAT(' THE CHI-SQUARED VALUE IS',E14.7,T42,'WITH'F6.2,T54,'DEGRE
     1ES OF FREEDOM')
      CALL CDTR(V,C,P1,D1,IER)
      P2=1-P1
      WRITE(IPRINT,50)P2,IER
   50 FORMAT(' THE PROBABILITY OF A WORSE VALUE OF CHI-SQUARED IS',F8.4,
     2/,' WITH ERROR CODE'I3)
      CALL APTR(P2)
      WRITE(IPRINT,90)LINE
   90 FORMAT('0',105A1//)
  100 CONTINUE
      RETURN
      END
```

```
C                                     PROGRAM(9)
C
C.......................................................................
C
C          SUBROUTINE POKER
C
C          PURPOSE
C          APPLIES THE POKER TEST .
C
C          USAGE
C          CALL POKER
C
      SUBROUTINE POKER
      INTEGER FI,FMT2
      COMMON FI(5),AA(9),NUM,FMT2(8),ISTART,IPRINT
      EQUIVALENCE (N,FI(5)),(LL,FI(2)),(M,FI(3)),(D,AA(1)),(E,AA(4))
      LOGICAL*1  LINE(105)/105*'*'/
      INTEGER STIRG(20),TEMP(20),A(20)
      INTEGER R,TEMP1
      DIMENSION COUNT(20)
      DIMENSION P(20)
      DIMENSION JSTIRG(20)
      WRITE(IPRINT,10)N,M,LL
   10 FORMAT('0THE POKER TEST '/' THE NUMBER OF TRIALS IS',I7,T33,'TAKEN
     1'I7,T47,'AT A TIME AND REPEATED',I7,T79,'TIMES')
      WRITE(IPRINT,12)LINE
   12 FORMAT('0',105A1///)
   15 J3=0
      DO 230 II=1,LL
      DO 16 J2=1,M
   16 COUNT(J2)=0.0
      DO 95 L=1,N
      CALL DATA1(M,A,J3)
      J3=1
      K=1
```

```
 4C I=K+1
 5C IF(A(K).LT.A(I))GC TC 70
 6C I=I+1
    IF(I.LE.M)GC TC 50
    K=K+1
    IF(K.LT.M)GC TC 40
    GO TO 80
 7C TEMP1=A(K)
    A(K)=A(I)
    A(I)=TEMP1
    GO TO 60
 8C R=1
    I=M
 9C IF(A(I-1).NE.A(I))R=R+1
    I=I-1
    IF(I.GE.2)GC TC 90
 95 COUNT(R)=CCUNT(R)+1
    V1=0.0
    WRITE(IPRINT,2C0)
2CC FORMAT(' THE NUMBER IN EACH CATEGCRY IS')
    WRITE(IPRINT,240)(COUNT(JJ),JJ=1,M)
24C FORMAT(10F8.2)
    CALL APSN(M,STIRG)
    WRITE(IPRINT,130)M,(STIRG(I),I=1,M)
13C FORMAT(' STIRGLING''S NUMBERS FOR THE VALUE',I3,T40,'ARE',/,5I6)
    DO 170 L=1,M
    FACT=1.0
    I=C
14C FACT=(D-I)*FACT/D
    I=I+1
    IF(I.LE.(L-1))GC TC 14C
17C P(L)=(FACT*STIRG(L))/(D**(M-L))
    IF(COUNT(1).GT.5.0)GC TC 172
    P(2)=P(2)+P(1)
    COUNT(2)=CCUNT(2)+CCUNT(1)
```

```
      C=M-2
      GO TO 174
  172 V1=(COUNT(1)**2)/P(1)
      C=M-1
  174 DO 175 L=2,M
      V1=V1+(COUNT(L)**2)/P(L)
  175 CONTINUE
      V=V1/N-N
      WRITE(IPRINT,220)V,C
  220 FORMAT(' THE CHI-SQUARED VALUE IS',E14.7,T42,'WITH',F6.2,T54,'DEGR
     1EES OF FREEDOM')
      CALL CDTR(V,C,P1,X1,IER)
      P2=1-P1
      WRITE(IPRINT,210)P2,IER
  210 FORMAT(' THE PROBABILITY OF A WORSE VALUE OF CHI-SQUARED IS',F8.4/
     1' WITH ERROR CODE'I3)
      CALL APTR(P2)
      WRITE(IPRINT,173)LINE
  173 FORMAT('0',105A1//)
  230 CONTINUE
      RETURN
      END
```

```
C                                PROGRAM(10)
C
C.......................................................................
C
C        SUBROUTINE GAP
C
C        PURPOSE
C        APPLIES THE GAP TEST.
C
C        USAGE
C        CALL GAP
C
      SUBROUTINE GAP
      INTEGER FI,FMT2
      COMMON FI(5),AA(9),NUM,FMT2(8),ISTART,IPRINT
      EQUIVALENCE  (A,AA(5)),(B,AA(6)),(N,FI(5)),(T,FI(4)),(M,FI(2)),(G,
     1AA(4))
      INTEGER X,T1
      INTEGER T,R
      LOGICAL*1 LINE(105)/105*'*'/
      WRITE(IPRINT,5)N,M,A,B
    5 FORMAT('0THE GAP TEST',/,' THE NUMBER OF TRIALS IS',I7,T33,'REPEAT
     1ED',I7,T50,'TIMES',/,' THE GAP IS BETWEEN'F5.2,T25,'AND',F5.2)
      WRITE(IPRINT,10)LINE
   10 FORMAT('0',105A1///)
      DIMENSION COUNT(20)
   12 J3=0
      PD=B-A
      DO 90 JJ=1,M
      J=0
      T1=T+1
      DO 15 II=1,T1
   15 COUNT(II)=0.0
   20 R=0
   25 CALL DATA2(1,U,J3)
```

```
      J3=1
      IF(U.GE.A.AND.U.LT.B)GO TO 30
      R=R+1
      GO TO 25
   30 J=J+1
      IF(R.EQ.0)GO TO 45
      IF(R.LT.T)GO TO 40
      COUNT(T)=COUNT(T)+1
      GO TO 50
   40 COUNT(R)=COUNT(R)+1
      GO TO 50
   45 COUNT(T+1)=COUNT(T+1)+1
   50 IF(J.LT.N)GO TO 20
      WRITE(IPRINT,53)
   53 FORMAT(' THE NUMBER IN EACH CATEGORY IS')
      WRITE(IPRINT,55)(COUNT(I),I=1,T1)
   55 FORMAT(8F8.2)
      V1=(COUNT(T+1)**2)/PD
      L=T-1
      DO 60 I=1,L
      P=PD*((1-PD)**I)
   60 V1=V1+(COUNT(I)**2)/P
      P=(1-PD)**T
      V1=V1+COUNT(T)**2/P
      V=V1/N-N
      C=T
      WRITE(IPRINT,70)V,C
   70 FORMAT(' THE CHI-SQUARED VALUE IS'E14.7,T42,'WITH'F5.1,T53,'DEGREE
     1S OF FREEDOM')
      CALL CDTR(V,C,P1,D1,IER)
      P2=1-P1
      WRITE(IPRINT,80)P2,IER
   80 FORMAT(' THE PROBABILITY OF A WORSE VALUE OF CHI-SQUARE IS'E14.7,T
     166,'WITH ERROR CODE'I3)
      CALL APTR(P2)
```

