

AD-A106 034

POLYTECHNIC INST OF NEW YORK BROOKLYN

F/S 9/2

SOFTWARE MODELING STUDIES EXPERIMENTAL STUDY OF A TWO DIMENSION--ETC(U)

JUL 81 M KLERER

F30602-78-C-0057

UNCLASSIFIED

RADC-TR-81-183-VOL-3

NL

1 OF 1
AD A
116034

END
DATE
FILMED
11-81
DTIC

AD A106034

LEVEL

13

RADC-TR-81-183, Vol III (of four)
Final Technical Report
July 1981



**SOFTWARE MODELING STUDIES
EXPERIMENTAL STUDY OF A TWO
DIMENSIONAL LANGUAGE Vs FORTRAN
FOR FIRST COURSE PROGRAMMERS**

Polytechnic Institute of New York

Melvin Klerer

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC

ELECTRONIC

OCT 22 1981

A

**ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441**

DTIC FILE COPY

10 10 19

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-81-183, Vol III (of three) has been reviewed and is approved for publication.

APPROVED:

Rocco F. Iuorno

ROCCO F. IUORNO
Project Engineer

APPROVED:

John J. Marciniak

JOHN J. MARCINIAK, Colonel, USAF
Chief, Information Sciences Division

FOR THE COMMANDER:

John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ISIE) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-81-183, Vol III (of four)	2. GOVT ACCESSION NO. AD-A206034	3. RECIPIENT'S CATALOG NUMBER 034
4. TITLE (and Subtitle) SOFTWARE MODELING STUDIES EXPERIMENTAL STUDY OF A TWO DIMENSIONAL LANGUAGE VS FORTRAN FOR FIRST-COURSE PROGRAMMERS		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report January 78 - October 80
7. AUTHOR(s) Melvin Klerer		6. PERFORMING ORG. REPORT NUMBER N/A
9. PERFORMING ORGANIZATION NAME AND ADDRESS Polytechnic Institute of New York 333 Jay Street Brooklyn NY 11201		8. CONTRACT OR GRANT NUMBER(s) F30602-78-C-0057
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (ISIE) Griffiss AFB NY 13441		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304J401
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same		12. REPORT DATE July 1981
		13. NUMBER OF PAGES 86
		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) Same		
18. SUPPLEMENTARY NOTES RADC Project Engineer: Rocco F. Iuorno (ISIE)		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Programming Language Language Performance Measurement Relative Efficiency of Programming Languages Language Performance Data Timing Experiments Klerer-May Two Dimensional Language		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The purpose of this study was to obtain quantitative measures of the relative performance of two very different programming languages. One language was Fortran and the other was the Klerer-May Two Dimensional Language. The report describes the experiments conducted, presents the experimental results and provides a discussion of the results obtained.		

DD FORM 1473

EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

101701

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

<u>Table of Contents</u>	<u>Page</u>
Introduction	1
Experiment I (Fortran Timing)	2
Purpose	2
Method	2
Results	2
Discussion	3
Experiment II (2-D vs. Fortran)	5
Purpose	5
Method	5
Results	7
Discussion	8
Acknowledgements	9
References	10
Fortran Timing Results	Appendix A
Analysis of Variance Results	Appendix B
Reference Manual	Appendix C
McCracken Problems vs. Corresponding K-M Programs	Appendix D
K-M Examples (Programs as Compiled and Executed)	Appendix E

The printing in Appendix D is intentionally incomplete per Ms. Shirley Lyons, RADC/TS

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

INTRODUCTION

The purpose of this work was to obtain some quantitative measure of relative performance for two very different programming languages. One language was FORTRAN and the other was the Klerer-May (4) (K-M) two-dimensional (2-D) language. In this 2-D language, programming of most algebraic expressions requires little or no alteration of the text book form when typed on a input/output typewriter with mathematical typing capability. The general syntax of the language was designed so as to minimize the learning period of the novice user, to minimize programming error by using ordinary mathematical notation and semantics, and to be self-documenting and easily readable by anyone with a minimum of mathematical literacy.

Some of the basic style associated with the K-M 2-D programming system is illustrated by the examples to be found in appendices D and E. Programs are input by typing on a modified I/O typewriter. Half-space subscripting and superscripting are under keyboard control and arbitrary-sized common mathematical symbols may be "drawn" by the use of eight special characters that "interlock" so that the complete symbol appears to be continuous. Corrections can be made by overtyping or by pressing a special "erase" key when positioned over the unwanted character. Mathematical symbols need not be typed neatly as the system was designed to recognize highly asymmetric representations of basic symbols and to tolerate non-uniform spacing in both horizontal and vertical directions. Arbitrary back and forward spacing, up and down spacing, intermixed with typing, is permitted within the boundaries of a single (compound) statement terminated horizontally by a period. The reference manual for the basic system is printed on two sides of a plastic 8 1/4 by 10 3/4 inch sheet which is illustrated in appendix C. The philosophy of the system is to permit the user to exercise a variety of (sometimes equivalent) syntactical forms identical to ordinary mathematical notation, to allow easy input even by awkward typists, and to minimize the amount of procedural and linguistic detail necessary for use of the system. Ambiguous input is resolved by the use of context dependent processing and, prior to full compilation and execution, by output to the user of a Fortran-like linear interpretation of his input. The user can then correct or edit his program if the system's interpretation differs from his own.

However, as has been previously noted (1,3,7), experiments to test the relative efficiency of programming languages are difficult to carry out for several reasons. Long term studies on professional programmers engaged in producing a large production program present administrative difficulties, since the interests of those responsible for producing the program (e.g. minimizing costs) are not necessarily the same as those who are interested in studying the project in ways that assure statistical validity. Also, having another group duplicate the program using a different language is nearly always not practical. Shorter studies on artificial test problems tend to produce results of dubious statistical validity. This stems from the difficulty in controlling the human factors than can affect the results of such an experiment, the small number of subjects usually available, but most importantly the tremendously large variance or range in performance from one individual to another (1,3,5). My own personal experience in directing a computing center for

many years and in managing programming efforts has led me to believe that a gifted programmer can produce checked-out code (regardless of the programming language) at a rate which appears to be 10 to 100 times faster than a programmer who is competent but of mediocre talent.

Because of these considerations it was decided to carry out some initial studies on a population consisting of students who were taking a first computing course using FORTRAN as a programming language. It could be expected that such a group would be relatively homogeneous in terms of education, work experience, and previous knowledge of computing. Also, the experimental procedures would be easier to administer if the instructor of the course agreed to cooperate and if the students were told that their participation in the experiment would be credited toward their work. However, the requirement that the experiment not interfere unduly with the normal curriculum of the course forced the use of test problems of minimal expectation effort to be assigned to that phase where the two languages were compared (Experiment II). Further, as a desirable side effect, it might be expected that the use of these very simple problems might narrow down the variance associated with natural programming ability. Also, in order to gain some feeling for the inherent variability of the results for less artificial problems, a separate study (Experiment I) was undertaken.

EXPERIMENT I (Fortran Timing)

PURPOSE

The purpose of this experiment was to gather performance (time) data for students learning FORTRAN.

METHOD

The subjects were students in a first level graduate computing course. There was no interference with the normal conduct of the course and students were asked only to supply time data for programming, debugging, keypunching, wait time, and number of debugging runs. The Fortran text was by McCracken (2) and problems were those picked by the instructor without regard to the purpose of this data sampling.

RESULTS

The detailed results of this experiment are given in Appendix A. The set of results for each assigned problem is first identified by the heading "Fortran Timing Results", followed by the problem number and page where it may be found in McCracken's book. The number of student responses for each problem is also given. The first block is the raw input data specifying programming time, keypunch time, the number of debug runs, debug time, debug keypunch time, and computer wait time, as reported by each student. Where no data was reported for any item, the code 9191 was entered at the appropriate place. The next block gives the statistical results computed from the raw data. In cases where data was not reported for either programming

time or debug time for a specific problem and student then the sum of the average programming time plus average debug time might differ from the average of total (programming plus debug) time since total programming was not defined unless both items were reported together.

In the next output block, the average, range (difference between maximum and minimum), performance ratio (maximum divided by minimum, where a line of asteriks indicates that the ratio was in excess of a meaningful value), variance, and standard deviation associated with each measured category are given.

The last output block for each problem is a distribution plot of total programming time for the set of students. In each plot the vertical line labeled M denotes the median point and the vertical line labeled A denotes the point which represents the average total programming time for the particular problem.

The actual problems are given in Appendix D. There, McCracken's problems are shown side by side with the corresponding K-M programs which are solutions to McCracken's problems. The purpose of this appendix is to illustrate how little translation is necessary to go from the problem statement stage to the actual 2-D programs. It should also be pointed out that anyone with elementary mathematical literacy should be able to understand the K-M programs without prior instruction. The only artifices that might require referral to the K-M reference manual (Appendix C) might be the DIMENSION declaration (but whose meaning would be obvious to anyone with experience in any other programming language) and the use of the "ket" brackets following a variable to enclose the number representing the field size of the integer to be printed.

DISCUSSION

The results of this experiment make clear that there is a wide variation in individual programming performance. This is consistent with previously reported results (3). For meaningful sample size, the performance ratio associated with the measure of total programming time varied from a low of 10 to a high of 50 over the set of problems. For a category such as debug time, it was difficult to assign a meaningful performance ratio since this could vary from zero to relatively large quotients. Even the performance ratio associated with keypunch time seemed to be dependent on the particular problem. This might indicate that a certain portion of what was reported as keypunch time was not just the timing of mechanical effort but might include "think time" connected with each problem.