```
      WRITE(IPRINT,10)LINE
   90 CONTINUE
      RETURN
      END
```

```
C                                    PROGRAM(11)
C
C.......................................................................
C
C        SUBROUTINE RUNS
C
C        PURPOSE
C        APPLIES THE RUNS TEST.
C
C        USAGE
C        CALL RUNS
C
      SUBROUTINE RUNS
      INTEGER FI,FMT2
      COMMON FI(5),AA(9),NUM,FMT2(8),ISTART,IPRINT
      EQUIVALENCE (N,FI(5)),(M,FI(2)),(D,AA(1)),(G,AA(4))
      DIMENSION XCOUNT(6),B(6),A(6,6),COUNT(6)
      LOGICAL*1 LINE(105)/105*'*'/
      DATA A/4529.4,9044.9,13568.0,18091.0,22615.0,27892.0,9044.9,18097.
     10,27139.0,36187.0,45234.0,55789.0,13568.0,27139.0,40721.0,54281.0,
     267852.0,83685.0,18091.0,36187.0,54281.0,72414.0,90470.0,111580.0,2
     32615.0,45234.0,67852.0,90470.0,113262.0,139476.0,27892.0,55789.0,8
     43685.0,111580.0,139476.0,172860.0/
      DATA    B/0.1666667,0.2083333,0.09166667,0.02638889,0.005753968,0.
     1001190476/
      WRITE(IPRINT,2)N,M
    2 FORMAT('0THE RUNS TEST ',/,' THE NUMBER OF TRAILS IS',I8,T34,'REPE
     1ATED',I6,T49,'TIMES')
      IF(FI(4))3,3,5
    3 WRITE(IPRINT,4)
    4 FORMAT(' RUNS UP')
      GO TO 7
    5 WRITE(IPRINT,6)
    6 FORMAT(' RUNS DOWN')
    7 WRITE(IPRINT,8)LINE
```

```
    8 FORMAT('0',105A1///)
      J3=0
      DO 200 IJ=1,M
      DO 10 J=1,6
   10 COUNT(J)=0.0
      L=1
      CALL DATA2(1,U,J3)
      J3=1
      J=2
      L=L+1
      KK=FI(4)+1
   15 I=1
      GO TO(20,210),KK
   20 CALL DATA2(1,V,J3)
      IF(U.LT.V)GO TO 30
      U=V
      L=L+1
      I=I+1
      J=J+1
      IF(J-N)20,21,22
   21 CALL DATA2(1,V,J3)
      IF(U.LT.V)GO TO 31
      I=I+1
   22 IF(I.GE.6)GO TO 25
      COUNT(I)=COUNT(I)+1
      GO TO 100
  210 CALL DATA2(1,V,J3)
      IF(U.GT.V)GO TO  30
      U=V
      L=L+1
      I=I+1
      J=J+1
      IF(J-N)210,211,22
  211 CALL DATA2(1,V,J3)
      IF(U.GT.V)GO TO 31
```

```
      I=I+1
      GO TO 22
   25 COUNT(6)=CCUNT(6)+1
      GO TO 100
   3C IF(I.GT.5)GC TC 35
      COUNT(I)=CCUNT(I)+1
      GO TC 50
   31 IF(I.GT.5)GC TC 32
      COUNT(I)=CCUNT(I)+1
      COUNT(1)=CCUNT(1)+1
      GO TC 100
   32 COUNT(6)=CCUNT(6)+1
      COUNT(1)=CCUNT(1)+1
      GC TC 100
   35 COUNT(6)=CCUNT(6)+1
   5C U=V
      J=J+1
      L=L+1
      IF(J-N)15,36,1CC
   36 I=1
      GO TC(21,211),KK
  11C WRITE(IPRINT,120)I
  12C FORMAT(' ERRCR,NC RUN' I6)
      WRITE(IPRINT,125)
  125 FORMAT(' THE NUMBER IN EACH CATEGCRY IS')
  1CC WRITE(IPRINT,40)COUNT
   4C FORMAT(6F1C.2)
      V1=0.0
      DO 60 I=1,6
   6C XCOUNT(I)=COUNT(I)-N*B(I)
      DO 73 I=1,6
      DO 70 J=1,6
   7C V1=V1+XCOUNT(I)*XCCUNT(J)*A(I,J)
   73 CONTINUE
      FK=6.0
```

```
      W=V1/N
      WRITE(IPRINT,140)W,FK
  140 FORMAT(' THE CHI-SQUARED VALUE IS',E14.7,T42,'WITH',F6.2,T54,'DEGR
     1EES OF FREEDOM')
      CALL CDTR(W,FK,P,D1,IER)
      P1=1-P
      WRITE(IPRINT,80)P1,IER
   80 FORMAT(' THE PROBABILITY OF A WORSE VALUE OF CHI-SQUARE IS',F8.4,/
     1,' WITH ERROR CODE'I3)
      CALL APTR(P1)
      WRITE(IPRINT,8)LINE
  200 CONTINUE
      RETURN
      END
```

```
C                                PROGRAM(12)
C
C.......................................................................
C
C        SUBROUTINE DSQUR
C
C        PURPOSE
C        APPLIES THE D SQUARED TEST.
C
C        USAGE
C        CALL DSQUR
C
      SUBROUTINE DSQUR
      INTEGER FI,FMT2
      COMMON FI(5),AA(9),NUM,FMT2(8),ISTART,IPRINT
      EQUIVALENCE (N,FI(5)),(K,FI(2)),(G,AA(4))
      INTEGER R
      LOGICAL*1 LINE(105)/105*'*'/
      DIMENSION P(20),COUNT(20)
      DIMENSION U(4)
      WRITE(IPRINT,5)N,K
    5 FORMAT('OTHE D SQUARED TEST ',/,' THE NUMBER OF TRAILS IS'I7,T33,'
     1REPEATED'I7,T50,'TIMES')
      WRITE(IPRINT,10)LINE
   10 FORMAT('0'105A1///)
   12 J3=0
      DO 87M=1,K
      DO 18 I=1,20
   18 COUNT(I)=0.0
      DO 40 J=1,N
      CALL DATA2(4,U,J3)
      J3=1
      D2=(U(3)-U(1))**2+(U(4)-U(2))**2
      Z=0.1
      I=1
```

```
20 IF(D2.LT.Z)GO TO 30
   Z=Z+0.1
   I=I+1
   IF(I.LE.20)GO TO 20
   WRITE(IPRINT,25)J
25 FORMAT(' D2 TOO BIG'I7)
   GO TO 40
30 COUNT(I)=COUNT(I)+1
40 CONTINUE
   WRITE(IPRINT,43)
43 FORMAT(' THE NUMBER IN EACH CATEGORY IS')
   WRITE(IPRINT,45)COUNT
45 FORMAT(5F10.2)
   Z=0.1
   D=3.1415926
   R=1
   Q=0.0
50 P(R)=Z*D-(Z**1.5)*8.0/3.0+Z*Z/2.0-Q
   Q=Q+P(R)
   R=R+1
   Z=Z+0.1
   IF(Z.LT.1.0)GO TO 50
60 B=1/SQRT(Z)
   P(R)=1.0/3.0+(        D-2.0)*Z+4*SQRT(Z-1.0)+8*(Z-1.0)**1.5/3.0-Z*Z
  1/2.0-4*Z*ARCOS(B)-Q
   Q=Q+P(R)
   R=R+1
   Z=Z+0.1
   IF(Z.LT.2.0)GO TO 60
   P(20)=1-Q
   Q=Q+P(20)
   V1=0.0
   DO 70 R=1,20
70 V1=V1+(COUNT(R)**2)/P(R)
   V=V1/N-N
```

```
      X=19.0
      WRITE(IPRINT,75)V,X
   75 FORMAT(' THE CHI-SQUARE VALUE IS'E14.7,T42,'WITH',F5.1,T53,'DEGREE
     1S OF FREEDOM')
      CALL CDTR(V,X,P1,X3,IER)
      P2=1-P1
      WRITE(IPRINT,85)P2,IER
   85 FORMAT(' THE PROBABILITY OF A WORSE VALUE OF CHI-SQUARE IS',E14.7,
     1T66,'WITH ERROR CODE'I3)
      CALL APTR(P2)
      WRITE(IPRINT,10)LINE
   87 CONTINUE
      RETURN
      END
```

PRCGRAM(13)

```
C
C
C.....................................................................
C
C         SUBROUTINE GAPRD
C
C         PURPOSE
C         APPLIES THE GAP TEST FCR RANDCM DIGITS.
C
C         USAGE
C         CALL GAFRD

      SUBROUTINE GAPRD
      INTEGER FI,FMT2
      COMMON FI(5),AA(9),NUM,FMT2(8),ISTART,IPRINT
      EQUIVALENCE (N,FI(5)),(D,AA(1)),(SL,AA(5)),(G,AA(4)),(P,AA(6)),(P2
     1,AA(7))
      INTEGER COLNT(1000),LEN(1000)
      REAL AVER(1CCO),VAR(1CCO)
      INTEGER X
      LOGICAL*1 LINE(105)/105*'*'/
      IF(SL.EQ.0.C)SL=0.05
      WRITE(IPRINT,5)N,SL
    5 FORMAT('OTHE GAP TEST  FCR RANDOM DIGITS',/,' THE NUMBER OF TRAILS
     1 IS',I8,T34,'THE TEST IS AT',F6.4,T56,'LEVEL OF SIGNIFICANCE')
      WRITE(IPRINT,10)LINE
   1C FORMAT('C',1C5A1///)
      ID=D
      IF(P.EQ.0.C)P=1.96
      IF(P2.EQ.0.0)P2=1.645
      DO 12 KK=1,ID
      COUNT(KK)=C
      LEN(KK)=0
      AVER(KK)=0.C
   12 VAR(KK)=0.C
```

```
14 J3=0
   K=0
   DO 30 JJ=1,N
   CALL DATA1(1,X,J3)
   X=X+1
   K=K+1
   J3=1
   AVER(X)=(K-COUNT(X))+AVER(X)
   VAR(X)=(K-COUNT(X))**2+VAR(X)
   LEN(X)=LEN(X)+1
30 COUNT(X)=K
   WRITE(IPRINT,35)(AVER(I),I=1,ID),(VAR(J),J=1,ID)
35 FORMAT(5F10.1)
   WRITE(IPRINT,40)(LEN(I),I=1,ID)
40 FORMAT(5I7)
C  TO TEST FOR SIGNIFICANCE OF THE MEAN AND THE VARIANCE.
   DO 130 JJ=1,ID
   IF(LEN(JJ).EQ.0)GO TO86
   S=D*(D-1)/LEN(JJ)
   Y1=D+P*SQRT(S)
   Y2=D-P*SQRT(S)
   AVER(JJ)=AVER(JJ)/LEN(JJ)
   VAR(JJ)=VAR(JJ)/LEN(JJ)-AVER(JJ)**2
   IF(AVER(JJ).GT.Y1.OR.AVER(JJ).LT.Y2)GO TO 80
   WRITE(IPRINT,70)JJ,AVER(JJ),SL
70 FORMAT(' FOR DIGIT',I3,T15,'THE MEAN IS',F6.2,T34,'WHICH IS NOT SI
  1GNIFICANTLY DIFFERENT FROM THE EXPECTED MEAN',/,' AT',E14.7,T19,'L
  2EVEL OF SIGNIFICANCE.')
   GO TO 90
86 WRITE(IPRINT,87)JJ
87 FORMAT(' THERE IS NO OCCURRENCE OF DIGIT 'I5,/,' AND THUS THE MEAN
  1 AND THE VARIANCE FAIL THE TEST OF SIGNIFICANCE')
   GO TO 122
80 WRITE(IPRINT,85)JJ,AVER(JJ)
85 FORMAT(' FOR DIGIT',I5,' THE MEAN FAILS THE TEST OF SIGNIFICANCE W
```

```
  1ITH VALUE'E14.7)
 90 A=2*(LEN(JJ)-1)
    Z1=(P2*SQRT(A)+LEN(JJ)-1)*D*(D-1)
    IF(LEN(JJ)*VAR(JJ).GT.Z1)GO TO 110
    WRITE(IPRINT,100)VAR(JJ)
100 FORMAT(' THE VARIANCE IS',F7.3,T25,'WHICH IS NOT SIGNIFICANT')
    GO TO 122
110 WRITE(IPRINT,120)VAR(JJ)
120 FORMAT(' THE VARIANCE FAILS THE TEST WITH VALUE'E14.7)
122 WRITE(IPRINT,10)LINE
130 CONTINUE
    RETURN
    END
```