Furthermore, the distribution of these results tend to be highly skewed with large variances. The asymmetric nature of each distribution of total programming time is indicated by the relative separation between average and median on each plot. It should be noted that each distribution was plotted on a relative scale which was a function of the maximum element in the set, i.e., the maximum element always occurs on the extreme right of the plot.

But it is indeed surprising that such wide performance variations appear for such elementary problems and in a novice population with an expectation of relative homogeneity. Previous results suggest that these wide variations are also consistent with the performance of experienced programmers (1,3,6).

An important question to be addressed is whether such results are unique to programming or are they typical of performance in other technical or professional fields? It is difficult to think of another field which both allows the formation of a metric of quantitative performance and for which there is typically a wide range of performance. The only endeavors which come to mind that are characterized by analogous or even a greater range in quantitative performance are those of invention or scientific discovery. However, it would appear that the quality of intellectual endeavor associated with invention or scientific discovery is of a much higher plane than the mundane task of programming. Or, indeed is this really so?

The problem of the great variance in performance for invention and scientific discovery was examined by Shockley (6). Even in a highly selected population sample of research workers in scientific laboratories, he found that some individuals were at least fifty times more productive than others in equivalent circumstances.

Shockley speculated that these statistics might be explained by a model of human intelligence where each individual had a capability of being able to be aware of M ideas and their relationships simultaneously. Furthermore, since a higher value of M would allow many more permutations and combinations of basic ideas, then a relatively small increment in the value of M would cause a disproportionally larger increase in the total number of permuted or combined basic ideas relevant to an invention or intellectual discovery. An alternate model, also proposed by Shockley, would link intellectual productivity to the product of independently varying different factors. If the number of factors were large, and if one individual's factors each exceed that of another individual by a modest amount, the overall product of factors will be very different between the two individuals. Shockley also gives some hints as to how one can determine the parameters of each model (e.g., the value of " M ") by studying the statistics of productivity.

For the case of programming, where one must keep in mind many considerations, Shockley's first model seems attractive. In fact, based on personal introspection, and informal discussions with other individuals as to how they function in the process of programming, it would appear that the capability of perceiving several ideas and their relationships simultaneously may be crucial to successful, efficient programming.

There are other ways of regarding these results. We could conclude that we must be more selective in training and employing programmers, since those programmers who do less well than the median exert a highly disproportionate negative effect on programming performance. But in view of the current shortage of programmers, this does not appear to be a practical alternative, even if one were to agree on an efficient selection criteria.

However, it does provide a clue as to why very large programming teams tend to be slower in producing a programming product than a highly selected tiny group. The overall performance of a group tends to be lower than its most inefficient member.

But one can treat this matter from a more disparate point of view. Put bluntly, it would appear that the results suggest that most people who do programming simply do not possess the special intellectual skills to easily program on an appropriately competent level. If one wishes to speculate within the framework of such a model, then it would appear that much of the present concern with program errors and program reliability may be missing the essence of the phenomenon. Instead of these errors being evidence of inadequate system methodology (e.g. an inadequate program structure), they may indeed point to essentially random psychological effects brought about by an inability to perform to the level of the programming task.

Regardless of the precise theoretical model to account for this wide variation in performance, it would seem that the nature of the phenomenon dictates the most efficient solution, i.e., automatic programming systems for that (large) part of programming tasks which are well formulated in some sense and where the translation from problem statement to computer code is essentially deterministic.

EXPERIMENT II (2-D vs Fortran)

PURPOSE

The purpose was to measure the comparative performance of programming novices, with some experience in FORTRAN, upon brief exposure to a 2-D language.

METHOD

Two very simple problems were chosen so as not to interfere with the usual classroom objectives. Problem #1 was:

" Print Y for values of X starting at X = 0.1 increasing in steps of 0.2 until X = 0.9 where

$$Y = \sum_{i=1}^5 iX^i "$$

and problem #2 was:

$$P = \frac{100 + 50X + 25X^2}{\sqrt{10X^3 + 2X^4}}$$

Print P for X = 1, 2, ..., 6."

The experiment was repeated for two different classes taking a graduate first course offering in computer science where FORTRAN was introduced as a programming language. At the time the students were asked to participate in the experiment, they had already had approximately 20 to 23 hours of formal classroom instruction in elementary computing using FORTRAN. Also, in the preceding 10 weeks, they had had the opportunity of solving problems using FORTRAN. The formal lecture on the 2-D language was approximately one hour long. Also, they were given a copy of the one-sheet user manual for the language and a set of 16 sample problems illustrating 2-D programs, the initial computer conversion to a linear program format, the output of the automatic translation phase into FORTRAN, data input and the output results. Students were advised that they should not spend more than two hours looking over this "take home" material before attempting the problem. Thus, in terms of formal training and practice, the students had a background favoring FORTRAN competency by a factor of at least 20 to 1. Of course, we are not unmindful that there is a transfer learning effect from one language to another, but this requires study under an experiment of different design. None-the-less, it appears unlikely that, for the case of a novice population, the transfer learning effect would be so large as to diminish significantly the large bias of the experiment toward FORTRAN competency. Put another way, any significant difference between the 2-D language and FORTRAN, if expressed as a performance ratio, should be multiplied by a "handicap" factor. This factor should have a magnitude lying somewhere between 1 and 20.

Each class was randomly divided into two equal groups. Group I was assigned problem #1 to be done in FORTRAN and Problem #2 to be done in K-M. Group II was assigned Problem #1 to be done in K-M and Problem #2 to be done in FORTRAN. The completed FORTRAN problems were required to be returned two weeks later. Since there were not sufficient terminals available for the class to input the K-M programs directly, within the given time limitations, they were asked to return their hand-written K-M programs one week later at the beginning of the class. These programs were visually inspected for correctness, and, where needed, error message output was simulated and returned to the students for further debugging. Final hand-written K-M programs were returned by the students one week later. The use of hand-written program input is not unusual with K-M systems practice. At Columbia University's Hudson Laboratories, where the K-M system was used as a production system (1) for several years, users were given the option of either typing their programs directly for online (or offline) processing or having their programs typed by the typists employed in the computing center and processed offline. Our experience at Columbia indicated that the effort and error rate involved in typing K-M programs were no greater than that involved in the equivalent typing of mathematical text using a standard office typewriter. We also concluded that the input typing error rate for the K-M program was substantially less than the error rate experienced in key-punching the equivalent FORTRAN program. Our experience, then, led us to believe that, in a practical sense, the K-M language was more suitable than FORTRAN, to a computing center environment which tried to convenience users by accepting hand written input for program compilation. However, we should note that these conclusions were based on our informal observations

and no formal experiments were made to obtain a precise measure of these comparisons.

RESULTS

Since a few of the final programs were still flawed by major or minor errors, an additional weighted data set for each class was processed to reflect these errors. The weighting was as follows: If the program was incorrect, then 50% of the programming time was added to the debug time, the sum being treated as the weighted debug time. If the program contained a minor or trivial error then 20% of the programming time was added to the debug time, the sum being treated as the weighted debug time. However, the results of these weighted sets were not substantially different from the unweighted data sets.

The results of this experiment are given in Appendix B. The block labeled as Set #1 first gives the raw data reported by the class of 20 students for K-M programming time, K-M debug time, FORTRAN programming time, and FORTRAN debug time. Data for FORTRAN keypunch time, number of debug runs, debug keypunch time, and wait time are also reported but were not processed at this time.

Following the raw data, the total programming time (programming time + debug time) is arranged as a two-by-two cellular array, where the elements of each cell list total programming time corresponding to problem number and language.

For problem 1, the mean (total) programming time ratio (R_t^{FK}) of FORTRAN vs K-M is 3.6 and for problem 2 the FORTRAN vs K-M time ratio is 2.9. The corresponding unbiased estimates of the standard deviations are given and are typically very large for each datum. As we have noted previously, these ratios should be multiplied by a "handicap" factor h , where $1 < h < 20$, to give a truer picture of the performance of one language relative to the other. Thus if we define economic efficiency (E) to be inversely proportional to total programming time, then the economic efficiency of K-M vs FORTRAN as a function of problem would be

$$E_{KF} = hR_t^{FK}.$$

An analysis of variance indicates that the difference between the two languages is significant at the $\epsilon = .05$ level, and that the difference between the two problems is not significant at the $\epsilon = .05$ level but may be considered significant at the $\epsilon = .1$ level.

The results for the weighted set 1 are not dramatically different. For problem 1, $R_t^{FK} = 3.96$ and for problem 2, $R_t^{FK} = 3.7$. The analysis of variance indicates that the difference between the two languages is significant at the $\epsilon = .05$ level and that the difference between the two problems is not significant. The main effect of the weighting was to increase the FORTRAN vs K-M programming time ratio for problem 2 and to also increase the relative variances, accounting for the lessened statistical significance of the results.

The results of set #2 are based on a much larger sample than that used in Set #1 (34 students compared to 20 in the previous sample). For problem #1, the mean (total) programming time ratio of FORTRAN vs K-M is $R_t^{FK} = 6.4$. For problem 2, $R_t^{FK} = 1.76$. In each case the economic efficiency of K-M vs FORTRAN is given by $E_{KF} = hR_t^{FK}$, $1 < h < 20$.

The analysis of variance for this set indicates that the difference between the two languages is significant at the $\epsilon = .001$ level and that the difference between the two problems is significant at the $\epsilon = .05$ level. Also, there is a non-negligible interaction between problem type and language.