```
C                               PRCGRAM(14)
C
C.......................................................................
C
C        SUBROUTINE MAX
C
C        PURPOSE
C        APPLIES THE MAXIMUM TEST.
C
C        USAGE
C        CALL MAX
C
      SUBROUTINE MAX
      INTEGER FI,FMT2
      COMMON FI(5),AA(9),NUM,FMT2(8),ISTART,IPRINT
      EQUIVALENCE (N,FI(5)),(M,FI(2)),(T,FI(3))
      INTEGER T
      LOGICAL*1 LINE(105)/105*'*'/
      DIMENSION V(5000)
      J=T-1
      WRITE(IPRINT,5)N,M,T
    5 FORMAT('0THE MAXIMUN TEST ',/,' THE NUMBER OF TRAILS IS',I7,T33,'R
     1EPEATED'I7,T50,'TIMES.THE NUMBER IN EACH TRAIL IS'I7)
      WRITE(6,8)LINE
    8 FORMAT('0',105A1///)
   11 J3=0
      DO 60 II=1,M
      DO 10 I=1,N
      CALL DATA2(1,V(I),J3)
      J3=1
      DO 10 K=1,J
      CALL DATA2(1,U,J3)
   10 IF(V(I).LT.U)V(I)=U
      CALL KOLMO(V,Z,PROB,IER)
      WRITE(IPRINT,30)Z,PROB,IER
```

```
30 FORMAT(' THE PROBABILITY OF THE STATISTIC BEING GREATER THAN OR EQ
  1UAL TO' E14.7,/,' IF THE HYPOTHESIS IS TRUE IS'E14.7,T45,'WITH ERR
  2OR CODE'I3)
   CALL APTR(PROB)
   WRITE(IPRINT,8)LINE
60 CONTINUE
   RETURN
   END
```

PROGRAM(15)

```
C
C.......................................................................
C
C         SUBROUTINE MIN
C
C         PURPOSE
C         APPLIES THE MINIMUM TEST.
C
C         USAGE
C         CALL MIN
C
      SUBROUTINE MIN
      INTEGER FI,FMT2
      COMMON FI(5),AA(9),NUM,FMT2(8),ISTART,IPRINT
      EQUIVALENCE (N,FI(5)),(M,FI(2)),(T,FI(3))
      INTEGER T
      LOGICAL*1 LINE(105)/105*'*'/
      DIMENSION V(5000)
      J=T-1
      WRITE(IPRINT,5)N,M,T
    5 FORMAT('OTHE MINIMUM TEST ',/,' THE NUMBER OF TRAILS IS',I7,T33,'R
     1EPEATED'I7,T50,'TIMES.THE NUMBER IN EACH TRAIL IS'I7)
      WRITE(IPRINT,8)LINE
    8 FORMAT('0',105A1///)
   11 J3=0
      DO 60 II=1,M
      DO 10 I=1,N
      CALL DATA2(1,V(I),J3)
      J3=1
      DO 10 K=1,J
      CALL DATA2(1,U,J3)
   10 IF(V(I).GT.U)V(I)=U
      CALL KOLMO(V,Z,PROB,IER)
      WRITE(IPRINT,30)Z,PROB,IER
```

```
30 FORMAT(' THE PROBABILITY OF THE STATISTIC BEING GREATER THAN OR EQ
  1UAL TO' E14.7,/,' IF THE HYPOTHESIS IS TRUE IS'E14.7,T45,'WITH ERR
  2OR CODE'I3)
   CALL APTR(PROB)
   WRITE(IPRINT,8)LINE
60 CONTINUE
   RETURN
   END
```

```
C                              PRCGRAM(16)
C
C...........................................................................
C
C        SUBROUTINE SUM
C
C        PURPOSE
C        APPLIES THE SUM TEST.
C
C        USAGE
C        CALL SUM
C
      SUBROUTINE SUM
      INTEGER FI,FMT2
      COMMON FI(5),AA(9),NUM,FMT2(8),ISTART,IPRINT
      EQUIVALENCE (N,FI(5)),(M,FI(2)),(JJ,FI(3))
      LOGICAL*1 LINE(105)/105*'*'/
      DIMENSION V(5C00)
      WRITE(IPRINT,5)N,M,JJ
    5 FORMAT('OTHE SUM TEST ',/,' THE NUMBER OF TRAILS IS',I7,T33,'REPEA
     1TED',I7,T5C,'TIMES.THE NUMBER IN EACH TRAIL IS',I7)
      WRITE(IPRINT,10)LINE
   1C FORMAT('C',105A1///)
   11 J3=0
      DO 5C L=1,M
      K=JJ-1
      DO 30 J=1,N
      CALL DATA2(1,V(J),J3)
      J3=1
      DO 20 I=1,K
      CALL DATA2(1,U,J3)
   2C V(J)=V(J)+U
   3C CONTINUE
      CALL KOLMO(V,Z,PROB,IER)
      WRITE(IPRINT,40)Z,PRCB,IER
```

```
40 FORMAT(' THE PROBABILITY OF THE STATISTIC BEING GREATER THAN OR EQ
  1UAL TO',E14.7,/,' IF THE HYPOTHESIS IS TRUE IS',E14.7,T45,'WITH ER
  2ROR CODE',I3)
   CALL APTR(PROB)
   WRITE(IPRINT,10)LINE
50 CONTINUE
   RETURN
   END
```

```
C                                  PRCGRAM(17)
C
C.............................................................................
C
C          SUBROUTINE AP261
C
C          PURPOSE
C          COMPUTES Y= PRCBABAILITY THAT A RANCOM VARIABLE V WITH
C          DISTRIBUTICN FUNCTICN--   (V**U)
C          IS LESS THAN OR EQUAL TC X.
C
C          USAGE
C          CALL AP261(X,Y,IER)
C
C          DESCRIPTICN CF PARAMETERS.
C          X-- INPUT SCALAR FCR WHICH Y IS CCMPUTED.
C          Y-- OUTPUT PRCBABILTY.
C          IER-- ERRCR CODE   THIS IS NON-ZERO IF ANY INPUT PARAMETERS
C          VIOLATE THE RULES FCR THE SUBROUTINE KOLMO. IER IS SET TO ZERO
C          ON ENTRY TC THIS SUBRCUTINE.
C
C.............................................................................
C
      SUBROUTINE AP261(X,Y,IER)
      INTEGER FI,FMT2
      COMMON FI(5),AA(9),NUM,FMT2(8)
      EQUIVALENCE (U,AA(6)),(S,AA(7))
      IER=C
      IF(X)30,30,32
   3C Y=C.0
      GO TO 34
   32 Y=X**U
   34 RETURN
      END
```

```
C                                 PROGRAM(18)
C
C.......................................................................
C
C        SUBROUTINE AP262
C
C        PURPOSE
C        COMPUTES Y= PROBABAILITY THAT A RANDOM VARIABLE V WITH
C        DISTRIBUTION FUNCTION-- (1-(1-V)**U)
C        IS LESS THAN OR EQUAL TO X.
C
C        USAGE
C        CALL AP262(X,Y,IER)
C
C        DESCRIPTION OF PARAMETERS.
C        X-- INPUT SCALAR FOR WHICH Y IS COMPUTED.
C        Y-- OUTPUT PROBABILTY.
C        IER-- ERROR CODE   THIS IS NON-ZERO IF ANY INPUT PARAMETERS
C        VIOLATE THE RULES FOR THE SUBROUTINE KOLMO. IER IS SET TO ZERO
C        ON ENTRY TO THIS SUBROUTINE.
C
C.......................................................................
C
      SUBROUTINE AP262(X,Y,IER)
      INTEGER FI,FMT2
      COMMON FI(5),AA(9),NUM,FMT2(8)
      EQUIVALENCE (U,AA(6)),(S,AA(7))
      IER=0
      IF(X)30,32,32
   30 Y=0.0
      GO TO 34
   32 Y=1-(1-X)**U
   34 RETURN
      END
```

```
C                                PROGRAM(19)
C
C......................................................................
C
C       SUBROUTINE AP263
C
C       PURPOSE
C       COMPUTES Y= PROBABAILITY THAT A RANDOM VARIABLE V WITH
C       THE FOLLOWING DISTRIBUTION --
C       V                                        0<V<1
C       V-2*(V-1)                                1<V<2
C       IS LESS THAN OR EQUAL TO X.
C
C       USAGE
C       CALL AP263(X,Y,IER)
C
C       DESCRIPTION OF PARAMETERS.
C       X-- INPUT SCALAR FOR WHICH Y IS COMPUTED.
C       Y-- OUTPUT PROBABILTY.
C       IER-- ERROR CODE   THIS IS NON-ZERO IF ANY INPUT PARAMETERS
C       VIOLATE THE RULES FOR THE SUBROUTINE KOLMO. IER IS SET TO ZERO
C       ON ENTRY TO THIS SUBROUTINE.
C
C......................................................................
C
      SUBROUTINE AP263(X,Y,IER)
      INTEGER FI,FMT2
      IF(X-1.0)30,30,32
   30 Y=(X**2)/2.0
      GO TO 34
   32 Y=1.0-((2.0-X)**2)/2.0
   34 RETURN
      END
```

```
C                                 PROGRAM(20)
C
C.......................................................................
C
C        SUBROUTINE AP264
C
C        PURPOSE
C        COMPUTES Y= PROBABAILITY THAT A RANDOM VARIABLE V WITH
C        THE FOLLOWING DISTRIBUTION --
C        (V**2)/2                                  0<V<1
C        (V**2-3*(V-1)**2)/2                       1<V<2
C        (V**2-3*(V-1)**2+3*(V-2)**2)              2<V<3
C        IS LESS THAN OR EQUAL TO X.
C
C        USAGE
C        CALL AP264(X,Y,IER)
C
C        DESCRIPTION OF PARAMETERS.
C        X-- INPUT SCALAR FOR WHICH Y IS COMPUTED.
C        Y-- OUTPUT PROBABILTY.
C        IER-- ERROR CODE   THIS IS NON-ZERO IF ANY INPUT PARAMETERS
C        VIOLATE THE RULES FOR THE SUBROUTINE KOLMO. IER IS SET TO ZERO
C        ON ENTRY TO THIS SUBROUTINE.
C
C.......................................................................
C
      SUBROUTINE AP264(X,Y,IER)
      IER=0
      IF(X-1.0)30,30,32
   30 Y=(X**3)/6.0
      GO TO 38
   32 IF(X-2.0)34,34,36
   34 Y=((X**3)/3.0-(X-1.0)**3)/2.0
      GO TO 38
   36 Y=((X**3)/3.0-(X-1.0)**3+(X-2.0)**3)/2.0
```

```
38 RETURN
   END
```

```
C                               PROGRAM(21)
C
C.......................................................................
C
C        SUBROUTINE AP265
C
C        PURPOSE
C        COMPUTES Y= PROBABAILITY THAT A RANDOM VARIABLE V WITH
C        THE FOLLOWING DISTRIBUTION --
C        (V**3)/6                                           0<V<1
C        (V**3-4*(V-1)**3)/6                                1<V<2
C        (V**3-4*(V-1)**3+6*(V-2)**3)/6                     2<V<3
C        (V**3-4*(V-1)**3+6*(V-2)**3-4*(V-3)**3)/6          3<V<4
C        IS LESS THAN OR EQUAL TC X.
C
C        USAGE
C        CALL AP265(X,Y,IER)
C
C        DESCRIPTION OF PARAMETERS.
C        X-- INPUT SCALAR FOR WHICH Y IS COMPUTED.
C        Y-- OUTPUT PROBABILTY.
C        IER-- ERROR CODE   THIS IS NON-ZERO IF ANY INPUT PARAMETERS
C        VIOLATE THE RULES FOR THE SUBROUTINE KOLMO. IER IS SET TO ZERO
C        ON ENTRY TO THIS SUBROUTINE.
C
C.......................................................................
C
      SUBROUTINE AP265(X,Y,IER)
      IER=0
      IF(X-1.0)30,30,32
   30 Y=((X**4)/24.0)
      GO TO 42
   32 IF(X-2.0)34,34,36
   34 Y=((X**4)/4.0-(X-1.0)**4)/6.0
      GO TO 42
```

```
36 IF(X-3.0)38,38,40
38 Y=((X**4)/4.0-(X-1.0)**4+1.5*(X-2.0)**4)/6.0
   GO TO 42
40 Y=((X**4)/4.0-(X-1.0)**4+1.5*(X-2.0)**4-(X-3.0)**4)/6.0
42 RETURN
   END
```

```
C                                   PROGRAM(22)
C
C.......................................................................
C
C         SUBROUTINE AP266
C
C         PURPOSE
C         COMPUTES Y= PROBABAILITY THAT A RANDOM VARIABLE V WITH
C         THE FOLLOWING DISTRIBUTION --
C         (V**4)/24                                                  0<V<1
C         (V**4-5*(V-1)**4)/24                                       1<V<2
C         (V**4-5*(V-1)**4+10*(V-2)**4)/24                           2<V<3
C         (V**4-5*(V-1)**4+10*(V-2)**4-10*(V-3)**4)/24               3<V<4
C         (V**4-5*(V-1)**4+10*(V-2)**4-10*(V-3)**4+5*(V-4)**4)/24   4<V<5
C         IS LESS THAN OR EQUAL TO X.
C
C         USAGE
C         CALL AP266(X,Y,IER)
C
C         DESCRIPTION OF PARAMETERS.
C         X-- INPUT SCALAR FOR WHICH Y IS COMPUTED.
C         Y-- OUTPUT PROBABILTY.
C         IER-- ERROR CODE   THIS IS NON-ZERO IF ANY INPUT PARAMETERS
C         VIOLATE THE RULES FOR THE SUBROUTINE KOLMO. IER IS SET TO ZERO
C         ON ENTRY TO THIS SUBROUTINE.
C
C.......................................................................
C
      SUBROUTINE AP266(X,Y,IER)
      IER=0
      IF(X-1.0)30,30,32
   30 Y=((X**5)/5.0)/24.0
      GO TO 46
   32 IF(X-2.0)34,34,36
   34 Y=((X**5)/5.0-(X-1.0)**5)/24.0
```