For the weighted set #2, $R_t^{FK} = 7.1$ for problem 1, $R_t^{FK} = 1.76$ for problem 2. The analysis of variance indicates that the difference between the two languages is significant at the $\epsilon = .005$ level and that the difference between the two problems is significant at the $\epsilon = .05$ level.

DISCUSSION

These experiments offer clear evidence that there is a decided economic advantage for novices in using a two-dimensional approach to scientific/engineering application programming. There is reason to believe that the relative advantage of the 2-D approach becomes even greater when used in a production environment for complex application programs (1). One of the several reasons for this is that the 2-D programming approach models exactly in many cases, or very closely in the remaining cases, visually complex mathematical formula. Therefore, a certain part of the debugging task simplifies to routine proof reading of the original problem formulae contrasted to the 2-D program statements. Thus there is a marked reduction of program error for complex formulae representation due to the fact that the translation from problem to program is either identical or characterized by minimal change. The same philosophy applies to the syntax of input-output which is a major source of program error in such languages as FORTRAN. The K-M language uses free-field and type-independent input and several kinds of output forms, both linear and two-dimensional, so that checking output syntax as a function of problem specification is also reduced to a proof reading task (see Appendices C and E for some examples).

However, the process of making precise objective judgements of relative language efficiency confronts many difficult problems of experimental design and practical implementation due to the large range of individual programming capability. Judgement of precise 2-D programming efficiency is particularly difficult because of its novel programming approach, the relative unavailability of suitable input terminals, and the artificial intelligence aspects of the system design for a 2-D effective system. None-the-less, the relative economic efficiency factor of a 2-D language such as K-M when compared to a linear programming language such as FORTRAN, appears to be so large that only an order of magnitude best estimate seems to be sufficient. This best estimate is expressed above by the term E_{KF} . Certainly, further experimentation along these lines is appropriate to obtain best estimates within a narrower range.

ACKNOWLEDGEMENTS

We wish to thank Professors John Chu, John Kao, and Leon Herbach for discussing some of the statistical aspects of this problem with us. Acknowledgement is made to the National Science Foundation, who supported, in part, some of the work used in this paper under NSF Grant #MCS75-22569, and to support of the Rome Air Development Center, under contract No. F-30602-78-C-0057 (SOFTY) that allowed us to complete this work. Our thanks are also to Professors Henry Ruston and Martin Shooman for their general encouragement. Of course, any errors or deficiencies of this work are solely the responsibility of the author.

REFERENCES

- (1) M. Klerer, The economics, politics, and sociology of two-dimensional systems ACM-SIGPLAN NOTICES, vol. 7, no. 10, October 1972.
- (2) D.D. McCracken, A Guide To Fortran IV Programming, 2nd edition, John Wiley, 1972.
- (3) H. Sackman, Man-Computer Problem Solving, Auerbach, 1970.
- (4) J.E. Sammet, Programming Languages, Prentice-Hall, 1969.
- (5) B. Shneiderman, et al, Experimental investigations of the utility of detailed flowcharts in programming, CACM, vol. 20, no. 6, June 1977.
- (6) W. Shockley, On the Statistics of Individual Variations of Productivity in Research Laboratories, Proceeding of the IRE, March, 1957, pp. 279-290.
- (7) G.M. Weinberg, The Psychology of Computer Programming, Van Nostrand Reinhold, 1971.

APPENDIX A
FORTRAN TIMING RESULTS

FORTRAN TIMING RESULTS

PROBLEM # 14 PAGE 195

NUMBER OF STUDENTS= 2

NOTE THAT ALL TIME INFORMATION IS IN SECONDS

(1)	(2)	(3)	(4)	(5)	(6)	
STUDENT ID	PROGRAMMING TIME	KEYPUNCH TIME	DEBUG RUNS	DEBUG TIME	DEBUG KEYPUNCH TIME	WAIT TIME
4.	1800.	900.	3.	1800.	900.	1800.
25.	1500.	1200.	2.	3600.	300.	600.

NOTE THAT THE NUMBER 9191 IS A CODE TO SIGNIFY
THAT NO DATA SUPPLIED FOR THIS INSTANCE

STATISTICAL RESULTS

NOTE THAT TOTAL PROGRAMMING TIME IS NOT
DEFINED IF THERE IS NO DATA FOR EITHER
PROGRAMMING TIME OR FOR DEBUGGING TIME

AVERAGE PROGRAMMING TIME = 1650. SECONDS
AVERAGE DEBUG TIME = 2700. SECONDS

AVERAGE PROGRAMMING TIME + AVERAGE DEBUG TIME = 4350. SECONDS

	AVERAGE	RANGE	PERFORMANCE RATIO	VARIANCE	STANDARD DEVIATION
PROGRAMMING TIME	1650.	300.	1.2	0.45000E 05	212.
KEYPUNCH TIME	1050.	300.	1.3	0.45000E 05	212.
DEBUG RUNS	3.	1.	1.5	0.50000E 00	1.
DEBUG TIME	2700.	1800.	2.0	0.16200E 07	1273.
DEBUG KEYPUNCH TIME	600.	600.	3.0	0.18000E 06	424.
WAIT TIME	1200.	1200.	3.0	0.72000E 06	849.
TCTIPROG+DEBUGTIME	4350.	1500.	1.4	0.11250E 07	1061.

NOTE THAT ALL TIMES ARE IN SECONDS

NOTE THAT IF THE MINIMUM DATA ELEMENT IS 0, THEN ONLY
IN THE CALCULATION FOR PERFORMANCE RATIO THE 0 IN THE
DENOMINATOR IS REPLACED BY 1

DISTRIBUTION OF TOTAL PROGRAMMING TIME

FORTRAN TIMING RESULTS

PROBLEM # 5 PAGE 19

NUMBER OF STUDENTS= 24

NOTE THAT ALL TIME INFORMATION IS IN SECONDS
(1) (2) (3) (4) (5) (6)

STUDENT ID	PROGRAMMING TIME	KEYPUNCH TIME	DEBUG RUNS	DEBUG TIME	DEBUG KEYPUNCH TIME	WAIT TIME
1.	1800.	1800.	1.	300.	0.	1200.
2.	600.	300.	2.	600.	1200.	4800.
3.	1020.	1200.	0.	0.	0.	1200.
4.	900.	2100.	1.	600.	2700.	8100.
5.	4200.	7500.	1.	300.	120.	900.
6.	2700.	3900.	3.	5400.	5400.	2700.
7.	1500.	3600.	3.	3600.	4200.	4500.
8.	300.	600.	0.	0.	0.	1200.
9.	1800.	2100.	2.	900.	900.	1800.
10.	300.	600.	0.	0.	0.	1500.
11.	2280.	1200.	0.	0.	0.	600.
12.	300.	1800.	0.	0.	0.	900.
13.	900.	900.	2.	2100.	300.	6000.
14.	4200.	7200.	3.	10800.	15000.	10800.
15.	1800.	3600.	1.	1200.	1500.	1800.
16.	300.	1020.	1.	300.	60.	1200.
17.	1800.	5400.	2.	2400.	1200.	3600.
18.	5100.	3900.	1.	900.	600.	300.
19.	1800.	3600.	2.	2700.	3600.	5400.
20.	1200.	2700.	3.	1200.	600.	1800.
21.	1800.	3600.	1.	900.	1800.	1680.
22.	2100.	2100.	0.	0.	0.	900.
23.	900.	1800.	3.	2700.	1800.	600.
24.	1500.	2400.	2.	900.	1200.	2100.

NOTE THAT THE NUMBER 9191 IS A CODE TO SIGNIFY
THAT NO DATA SUPPLIED FOR THIS INSTANCE

STATISTICAL RESULTS

NOTE THAT TOTAL PROGRAMMING TIME IS NOT
DEFINED IF THERE IS NO DATA FOR EITHER
PROGRAMMING TIME OR FOR DEBUGGING TIME

AVERAGE PROGRAMMING TIME = 1713. SECONDS
AVERAGE DEBUG TIME = 1575. SECONDS

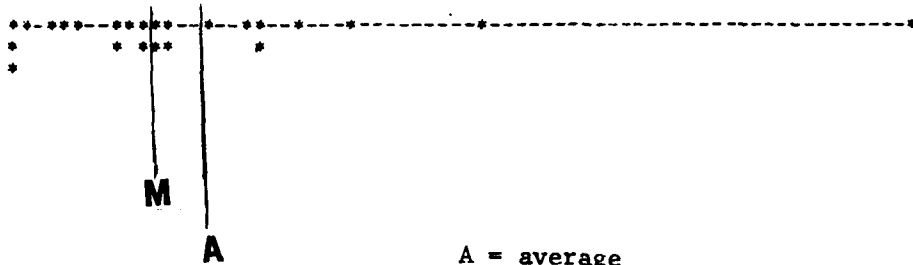
AVERAGE PROGRAMMING TIME + AVERAGE DEBUG TIME = 3288. SECONDS

	AVERAGE	RANGE	PERFORMANCE RATIO	VARIANCE	STANDARD DEVIATION
PROGRAMMING TIME	1713.	4800.	17.0	0.15573E 07	1248.
KEYPUNCH TIME	2705.	7200.	25.0	0.35601E 07	1887.
DEBUG RUNS	1.	3.	3.0	0.11597E 01	1.
DEBUG TIME	1575.	10800.	****	0.54619E 07	2337.
DEBUG KEYPUNCH TIME	1758.	15000.	****	0.96919E 07	3113.
WAIT TIME	2983.	10500.	36.0	0.73360E 07	2709.
TOT(PROG+DEBUG) TIME	3288.	14700.	50.0	0.94922E 07	3091.

NOTE THAT ALL TIMES ARE IN SECONDS

NOTE THAT IF THE MINIMUM DATA ELEMENT IS 0, THEN ONLY
IN THE CALCULATION FOR PERFORMANCE RATIO THE 0 IN THE
DENOMINATOR IS REPLACED BY 1

DISTRIBUTION OF TOTAL PROGRAMMING TIME



A = average
M = median

FORTRAN TIMING RESULTS

PROBLEM # 8 PAGE 19

NUMBER OF STUDENTS= 24

NOTE THAT ALL TIME INFORMATION IS IN SECONDS

	(1)	(2)	(3)	(4)	(5)	(6)
STUDENT ID	PROGRAMMING TIME	KEYPUNCH TIME	DEBUG RUNS	DEBUG TIME	DEBUG KEYPUNCH TIME	WAIT TIME
1.	900.	600.	1.	60.	180.	1200.
2.	600.	900.	2.	600.	1200.	4800.
3.	2220.	1500.	0.	0.	0.	1200.
4.	1800.	2400.	1.	600.	60.	4200.
5.	2700.	5400.	0.	0.	0.	600.
6.	3600.	4500.	2.	3000.	4800.	2700.
7.	1200.	3600.	2.	2700.	4200.	2400.
8.	300.	480.	0.	0.	0.	7200.
9.	2100.	1800.	2.	900.	900.	1800.
10.	420.	900.	1.	300.	180.	1200.
11.	3000.	1320.	0.	0.	0.	1200.
12.	300.	1800.	0.	0.	0.	900.
13.	600.	900.	1.	300.	120.	1500.
14.	3600.	4500.	2.	7800.	3600.	3600.
15.	2700.	4500.	0.	0.	0.	1800.
16.	540.	1200.	0.	0.	0.	1800.
17.	1800.	5400.	2.	1200.	600.	3600.
18.	2100.	1500.	0.	0.	0.	300.
19.	3600.	3600.	1.	900.	900.	1800.
20.	900.	2700.	2.	900.	600.	900.
21.	1200.	2400.	3.	4800.	900.	2040.
22.	2700.	2700.	1.	1200.	300.	1800.
23.	900.	900.	9191.	3600.	1800.	600.
24.	1800.	2100.	2.	1200.	900.	2400.

NOTE THAT THE NUMBER 9191 IS A CODE TO SIGNIFY THAT NO DATA SUPPLIED FOR THIS INSTANCE

STATISTICAL RESULTS

NOTE THAT TOTAL PROGRAMMING TIME IS NOT DEFINED IF THERE IS NO DATA FOR EITHER PROGRAMMING TIME OR FOR DEBUGGING TIME

AVERAGE PROGRAMMING TIME = 1733. SECONDS

AVERAGE DEBUG TIME = 1253. SECONDS

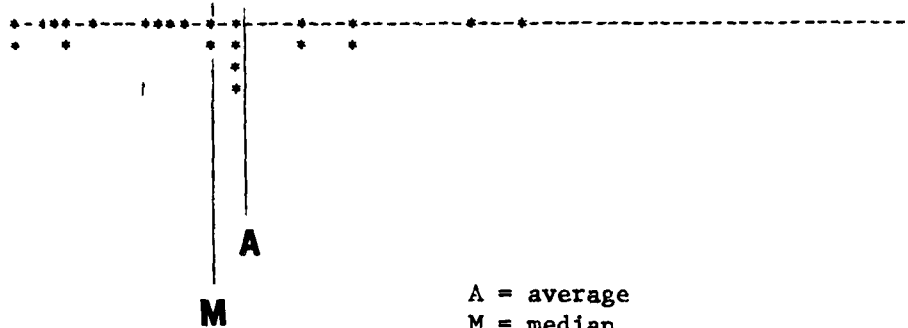
AVERAGE PROGRAMMING TIME + AVERAGE DEBUG TIME = 2985. SECONDS

	AVERAGE	RANGE	PERFORMANCE RATIO	VARIANCE	STANDARD DEVIATION
PROGRAMMING TIME	1733.	3300.	12.0	0.11608E 07	1077.
KEYPUNCH TIME	2400.	4920.	11.3	0.23022E 07	1517.
DEBUG RUNS	1.	3.	3.0	0.86201E 00	1.
DEBUG TIME	1253.	7800.	*****	0.34639E 07	1861.
DEBUG KEYPUNCH TIME	885.	4800.	*****	0.18190E 07	1349.
WAIT TIME	2148.	6900.	24.0	0.23904E 07	1546.
TOT(PROG+DEBUG)TIME	2985.	11100.	33.0	0.58198E 07	2412.

NOTE THAT ALL TIMES ARE IN SECONDS

NOTE THAT IF THE MINIMUM DATA ELEMENT IS 0, THEN ONLY
IN THE CALCULATION FOR PERFORMANCE RATIO THE 0 IN THE
DENOMINATOR IS REPLACED BY 1

DISTRIBUTION OF TOTAL PROGRAMMING TIME



A = average

M = median

FERIRAN TIMING RESULTS

PROBLEM # 4 PAGE 99 HWK 5

NUMBER OF STUDENTS= 2

NOTE THAT ALL TIME INFORMATION IS IN SECONDS

	(1)	(2)	(3)	(4)	(5)	(6)
SILENT ID	PROGRAMMING TIME	KEYPUNCH TIME	DEBUG RUNS	DEBUG TIME	DEBUG KEYPUNCH TIME	WAIT TIME
7.	2400.	2400.	2.	2400.	3000.	1800.
5.	1800.	600.	0.	0.	0.	300.

NOTE THAT THE NUMBER 9191 IS A CODE TO SIGNIFY THAT NO DATA SUPPLIED FOR THIS INSTANCE

STATISTICAL RESULTS

NOTE THAT TOTAL PROGRAMMING TIME IS NOT DEFINED IF THERE IS NO DATA FOR EITHER PROGRAMMING TIME OR FOR DEBUGGING TIME

AVERAGE PROGRAMMING TIME = 2100. SECONDS

AVERAGE DEBUG TIME = 1200. SECONDS

AVERAGE PROGRAMMING TIME + AVERAGE DEBUG TIME = 3300. SECONDS

	AVERAGE	RANGE	PERFORMANCE RATIO	VARIANCE	STANDARD DEVIATION
PROGRAMMING TIME	2100.	600.	1.3	0.18000E 06	424.
KEYPUNCH TIME	1500.	1800.	4.0	0.16200E 07	1273.
DEBUG RUNS	1.	2.	2.0	0.20000E 01	1.
DEBUG TIME	1200.	2400.	*****	0.28800E 07	1697.
CEBUG KEYPUNCH TIME	1500.	3000.	*****	0.45000E 07	2121.
WAIT TIME	1050.	1500.	6.0	0.11250E 07	1061.
TCT(PROG+DEBUG) TIME	3300.	3000.	2.7	0.45000E 07	2121.

NOTE THAT ALL TIMES ARE IN SECONDS

NOTE THAT IF THE MINIMUM DATA ELEMENT IS 0, THEN ONLY IN THE CALCULATION FOR PERFORMANCE RATIO THE 0 IN THE DENOMINATOR IS REPLACED BY 1

DISTRIBUTION OF TOTAL PROGRAMMING TIME

-----*

FORTRAN TIMING RESULTS

PROBLEM # 13 PAGE 90 HWK 5(11 4)

NUMBER OF STUDENTS= 16

NOTE THAT ALL TIME INFORMATION IS IN SECONDS

	(1)	(2)	(3)	(4)	(5)	(6)
STUDENT ID	PROGRAMMING TIME	KEYPUNCH TIME	DEBUG RUNS	DEBUG TIME	DEBUG KEYPUNCH TIME	WAIT TIME
8.	600.	900.	2.	600.	300.	600.
6.	7800.	4200.	4.	900.	7800.	3900.
7.	1800.	1800.	2.	1800.	2400.	1500.
21.	4500.	2700.	3.	6300.	1800.	5400.
16.	1800.	1800.	3.	1500.	900.	3600.
15.	3600.	3600.	1.	4500.	900.	5400.
12.	1200.	2700.	0.	0.	0.	2700.
17.	3600.	3600.	1.	1800.	900.	1800.
22.	4800.	1500.	1.	900.	600.	6000.
25.	1800.	1200.	1.	600.	300.	600.
26.	600.	1200.	2.	360.	300.	1200.
18.	9000.	3600.	1.	600.	600.	300.
3.	5400.	2700.	8.	13800.	3000.	9000.
24.	2700.	1800.	4.	3000.	2400.	1800.
23.	3600.	2700.	5.	3600.	3600.	3600.
1.	3600.	3600.	2.	3600.	3600.	1800.