```
      GO TO 46
   36 IF(X-3.0)38,38,40
   38 Y=((X**5)/5.0-(X-1.0)**5+2.0*(X-2.0)**5)/24.0
      GO TO 46
   40 IF(X-4.0)42,42,44
   42 Y=((X**5)/5.0-(X-1.0)**5+2.0*(X-2.0)**5-2.0*(X-3.0)**5)/24.0
      GO TO 46
   44 Y=((X**5)/5.0-(X-1.0)**5+2.0*(X-2.0)**5-2.0*(X-3.0)**5+(X-4.0)**5)
     1/24.0
   46 RETURN
      END
```

```
C                                  PROGRAM(23)
C
C.......................................................................
C
C        SUBROUTINE AP267
C
C        PURPOSE
C        COMPUTES Y= PROBABAILITY THAT A RANDOM VARIABLE V WITH
C        THE FOLLOWING DISTRIBUTION --
C        (V**5)/120                                              0<V<1
C        (V**5-6*(V-1)**5)/120                                   1<V<2
C        (V**5-6*(V-1)**5+15*(V-2)**5)/120                       2<V<3
C        (V**5-6*(V-1)**5+15*(V-2)**5-20*(V-3)**5)/120           3<V<4
C        (V**5-6*(V-1)**5+15*(V-2)**5-20*(V-3)**5+15*(V-4)**5)
C        /120                                                    4<V<5
C        (V**5-6*(V-1)**5+15*(V-2)**5-20*(V-3)**5+15*(V-4)**5-
C        6*(V-5)**5)/120                                         5<V<6
C        IS LESS THAN OR EQUAL TO X.
C
C        USAGE
C        CALL AP267(X,Y,IER)
C
C        DESCRIPTION OF PARAMETERS.
C        X-- INPUT SCALAR FOR WHICH Y IS COMPUTED.
C        Y-- OUTPUT PROBABILTY.
C        IER-- ERROR CODE   THIS IS NON-ZERO IF ANY INPUT PARAMETERS
C        VIOLATE THE RULES FOR THE SUBROUTINE KOLMO. IER IS SET TO ZERO
C        ON ENTRY TO THIS SUBROUTINE.
C
C.......................................................................
C
      SUBROUTINE AP267(X,Y,IER)
      IER=0
      IF(X-1.0)30,30,32
   30 Y=((X**6)/6.0)/120.0
```

```
      GO TO 50
   32 IF(X-2.0)34,34,36
   34 Y=((X**6)/6.0-(X-1.0)**6)/120.0
      GO TO 50
   36 IF(X-3.0)38,38,40
   38 Y=((X**6)/6.0-(X-1.0)**6+2.5*(X-2.0)**6)/120.0
      GO TO 50
   40 IF(X-4.0)42,42,44
   42 Y=((X**6)/6.0-(X-1.0)**6+2.5*(X-2.0)**6-((X-3.0)**6)*(10.0/3.0))/1
     120.0
      GO TO 50
   44 IF(X-5.0)46,46,48
   46 Y=((X**6)/6.0-(X-1.0)**6+2.5*(X-2.0)**6-((X-3.0)**6)*(10.0/3.0)+2.
     15*(X-4.0)**6)/120.0
      GO TO 50
   48 Y=((X**6)/6.0-(X-1.0)**6+2.5*(X-2.0)**6-((X-3.0)**6)*(10.0/3.0)+2.
     15*(X-4.0)**6-(X-5.0)**6)/120.0
   50 RETURN
      END
```

```
C                          PROGRAM(24)
C
C.......................................................................
C
C       SUBROUTINE  USER
C
C       PURPOSE
C       THIS IS A DUMMY SUBROUTINE.
C
C       USAGE
C       CALL USER(X,Y,IER)
C
C       DESCRIPTION OF PARAMETERS.
C       X-- INPUT SCALAR FOR WHICH Y IS COMPUTED.
C       Y-- OUTPUT PROBABILTY.
C       IER-- ERROR CODE   THIS IS NON-ZERO IF ANY INPUT PARAMETERS
C       VIOLATE THE RULES FOR THE SUBROUTINE KOLMO. IER IS SET TO ONE
C       BEFORE ENTERING THIS SUBROUTINE.
C
C.......................................................................
C
      SUBROUTINE  USER(X,Y,IER)
      RETURN
      END
```

```
C                                  PROGRAM(25)
C
C.......................................................................
C
C        SUBROUTINE READ1
C
C        PURPOSE
C        THIS A DUMMY SUBROUTINE,CALLED FROM SUBROUTINE DATA1.
C
C        USAGE
C        CALL READ1(M,K,LL)
C
C        DESCRIPTION OF PARAMETERS.
C        M -- INPUT SCALAR. THE NUMBER OF INTEGERS REQUESTED.
C        K -- OUTPUT ARRAY OR NUMBERS
C        LL -- =0  IF FIRST DATA CALL
C              =1  OTHERWISE
C
C.......................................................................
C
      SUBROUTINE  READ1(M,K,LL)
      RETURN
      END
```

```
C                                 PROGRAM(26)
C
C.......................................................................
C
C
C        SUBROUTINE READ2
C
C        PURPOSE
C        THIS A DUMMY SUBROUTINE,CALLED FROM SUBROUTINE DATA2.
C
C        USAGE
C        CALL READ2(M,V,LL)
C
C        DESCRIPTION OF PARAMETERS.
C        M -- INPUT SCALAR. THE NUMBER OF REAL NUMBERS REQUESTED.
C        V -- OUTPUT ARRAY OR NUMBERS
C        LL -- =0  IF FIRST DATA CALL
C              =1  OTHERWISE
C
C.......................................................................
C
      SUBROUTINE  READ2(M,V,LL)
      RETURN
      END
```