NOTE THAT THE NUMBER 9191 IS A CODE TO SIGNIFY THAT NO DATA SUPPLIED FOR THIS INSTANCE

STATISTICAL RESULTS

NOTE THAT TOTAL PROGRAMMING TIME IS NOT DEFINED IF THERE IS NO DATA FOR EITHER PROGRAMMING TIME OR FOR DEBUGGING TIME

AVERAGE PROGRAMMING TIME = 3525. SECONDS
AVERAGE DEBUG TIME = 2741. SECONDS

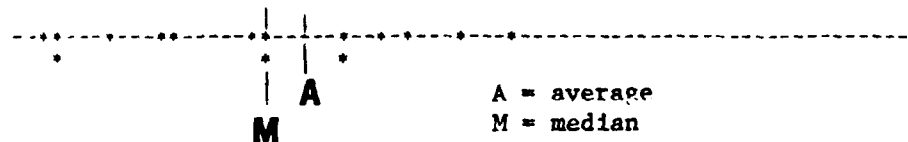
AVERAGE PROGRAMMING TIME + AVERAGE DEBUG TIME = 6266. SECONDS

	AVERAGE	RANGE	PERFORMANCE RATIO	VARIANCE	STANDARD DEVIATION
PROGRAMMING TIME	3525.	8400.	15.0	0.54056E 07	2325.
KEYPUNCH TIME	2475.	3300.	4.7	0.10181E 07	1009.
DEBUG RUNS	3.	8.	8.0	0.37500E 01	2.
DEBUG TIME	2741.	13800.	*****	0.11039E 08	3323.
DEBUG KEYPUNCH TIME	1838.	7800.	*****	0.37448E 07	1935.
WAIT TIME	3075.	8700.	30.0	0.54169E 07	2327.
TOTAL (PROG+DEBUG) TIME	6266.	18240.	20.0	0.20107E 08	4484.

NOTE THAT ALL TIMES ARE IN SECONDS

NOTE THAT IF THE MINIMUM DATA ELEMENT IS 0, THEN ONLY IN THE CALCULATION FOR PERFORMANCE RATIO THE 0 IN THE DENOMINATOR IS REPLACED BY 1

DISTRIBUTION OF TOTAL PROGRAMMING TIME



FORTRAN TIMING RESULTS

PROBLEM # 19 PAGE 90 HWK 5

NUMBER OF STUDENTS= 16

NOTE THAT ALL TIME INFORMATION IS IN SECONDS

	(1)	(2)	(3)	(4)	(5)	(6)
STUDENT ID	PROGRAMMING TIME	KEYPUNCH TIME	DEBUG RUNS	DEBUG TIME	DEBUG KEYPUNCH TIME	WAIT TIME
8.	900.	900.	3.	900.	300.	600.
6.	7200.	3300.	3.	7800.	6300.	3600.
7.	1500.	1500.	1.	1800.	1800.	600.
21.	3600.	2100.	2.	3600.	1500.	3600.
16.	2700.	2100.	5.	2400.	1200.	5400.
12.	1800.	2700.	1.	600.	0.	1200.
17.	1800.	3600.	1.	900.	900.	1800.
22.	5700.	1740.	2.	1740.	1200.	7200.
20.	2700.	1800.	5.	1800.	1800.	9191.
25.	1800.	1200.	3.	3600.	900.	1500.
26.	600.	1200.	1.	60.	60.	900.
18.	3600.	2700.	5.	7200.	3600.	2400.
3.	9000.	3000.	8.	4800.	3600.	9000.
24.	1800.	1200.	3.	3000.	2400.	1200.
23.	2700.	2700.	5.	3600.	3600.	3600.
1.	3600.	3600.	4.	3600.	1800.	1800.

NOTE THAT THE NUMBER 9191 IS A CODE TO SIGNIFY
THAT NO DATA SUPPLIED FOR THIS INSTANCE

STATISTICAL RESULTS

NOTE THAT TOTAL PROGRAMMING TIME IS NOT
DEFINED IF THERE IS NO DATA FOR EITHER
PROGRAMMING TIME OR FOR DEBUGGING TIME

AVERAGE PROGRAMMING TIME = 3188. SECONDS

AVERAGE DEBUG TIME = 2963. SECONDS

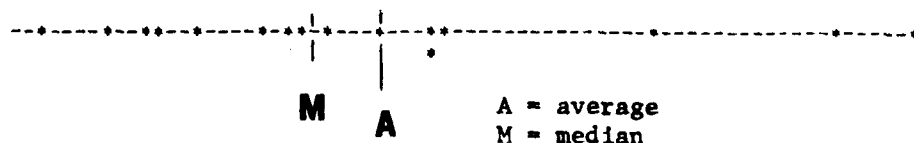
AVERAGE PROGRAMMING TIME + AVERAGE DEBUG TIME = 6150. SECONDS

	AVERAGE	RANGE	PERFORMANCE RATIO	VARIANCE	STANDARD DEVIATION
PROGRAMMING TIME	3188.	8400.	15.0	0.49936E 07	2235.
KEYPUNCH TIME	2209.	2700.	4.0	0.75565E 06	969.
DEBUG RUNS	3.	7.	8.0	0.36875E 01	2.
DEBUG TIME	2963.	7740.	130.0	0.45868E 07	2142.
DEBUG KEYPUNCH TIME	1935.	6300.	****	0.25616E 07	1601.
WAIT TIME	2960.	8400.	15.0	0.58904E 07	2427.
TOT (PROG+DEBUG) TIME	6150.	14340.	22.7	0.15732E 08	3966.

NOTE THAT ALL TIMES ARE IN SECONDS

NOTE THAT IF THE MINIMUM DATA ELEMENT IS 0, THEN ONLY
IN THE CALCULATION FOR PERFORMANCE RATIO THE 0 IN THE
DENOMINATOR IS REPLACED BY 1

DISTRIBUTION OF TOTAL PROGRAMMING TIME



PERIKAN TIMING RESULTS

PROBLEM # 3 PAGE 115 HWK 6(EN 5)

NUMBER OF STUDENTS= 18

NOTE THAT ALL TIME INFORMATION IS IN SECONDS

	(1)	(2)	(3)	(4)	(5)	(6)
SILENT IL	PROGRAMMING TIME	KEYPUNCH TIME	DEBUG RUNS	DEBUG TIME	DEBUG KEYPUNCH TIME	WAIT TIME
5.	900.	900.	3.	900.	600.	1800.
6.	3600.	3300.	4.	6900.	3600.	3900.
7.	1500.	1500.	3.	3000.	3000.	1800.
21.	6300.	5700.	2.	5400.	600.	5400.
16.	1200.	900.	2.	1500.	600.	4500.
12.	1800.	2700.	0.	0.	0.	1800.
15.	2700.	9191.	1.	1800.	900.	1800.
4.	1500.	2100.	3.	1200.	600.	2400.
22.	5400.	1980.	1.	1500.	900.	7200.
20.	900.	2100.	2.	1500.	900.	9191.
25.	1800.	1200.	4.	3600.	1200.	1800.
26.	1200.	1500.	1.	600.	240.	1200.
13.	14400.	5400.	3.	3600.	2700.	600.
13.	1800.	900.	7.	8400.	1800.	4500.
3.	3000.	1800.	6.	10200.	3600.	7200.
24.	1800.	1200.	4.	2400.	2100.	1800.
21.	3600.	3600.	6.	5400.	2700.	2700.
1.	3600.	1800.	2.	3600.	2400.	1800.

NOTE THAT THE NUMBER 9191 IS A CODE TO SIGNIFY
THAT NO DATA SUPPLIED FOR THIS INSTANCE

STATISTICAL RESULTS

NOTE THAT TOTAL PROGRAMMING TIME IS NOT
DEFINED IF THERE IS NO DATA FOR EITHER
PROGRAMMING TIME OR FOR DEBUGGING TIME

AVERAGE PROGRAMMING TIME = 3167. SECONDS
AVERAGE DEBUG TIME = 3417. SECONDS

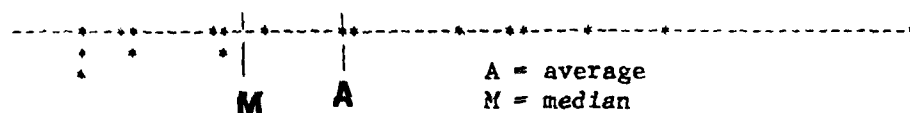
AVERAGE PROGRAMMING TIME + AVERAGE DEBUG TIME = 6583. SECONDS

	AVERAGE	RANGE	PERFORMANCE RATIO	*VARIANCE	STANDARD DEVIATION
PROGRAMMING TIME	3167.	13500.	16.0	0.96022E 07	3099.
KEYPUNCH TIME	2269.	4800.	6.3	0.20157E 07	1420.
DEBUG RUNS	3.	7.	7.0	0.34444E 01	2.
DEBUG TIME	3417.	10200.	*****	0.75914E 07	2755.
DEBUG KEYPUNCH TIME	1580.	3600.	*****	0.12968E 07	1139.
WAIT TIME	3071.	6600.	12.0	0.38703E 07	1967.
TOT(PROG+DEBUG)TIME	6583.	16200.	10.0	0.20395E 08	4516.

NOTE THAT ALL TIMES ARE IN SECONDS

NOTE THAT IF THE MINIMUM DATA ELEMENT IS 0, THEN ONLY
IN THE CALCULATION FOR PERFORMANCE RATIO THE 0 IN THE
DENOMINATOR IS REPLACED BY 1

DISTRIBUTION OF TOTAL PROGRAMMING TIME



FORTRAN TIMING RESULTS

PROBLEM # 9 PAGE 115 HWK 61CH 51

NUMBER OF STUDENTS= 16

NOTE THAT ALL TIME INFORMATION IS IN SECONDS

(1)	(2)	(3)	(4)	(5)	(6)
STUDENT ID	PROGRAMMING TIME	KEYPUNCH TIME	DEBUG RUNS	DEBUG TIME	DEBUG KEYPUNCH TIME
7.	1800.	1800.	4.	3000.	3000.
21.	7200.	7800.	3.	10800.	2400.
16.	600.	1200.	3.	1800.	900.
12.	2100.	3000.	1.	1200.	600.
15.	4200.	3600.	5.	7200.	5400.
4.	1800.	3300.	4.	4200.	2400.
22.	5280.	1920.	2.	2400.	1320.
20.	1800.	3600.	3.	2700.	1200.
25.	1800.	1200.	3.	5400.	900.
9.	900.	900.	2.	600.	300.
6.	8400.	4800.	4.	7800.	6900.
26.	480.	900.	1.	9191.	9191.
3.	6000.	5400.	15.	6600.	3600.
24.	1500.	1800.	3.	1800.	1500.
23.	3600.	3600.	6.	7200.	2700.
1.	3600.	1800.	2.	3600.	60.