```
C                                 PROGRAM(27)
C
C.......................................................................
C
C        SUBROUTINE DATA1
C
C
C        PURPOSE
C        OBTAINS M INTEGERS
C
C        USAGE
C        CALL DATA1(M,K,LL)
C
C        DESCRIPTION OF PARAMETERS.
C        M -- INPUT SCALAR. THE NUMBER OF INTEGERS REQUESTED.
C        K -- OUTPUT ARRAY OR NUMBERS
C        LL -- =0  IF FIRST DATA CALL
C              =1  OTHERWISE
C
C.......................................................................
C
      SUBROUTINE DATA1(M,K,LL)
      INTEGER FI,FMT2
      COMMON FI(5),AA(9),NUM,FMT2(8),ISTART
      COMMON IPRINT,IRED,JRED
      EQUIVALENCE (D,AA(1)),(E,AA(4)),(X,AA(8))
      DIMENSION U(82),K(1)
      IF(D.EQ.0.0)D=1.0
      IF(E.EQ.0.0)GO TO 2
      NOS=E
      GO TO 4
    2 NOS=1
    4 IF(LL.NE.0)GO TO 5
      L=1
      N=1
```

```
 5 JJ=X+1
   GO TO(6,20,35,65,90),JJ
 6 DO 15 J=1,M
   IF(N.NE.1)GO TO 10
   READ(IRED)(U(I),I=1,82)
10 K(J)=D*U(L)
   L=L+NOS
   IF(L.LE.82)GO TO 12
   L=L-82
   N=1
   GO TO 15
12 N=0
15 CONTINUE
   GO TO 100
20 DO 30 J=1,M
   IF(N.NE.1)GO TO 22
   READ(IRED)(U(I),I=1,82)
22 K(J)=U(L)
   L=L+NOS
   IF(L.LE.82)GO TO 25
   L=L-82
   N=1
   GO TO 30
25 N=0
30 CONTINUE
   GO TO 100
35 DO 60 J=1,M
   IF(N.NE.1)GO TO 50
40 READ(JRED,FMT2)(U(I),I=1,NUM)
   IF(L.LE.NUM)GO TO 50
   L=L-NUM
   GO TO 40
50 K(J)=D*U(L)
   L=L+NOS
   IF(L.LE.NUM)GO TO 55
```

```
      L=L-NUM
      N=1
      GO TO 60
   55 N=0
   60 CONTINUE
      GO TO 100
   65 DO 85 J=1,M
      IF(N.NE.1)GO TO 75
   70 READ(JRED,FMT2)(U(I),I=1,NUM)
      IF(L.LE.NUM)GO TO 75
      L=L-NUM
      GO TO 70
   75 K(J)=U(L)
      L=L+NOS
      IF(L.LE.NUM)GO TO 80
      L=L-NUM
      N=1
      GO TO 85
   80 N=0
   85 CONTINUE
      GO TO 100
   90 CALL READ1(M,K,LL)
  100 RETURN
      END
```

```
C                                  PROGRAM(28)
C
C.......................................................................
C
C
C         SUBROUTINE DATA2
C
C         PURPOSE
C         OBTAINS M REAL NUMBERS.
C
C         USAGE
C         CALL DATA2(M,V,LL)
C
C         DESCRIPTION OF PARAMETERS.
C         M -- INPUT SCALAR. THE NUMBER OF REAL NUMBERS REQUESTED.
C         V -- OUTPUT ARRAY OR NUMBERS
C         LL -- =0  IF FIRST DATA CALL
C               =1  OTHERWISE
C
C.......................................................................
C
      SUBROUTINE DATA2(M,V,LL)
      INTEGER FI,FMT2
      COMMON FI(5),AA(9),NUM,FMT2(8),ISTART
      COMMON IPRINT,IRED,JRED
      EQUIVALENCE (D,AA(1)),(E,AA(4)),(X,AA(8))
      DIMENSION U(82),V(1)
      IF(D.EQ.0.0)D=1.0
      IF(E.EQ.0.0)GO TO 2
      NOS=E
      GO TO 4
    2 NOS=1
    4 IF(LL.NE.0)GO TO 5
      L=1
      N=1
```

```
 5 JJ=X+1
   GO TO(6,20,35,65,90),JJ
 6 DO 15 J=1,M
   IF(N.NE.1)GO TO 10
   READ(IRED)(U(I),I=1,82)
10 V(J)=D*U(L)
   L=L+NOS
   IF(L.LE.82)GO TO 12
   L=L-82
   N=1
   GO TO 15
12 N=0
15 CONTINUE
   GO TO 100
20 DO 30 J=1,M
   IF(N.NE.1)GO TO 22
   READ(IRED)(U(I),I=1,82)
22 V(J)=U(L)
   L=L+NOS
   IF(L.LE.82)GO TO 25
   L=L-82
   N=1
   GO TO 30
25 N=0
30 CONTINUE
   GO TO 100
35 DO 60 J=1,M
   IF(N.NE.1)GO TO 50
40 READ(JRED,FMT2)(U(I),I=1,NUM)
   IF(L.LE.NUM)GO TO 50
   L=L-NUM
   GO TO 40
50 V(J)=D*U(L)
   L=L+NOS
   IF(L.LE.NUM)GO TO 55
```

```
      L=L-NUM
      N=1
      GO TO 60
   55 N=0
   60 CONTINUE
      GO TO 100
   65 DO 85 J=1,M
      IF(N.NE.1)GO TO 75
   70 READ(JRED,FMT2)(U(I),I=1,NUM)
      IF(L.LE.NUM)GO TO 75
      L=L-NUM
      GO TO 70
   75 V(J)=U(L)
      L=L+NOS
      IF(L.LE.NUM)GO TO 80
      L=L-NUM
      N=1
      GO TO 85
   80 N=0
   85 CONTINUE
      GO TO 100
   90 CALL READ2(M,V,LL)
  100 RETURN
      END
```

```
C                             PROGRAM(29)
C
C.......................................................................
C
C        SUBROUTINE GETFMT
C
C        PURPOSE
C        TO OBTAIN THE REQUIRED FORMAT STATEMENT.
C
C        USAGE
C        CALL GETFMT(IN,SWITCH,FMT)
C
C        DESCRIPTION OF PARAMETERS.
C        IN -- INPUT VECTOR. IT IS EITHER BLANK OR CONTAINS A FORMAT
C        STATEMENT.
C        SWITCH -- STANDARD FORMAT.
C                  =0.0 THEN  (4I4,I10,5F5.1,2F9.4,2F5.1)
C                  =1.0 THEN  (5E14.7)
C
C.......................................................................
C
      SUBROUTINE GETFMT(IN,SWITCH,FMT)
      INTEGER FI,FMT2
      COMMON FI(5),AA(9),NUM,FMT2(8)
      INTEGER*4 IN(1),FMT(1)
      INTEGER*4 B(8)/8*'    '/,STAND1(7)/'(4I4',',I10',',5F5','.1,2','F9
     1.4',',2F5','.1) '/,STAND2(2)/'(5E1','4.7)'/
      IF(SWITCH.NE.0.0)GO TO 50
      DO 10 J=1,8
      IF(IN(J).NE.B(J))GO TO 30
   10 CONTINUE
      DO 20JJ=1,7
   20 FMT(JJ)=STAND1(JJ)
      GO TO 70
   30 DO 40 J=1,8
```