NOTE THAT THE NUMBER 9191 IS A CODE TO SIGNIFY THAT NO DATA SUPPLIED FOR THIS INSTANCE

STATISTICAL RESULTS

NOTE THAT TOTAL PROGRAMMING TIME IS NOT DEFINED IF THERE IS NO DATA FOR EITHER PROGRAMMING TIME OR FOR DEBUGGING TIME

AVERAGE PROGRAMMING TIME = 3191. SECONDS
AVERAGE DEBUG TIME = 4420. SECONDS

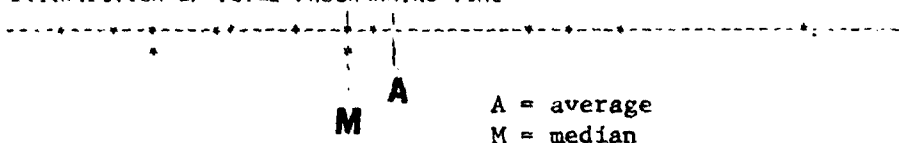
AVERAGE PROGRAMMING TIME + AVERAGE DEBUG TIME = 7611. SECONDS

	AVERAGE	RANGE	PERFORMANCE RATIO	VARIANCE	STANDARD DEVIATION
PROGRAMMING TIME	3191.	7920.	17.5	0.54946E 07	2344.
KEYPUNCH TIME	2914.	6900.	8.7	0.33673E 07	1935.
DEBUG RUNS	4.	14.	15.0	0.10027E 02	3.
DEBUG TIME	4420.	10200.	18.0	0.81176E 07	2849.
DEBUG KEYPUNCH TIME	2212.	6840.	115.0	0.34435E 07	1956.
WAIT TIME	4600.	12600.	11.5	0.13208E 08	3634.
TCT (PROG+DEBUG) TIME	7792.	16500.	12.0	0.23521E 08	4850.

NOTE THAT ALL TIMES ARE IN SECONDS

NOTE THAT IF THE MINIMUM DATA ELEMENT IS 0, THEN ONLY IN THE CALCULATION FOR PERFORMANCE RATIO THE 0 IN THE DENOMINATOR IS REPLACED BY 1

DISTRIBUTION OF TOTAL PROGRAMMING TIME



FORTRAN TIMING RESULTS

PROBLEM # 11 PAGE 115 HWK 6001 51

NUMBER OF STUDENTS= 19

NOTE THAT ALL TIME INFORMATION IS IN SECONDS

	(1)	(2)	(3)	(4)	(5)	(6)
STUDENT ID	PROGRAMMING TIME	KEYPUNCH TIME	DEBUG RUNS	DEBUG TIME	DEBUG KEYPUNCH TIME	WAIT TIME
3.	1800.	1800.	3.	1800.	900.	1800.
6.	5400.	3000.	2.	3600.	5400.	3000.
7.	1500.	1800.	4.	3000.	3000.	1200.
21.	6600.	2400.	2.	4800.	1300.	7200.
5.	4500.	3600.	3.	7800.	600.	300.
16.	2100.	1800.	4.	2100.	1200.	5400.
12.	4200.	3600.	1.	2100.	1200.	3000.
15.	4800.	2400.	1.	600.	900.	1200.
17.	3600.	3600.	2.	3600.	1800.	1800.
4.	6600.	2700.	3.	1200.	900.	4200.
22.	6300.	2100.	1.	1500.	900.	7200.
20.	1200.	1800.	2.	600.	1200.	9191.
25.	5700.	3300.	5.	14400.	1500.	2700.
26.	1500.	1800.	2.	600.	480.	1200.
18.	14400.	5400.	0.	0.	0.	300.
3.	6300.	3600.	15.	10800.	4800.	7200.
24.	3600.	2400.	4.	2700.	2400.	1800.
23.	3600.	2700.	6.	5400.	2700.	3600.
1.	3600.	3600.	2.	3600.	1800.	3000.

NOTE THAT THE NUMBER 9191 IS A CODE TO SIGNIFY THAT NO DATA SUPPLIED FOR THIS INSTANCE

STATISTICAL RESULTS

NOTE THAT TOTAL PROGRAMMING TIME IS NOT DEFINED IF THERE IS NO DATA FOR EITHER PROGRAMMING TIME OR FOR DEBUGGING TIME

AVERAGE PROGRAMMING TIME = 4595. SECONDS
AVERAGE DEBUG TIME = 3695. SECONDS

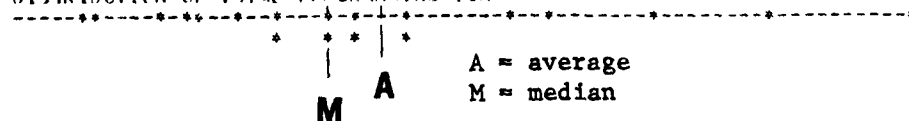
AVERAGE PROGRAMMING TIME + AVERAGE DEBUG TIME = 8289. SECONDS

	AVERAGE	RANGE	PERFORMANCE RATIO	VARIANCE	STANDARD DEVIATION
PROGRAMMING TIME	4595.	13200.	12.0	0.84605E 07	2909.
KEYPUNCH TIME	2811.	3600.	3.0	0.85463E 06	924.
DEBUG RUNS	3.	15.	15.0	0.97729E 01	3.
DEBUG TIME	3695.	14400.	*****	0.13140E 08	3625.
DEBUG KEYPUNCH TIME	1762.	5400.	*****	0.18618E 07	1364.
WAIT TIME	3117.	6900.	24.0	0.49614E 07	2223.
TCT (PROG+DEBUG) TIME	8289.	18300.	11.2	0.22834E 08	4778.

NOTE THAT ALL TIMES ARE IN SECONDS

NOTE THAT IF THE MINIMUM DATA ELEMENT IS 0, THEN ONLY IN THE CALCULATION FOR PERFORMANCE RATIO THE 0 IN THE

DISTRIBUTION OF TOTAL PROGRAMMING TIME



FORTRAN TYPING RESULTS

PROBLEM # 2 PAGE 194 HWK 8

NUMBER OF STUDENTS= 4

NOTE THAT ALL TIME INFORMATION IS IN SECONDS

	(1)	(2)	(3)	(4)	(5)	(6)
STUDENT ID	PROGRAMMING TIME	KEYPUNCH TIME	DEBUG RUNS	DEBUG TIME	DEBUG KEYPUNCH TIME	WAIT TIME
8.	1200.	1200.	2.	900.	600.	1200.
25.	2100.	1200.	3.	2400.	600.	1200.
19.	1800.	1800.	0.	0.	0.	300.
1.	1800.	1200.	1.	600.	120.	300.

NOTE THAT THE NUMBER 9191 IS A CODE TO SIGNIFY THAT NO DATA SUPPLIED FOR THIS INSTANCE

STATISTICAL RESULTS

NOTE THAT TOTAL PROGRAMMING TIME IS NOT DEFINED IF THERE IS NO DATA FOR EITHER PROGRAMMING TIME OR FOR DEBUGGING TIME

AVERAGE PROGRAMMING TIME = 1725. SECONDS
AVERAGE DEBUG TIME = 975. SECONDS

AVERAGE PROGRAMMING TIME + AVERAGE DEBUG TIME = 2700. SECONDS

	AVERAGE	RANGE	PERFORMANCE RATIO	VARIANCE	STANDARD DEVIATION
PROGRAMMING TIME	1725.	900.	1.8	0.10688E 06	327.
KEYPUNCH TIME	1350.	600.	1.5	0.67500E 05	260.
DEBLG RUNS	2.	3.	3.0	0.12500E 01	1.
DEBUG TIME	975.	2400.	****	0.78188E 06	884.
DEBUG KEYPUNCH TIME	330.	600.	600.0	0.74700E 05	273.
WAIT TIME	750.	900.	4.0	0.20250E 06	450.
TOT(PROG+DEBUG)TIME	2700.	2700.	2.5	0.11250E 07	1061.

NOTE THAT ALL TIMES ARE IN SECONDS

NOTE THAT IF THE MINIMUM DATA ELEMENT IS 0, THEN ONLY IN THE CALCULATION FOR PERFORMANCE RATIO THE 0 IN THE DENOMINATOR IS REPLACED BY 1

DISTRIBUTION OF TOTAL PROGRAMMING TIME

-----*

FORTRAN TIMING RESULTS

PROBLEM # 2181 PAGE 49 HWK 4

NUMBER OF STUDENTS= 18

NOTE THAT ALL TIME INFORMATION IS IN SECONDS

	(1)	(2)	(3)	(4)	(5)	(6)
STUDENT ID	PROGRAMMING TIME	KEYPUNCH TIME	DEBUG RUNS	DEBUG TIME	DEBUG KEYPUNCH TIME	WAIT TIME
8.	900.	900.	2.	300.	60.	1800.
6.	2400.	1800.	3.	6000.	4300.	2700.
7.	1200.	1200.	2.	1800.	2400.	900.
21.	1800.	2700.	1.	2400.	900.	3600.
19.	3600.	2700.	2.	3600.	1800.	7200.
12.	900.	900.	0.	0.	0.	1800.
15.	2700.	2400.	2.	1800.	900.	1800.
17.	3600.	3600.	3.	7200.	1800.	2700.
4.	2400.	6000.	1.	600.	300.	6300.
22.	2400.	900.	0.	0.	0.	3300.
20.	1200.	1800.	4.	2700.	2700.	9191.
25.	1800.	1200.	1.	1800.	300.	900.
26.	480.	900.	1.	120.	120.	1200.
18.	7200.	2400.	1.	1500.	900.	300.
3.	3600.	1800.	2.	2400.	1200.	1500.
24.	1500.	1500.	2.	900.	900.	1200.
23.	2700.	2700.	4.	3600.	3600.	3600.
1.	3600.	3600.	4.	3600.	3600.	3600.

NOTE THAT THE NUMBER 9191 IS A CODE TO SIGNIFY
THAT NO DATA SUPPLIED FOR THIS INSTANCE

STATISTICAL RESULTS

NOTE THAT TOTAL PROGRAMMING TIME IS NOT
DEFINED IF THERE IS NO DATA FOR EITHER
PROGRAMMING TIME OR FOR DEBUGGING TIME

AVERAGE PROGRAMMING TIME = 2443. SECONDS

AVERAGE DEBUG TIME = 2240. SECONDS

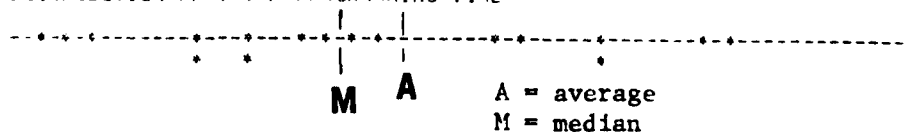
AVERAGE PROGRAMMING TIME + AVERAGE DEBUG TIME = 4683. SECONDS

	AVERAGE	RANGE	PERFORMANCE RATIO	VARIANCE	STANDARD DEVIATION
PROGRAMMING TIME	2443.	6720.	15.0	0.23079E 07	1519.
KEYPUNCH TIME	2167.	5100.	6.7	0.16056E 07	1267.
DEBUG RUNS	2.	4.	4.0	0.14969E 01	1.
DEBUG TIME	2240.	7200.	*****	0.38032E 07	1950.
DEBUG KEYPUNCH TIME	1460.	4800.	*****	0.19444E 07	1394.
WAIT TIME	2612.	6900.	24.0	0.33222E 07	1823.
TOT(PRG+DEBUG) TIME	4683.	10200.	18.0	0.80914E 07	2845.

NOTE THAT ALL TIMES ARE IN SECONDS

NOTE THAT IF THE MINIMUM DATA ELEMENT IS 0, THEN ONLY
IN THE CALCULATION FOR PERFORMANCE RATIO THE 0 IN THE
DENOMINATOR IS REPLACED BY 1

DISTRIBUTION OF TOTAL PROGRAMMING TIME



PERMAN TIMING RESULTS

PROBLEM # 21E) PAGE 50 HWK 4

NUMBER OF STUDENTS= 19

NOTE THAT ALL TIME INFORMATION IS IN SECONDS

	(1)	(2)	(3)	(4)	(5)	(6)
SILENT ID	PROGRAMMING TIME	KEYPUNCH TIME	DEBUG RUNS	DEBUG TIME	DEBUG KEYPUNCH TIME	WAIT TIME
9.	1200.	900.	2.	600.	300.	900.
6.	3900.	2700.	3.	5400.	5400.	3000.
7.	1200.	1200.	2.	1800.	2400.	900.
21.	2700.	1500.	2.	1500.	1860.	1800.
19.	3600.	2700.	2.	3600.	1800.	7200.
5.	1800.	600.	0.	0.	0.	600.
12.	900.	1200.	0.	0.	0.	1500.
15.	1200.	2700.	1.	900.	1800.	1800.
17.	3600.	3600.	4.	10800.	2700.	3600.
4.	1500.	4800.	2.	600.	300.	3600.
22.	2700.	1500.	0.	0.	0.	4200.
20.	1200.	2700.	3.	1500.	1500.	9191.
25.	1500.	1200.	1.	1800.	420.	1200.
26.	900.	900.	1.	300.	300.	1500.
18.	3600.	2400.	2.	900.	1800.	1800.
3.	6600.	3000.	3.	1800.	1500.	1500.
24.	1200.	1500.	1.	900.	600.	900.
23.	2700.	3600.	5.	3600.	3600.	3600.
1.	3600.	3600.	7.	3600.	3600.	3600.

NOTE THAT THE NUMBER 9191 IS A CODE TO SIGNIFY
THAT NO DATA SUPPLIED FOR THIS INSTANCE

STATISTICAL RESULTS

NOTE THAT TOTAL PROGRAMMING TIME IS NOT
DEFINED IF THERE IS NO DATA FOR EITHER
PROGRAMMING TIME OR FOR DEBUGGING TIME

AVERAGE PROGRAMMING TIME = 2400. SECONDS
AVERAGE DEBUG TIME = 2084. SECONDS

AVERAGE PROGRAMMING TIME + AVERAGE DEBUG TIME = 4484. SECONDS

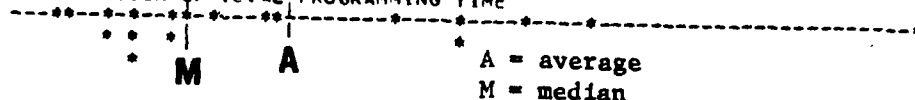
	AVERAGE	RANGE	PERFORMANCE RATIO	VARIANCE	STANDARD DEVIATION
PROGRAMMING TIME	2400.	5700.	7.3	0.20842E 07	1444.
KEYPUNCH TIME	2226.	4200.	8.0	0.13009E 07	1141.
DEBUG RUNS	2.	7.	7.0	0.29751E 01	2.
DEBUG TIME	2084.	10800.	*****	0.62950E 07	2509.
DEBUG KEYPUNCH TIME	1573.	5400.	*****	0.20856E 07	1444.
WAIT TIME	2400.	6600.	12.0	0.26300E 07	1622.
TOT(PROG+DEBUG) TIME	4484.	13500.	16.0	0.11581E 08	3403.

NOTE THAT ALL TIMES ARE IN SECONDS

NOTE THAT IF THE MINIMUM DATA ELEMENT IS 0, THEN ONLY
IN THE CALCULATION FOR PERFORMANCE RATIO THE 0 IN THE

DEACMINATOR IS REPLACED BY 1

DISTRIBUTION OF TOTAL PROGRAMMING TIME



APPENDIX B
ANALYSIS OF VARIANCE RESULTS

SET#1

RAW DATA

COL.1=GROUP NUMBER COL.2=STUDENT NUMBER WITHIN GROUP
 COL.3=K-PROBLEM NUMBER COL.4=K-PROGRAMMING TIME COL.5=K-DEBUG TIME
 COL.6=F-PROBLEM NUMBER COL.7=F-PROGRAMMING TIME COL.8=F-KEYPUNCH TIME
 COL.9=NUMBER OF DEBUG RUNS COL.10=F-DEBUG TIME COL.11=DEBUG KEYPUNCH TIME
 COL.12=WAIT TIME FOR RESULTS

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
1	1	2	60.	0.	1	600.	600.	2	1800.	120.	2700.
1	2	2	90.	0.	1	300.	240.	0	0.	0.	1200.
1	3	2	900.	0.	1	660.	300.	0	0.	0.	0.
1	4	2	60.	0.	1	3600.	600.	0	0.	0.	0.
1	5	2	120.	0.	1	7200.	9191.	0	0.	0.	0.
1	6	2	600.	0.	1	1200.	1200.	2	600.	600.	1800.
1	7	2	25.	15.	1	60.	9191.	0	0.	0.	0.
1	8	2	50.	0.	1	300.	300.	0	0.	0.	1800.
1	9	2	120.	120.	1	1800.	1500.	0	0.	0.	0.
1	10	2	240.	0.	1	900.	900.	0	0.	0.	0.
2	1	1	150.	0.	2	300.	600.	0	0.	0.	600.
2	2	1	90.	0.	2	420.	1200.	0	0.	0.	600.
2	3	1	120.	120.	2	300.	300.	0	0.	0.	600.
2	4	1	120.	60.	2	300.	180.	1	90.	60.	2700.
2	5	1	240.	60.	2	480.	600.	0	0.	0.	1200.
2	6	1	120.	120.	2	300.	600.	2	300.	300.	3600.
2	7	1	1800.	900.	2	3600.	9191.	0	0.	0.	0.
2	8	1	300.	300.	2	300.	600.	0	0.	0.	0.
2	9	1	180.	300.	2	300.	180.	0	0.	0.	1500.
2	10	1	180.	120.	2	300.	600.	0	0.	0.	0.

NUMBER OF STUDENTS= 20
 DIVIDED EQUALLY INTO 2 GROUPS

ALL TIMES ARE IN SECONDS

THE DATA ELEMENT=9191. OR 91 SIGNIFIES THAT NO ACTUAL DATA SUPPLIED

TOTAL PROGRAMMING TIME (INCLUDES DEBUG TIME)

PROBLEM 1	K-LANGUAGE	F-LANGUAGE
	150.	2400.
	90.	300.
	240.	660.
	180.	3600.
	300.	7200.
	240.	1800.
	2700.	60.
	600.	300.
	480.	1800.
	300.	900.