```
40 FMT(J)=IN(J)
   GO TO 70
50 DO 60 J=1,8
   IF(IN(J).NE.B(J))GO TO 30
60 CONTINUE
   DO 65 JJ=1,2
65 FMT(JJ)=STAND2(JJ)
   NUM=5
70 RETURN
   END
```

```
C                             PROGRAM(30)
C
C.......................................................................
C
C        SUBROUTINE APSN
C
C     PURPOSE
C        TO CALCULATE STIRLINGS NUMBERS FOR K.
C
C        USAGE
C        CALL APSN(K,IS)
C
C        DESCRIPTION OF PARAMETERS.
C        K-- INPUT SCALAR,NUMBER OF DIGITS FOR WHICH STIRLINGS NUMBERS
C            ARE REQUIRED.
C        IS-- OUTPUT VECTOR OF STIRLINGS NUMBERS.
C
C.......................................................................
C
      SUBROUTINE  APSN(K,IS)
      DIMENSION IS(20),JTEMP(20),JS(20)
      IS(1)=1
      DO 20 J=2,20
   20 IS(J)=0
      DO 50 J=2,K
      DO 30 L=2,J
   30 JTEMP(L)=IS(L-1)+IS(L)*L
      DO 40 M=2,J
   40 IS(M)=JTEMP(M)
   50 CONTINUE
      RETURN
      END
```

```
C                                PROGRAM(31)
C
C.......................................................................
C
C     SUBROUTINE  APTR(B)
C
C     PURPOSE
C        TEST FOR DEGREE OF RANDOMNESS.
C
C     USAGE
C        CALL  APTR(B)
C                                                                       B
C     DESCRIPTION OF PARAMETERS
C        B -PROBILITY OF A WORSE VALUE OF CHI-SQUARE.
C
C.......................................................................
C
      SUBROUTINE  APTR(B)
      COMMON FI(5),AA(9),NUM,FMT2(8),ISTART,IPRINT
      IF(B.GE.0.99.OR.B.LE.0.01)GO TO 10
      IF(B .GE..95.OR.B .LE.0.05)GO TO 30
      IF(B .GE.0.90.OR.B .LE.0.10)GO TO 50
      WRITE(IPRINT,70)
   70 FORMAT(' THIS DATA IS CONSIDERED TO BE RANDOM')
      GO TO 80
   10 WRITE(IPRINT,20)
   20 FORMAT(' THIS DATA CANNOT BE CONSIDERED RANDOM')
      GO TO 80
   30 WRITE(IPRINT,40)
   40 FORMAT(' THIS DATA IS SUSPECT OF NOT BEING RANDOM')
      GO TO 80
   50 WRITE(IPRINT,60)
   60 FORMAT(' THIS DATA IS  SLIGHTY SUSPECT OF NOT BEING RANDOM')
   80 RETURN
      END
```

REFERENCES

Abramowitz, M. and Stegun, I.A. (1964) Handbook on Mathematical Functions U.S. Government Printing Office

Bofinger, E. and Bofinger, V.J. (1958) On a periodic property of psuedo-random sequences Journal Association Computing Machinery V5, p261

Conveyou, R.R. and MacPherson, R.D. (1967) Fourier Analysis of Uniform Random Number Generators Journal Association Computing Machinery V14, p100-119

Cramer, H. (1951) Mathematical Methods of Statistics Princeton University Press p244

Cress, P. and Dirksen, P and Graham, J.W. (1968) Fortran IV with Watfor Prentice-Hall, Inc.

Downham, D.Y. and Roberts, F.D.K. (1967-1968) Multiplicative congruential pseudo-random number generators The Computer Journal V10, p74-77

Feller, W. (1968) An Introduction to probability theory and its applications Wiley V1 3rd edition

Fisz, M. (1963) Probability Theory and Mathematical Statistics Wiley 3rd edition

Forsythe, A.B. (1968) Aim to demonstrate the need for careful testing for characteristics of randomness Computers and Biomedical Research V1, p470-474

Gelder, A. van (1967) Some New results in Psuedo-Random Number Generation Journal Association Computing Machinery V14, p785-792

Good, I.J. (1953) The serial test for sampling numbers and other tests for randomness Proc. Cambridge Phil. Soc. V49, p276

Gorenstari, S. (1967) Testing a random number generator Comm. ACM V10, p111-118

Greenberger, M. (1961 and 1962) A prior determination of serial correlation in computer generated random numbers Math of Computation V15, p383

Greenberger, F. and Jaffrey, G. (1965) Problems for Computer Solution Wiley

Hammersley, J.M. and Handscomb, D.C. (1964) Monte Carlo Methods Methven's Monographs

IBM SYSTEM/360 44 C28-6811-1 Programming Systems Assembler Language

IBM SYSTEM/360 44 C28-6812 Programming System Guide to System Use

IBM SYSTEM/360 44 C28-6813 Programming System Guide to System Use for FORTRAN Programmers

IBM SYSTEM/360 44 C28-6515-7 FORTRAN IV Language

IBM Random Number Generation and Testing GC20-8011-0

Jansson, B. (1964) Autocorrelation between Pseudo-Random Numbers B.I.T. V4

Jansson, B. (1966) Random Number Generators Arnquist and Wiksel

Knuth, D.E. The Art of Computer Programming (1968) V1 Fundamental Algorithms (1969) V2 Seminumerical Algorithms Addison and Wesley

Lindgren, B.W. (1960) Statistical Theory Macmillan

MacLaren, M.D. and Marsaglia, G. (1965) Uniform Random Number Generators JACM V12, p83-89

Page, E.S. (1967) Digital Simulation in Operational Research Congress - Hamburg - Scientific Affairs Division p55

Teichroew, D. (1965) Distribution Sampling Prior to the Computer J.A.S.A. V60, p36

Tocher, K.D. (1963) The Art of Simulation English U. Press

Todd, J. and Taussky, O. (1956) Generation of psuedo-random numbers Symposium on Monte Carlo Methods p15-28 Wiley (H.A. Mayer edition)

Whittlesey, J.R.B. (1968) A Comparison of the Correlational Behaviour of Random Number Generators for the IBM 360 Comm. ACM V11, No.9, p641-644

Wilson, B. (1891) Integral Calculus London p390

Yamada, S. (1960) On the period of psuedo-random numbers generated by Lehmers congruential method J. Op. Res. Soc. Japan V3, p113-128