PROBLEM 2	K-LANGUAGE	F-LANGUAGE
	60.	300.
	90.	420.
	900.	300.
	60.	390.
	120.	480.
	600.	600.
	40.	3600.
	50.	500.
	240.	300.
	240.	300.

MEAN PROGRAMMING TIME (TOTAL) OF EACH PROBLEM-LANGUAGE COMBINATION

	K-LANGUAGE	F-LANGUAGE
PROBLEM 1	528.	1902.
PROBLEM 2	240.	699.

STANDARD DEVIATION (PROG. TIME) OF EACH PROBLEM LANGUAGE COMBINATION

	K-LANGUAGE	F-LANGUAGE
PROBLEM 1	738.	2057.
PROBLEM 2	273.	972.

S1= 0.1023557E 09.

S2= 0.4442586E 08

S3= 0.3677510E 08

S4= 0.3393309E 08

S5= 0.2337539E 08

S6=INITIAL SUM OF SQUARES= 0.7398029E 08

S7=WITHIN-CELLS OF SQUARES= 0.5792982E 08

S8=ROWS SUMS OF SQUARES= 0.5557696E 07

S9=COLUMNS SUM OF SQUARES= 0.8399712E 07

S10=INTERACTION SUM OF SQUARES= 0.2093056E 07

MSWC= 0.1609161E 07

PROBLEM F(1,4(L-1))= 0.3453784E 01

LANGUAGE F(1,4(L-1))= 0.5219933E 01

4(L-1)= 0.3600000E 02

SET#1, (WEIGHTED DEBUG TIME, +50% INCORRECT PROGRAM, +20% MINOR ERROR)

RAW DATA

COL.1=GROUP NUMBER COL.2=STUDENT NUMBER WITHIN GROUP
 COL.3=K-PROBLEM NUMBER COL.4=K-PROGRAMMING TIME COL.5=K-DEBUG TIME
 COL.6=F-PROBLEM NUMBER COL.7=F-PROGRAMMING TIME COL.8=F-KEYPUNCH TIME
 COL.9=NUMBER OF DEBUG RUNS COL.10=F-DEBUG TIME COL.11=DEBUG KEYPUNCH TIME
 COL.12=WAIT TIME FOR RESULTS

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
1	1	2	60.	0.	1	600.	600.	2	1800.	120.	2700.
1	2	2	90.	0.	1	300.	240.	0	0.	0.	1200.
1	3	2	900.	0.	1	660.	300.	0	132.	0.	0.
1	4	2	60.	0.	1	3600.	600.	0	1800.	0.	0.
1	5	2	120.	0.	1	7200.	9191.	0	3600.	0.	0.
1	6	2	600.	0.	1	1200.	1200.	2	600.	600.	1800.
1	7	2	25.	15.	1	60.	9191.	0	30.	0.	0.
1	8	2	50.	0.	1	300.	300.	0	150.	0.	1800.
1	9	2	120.	144.	1	1800.	1500.	0	0.	0.	0.
1	10	2	240.	0.	1	900.	900.	0	0.	0.	0.
2	1	1	150.	0.	2	300.	600.	0	0.	0.	600.
2	2	1	90.	0.	2	420.	1200.	0	210.	0.	600.
2	3	1	120.	120.	2	300.	300.	0	0.	0.	600.
2	4	1	120.	60.	2	300.	180.	1	90.	60.	2700.
2	5	1	240.	60.	2	480.	600.	0	0.	0.	1200.
2	6	1	120.	120.	2	300.	600.	2	300.	300.	3600.
2	7	1	1800.	1800.	2	3600.	9191.	0	1800.	0.	0.
2	8	1	300.	360.	2	300.	600.	0	60.	0.	0.
2	9	1	180.	300.	2	300.	180.	0	0.	0.	1500.
2	10	1	180.	120.	2	300.	600.	0	0.	0.	0.

NUMBER OF STUDENTS= 20
 DEVIDED EQUALLY INTO 2 GROUPS

ALL TIMES ARE IN SECONDS

THE DATA ELEMENT=9191. OR 91 SIGNIFIES THAT NO ACTUAL DATA SUM
 TOTAL PROGRAMMING TIME (INCLUDES DEBUG TIME)

PROBLEM 1	K-LANGUAGE	F-LANGUAGE
	150.	2400.
	90.	300.
	240.	792.
	180.	5400.
	300.	10900.
	240.	1800.
	3600.	90.
	660.	450.
	480.	1800.
	300.	900.

PROBLEM 2	K-LANGUAGE	F-LANGUAGE
	60.	300.
	90.	630.
	900.	300.
	60.	390.
	120.	480.
	600.	600.
	40.	5400.
	50.	360.
	264.	300.
	240.	300.

MEAN PROGRAMMING TIME (TOTAL) OF EACH PROBLEM-LANGUAGE COMBINATION

	K-LANGUAGE	F-LANGUAGE
PROBLEM 1	624.	2473.
PROBLEM 2	242.	906.

STANDARD DEVIATION (PROG. TIME) OF EACH PROBLEM LANGUAGE COMBINATION

	K-LANGUAGE	F-LANGUAGE
PROBLEM 1	1005.	3160.
PROBLEM 2	273.	1503.

S1= 0.2058821E 09

S2= 0.7385696E 08

S3= 0.6084819E 08

S4= 0.5455734E 08

S5= 0.4506278E 08

S6=TOTAL SUM OF SQUARES= 0.1608193E 09

S7=WITHIN-CELLS OF SQUARES= 0.1320253E 09

S8=ROWS SUMS OF SQUARES= 0.9494560E 07

S9=COLUMNS SUM OF SQUARES= 0.1578541E 08

S10=INTERACTION SUM OF SQUARES= 0.3514112E 07

MSWC= 0.3667368E 07

PROBLEM F(1,4(L-1))= 0.2588930E 01

LANGUAGE F(1,4(L-1))= 0.4304288E 01

4(L-1)= 0.3600000E 02

SET#2

RAW DATA

CCL.1=GROUP NUMBER CCL.2=STUDENT NUMBER WITHIN GROUP
 CCL.3=K-PROBLEM NUMBER CCL.4=K-PROGRAMMING TIME CCL.5=K-DEBUG TIME
 CCL.6=F-PROBLEM NUMBER CCL.7=F-PROGRAMMING TIME CCL.8=F-KEYPUNCH TIME
 CCL.9=NUMBER OF DEBUG RUNS CCL.10=F-DEBUG TIME CCL.11=DEBUG KEYPUNCH TIME
 CCL.12=WAIT TIME FOR RESULTS

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
1	1	2	380.	0.	1	5100.	1800.	1	1200.	0.	2700.
1	2	2	50.	0.	1	70.	165.	0	0.	0.	2400.
1	3	2	60.	0.	1	1200.	1200.	2	600.	300.	1800.
1	4	2	300.	0.	1	300.	420.	0	0.	0.	1200.
1	5	2	10.	0.	1	600.	300.	1	0.	0.	900.
1	6	2	55.	0.	1	1200.	900.	0	0.	0.	900.
1	7	2	300.	0.	1	2700.	9191.	1	1350.	300.	1300.
1	8	2	600.	0.	1	960.	300.	0	0.	0.	900.
1	9	2	300.	180.	1	600.	300.	0	0.	0.	1800.
1	10	2	900.	180.	1	600.	1200.	2	3600.	1800.	1800.
1	11	2	200.	180.	1	1200.	600.	2	900.	300.	1800.
1	12	2	240.	120.	1	600.	600.	2	5400.	180.	7200.
1	13	2	300.	300.	1	1800.	900.	1	300.	180.	600.
1	14	2	180.	120.	1	600.	900.	4	2700.	1200.	16200.
1	15	2	2700.	300.	1	9000.	1200.	3	6300.	900.	5100.
1	16	2	1800.	35.	1	2700.	900.	2	6300.	900.	18000.
1	17	2	300.	300.	1	3600.	600.	3	1800.	600.	16200.
2	1	1	180.	0.	2	2700.	2700.	2	1200.	600.	7200.
2	2	1	60.	0.	2	600.	900.	0	0.	0.	1500.
2	3	1	50.	0.	2	90.	300.	1	120.	60.	4500.
2	4	1	180.	60.	2	420.	300.	0	0.	0.	600.
2	5	1	1200.	300.	2	600.	600.	91	1200.	9191.	1200.
2	6	1	900.	600.	2	1200.	600.	1	240.	60.	2100.
2	7	1	120.	60.	2	180.	600.	2	600.	1200.	5400.
2	8	1	620.	185.	2	300.	360.	3	120.	60.	1800.
2	9	1	1500.	60.	2	900.	600.	1	300.	0.	1800.
2	10	1	120.	5.	2	300.	300.	2	1200.	300.	3600.
2	11	1	130.	300.	2	180.	600.	1	60.	60.	2700.
2	12	1	900.	150.	2	600.	420.	0	0.	0.	720.
2	13	1	300.	30.	2	900.	600.	1	300.	120.	2700.
2	14	1	315.	50.	2	600.	900.	2	300.	60.	900.
2	15	1	300.	0.	2	900.	1200.	1	60.	120.	720.
2	16	1	120.	60.	2	900.	900.	1	60.	30.	1800.
2	17	1	660.	360.	2	900.	1200.	1	300.	180.	3900.

NUMBER OF STUDENTS= 34
 DEVIDED EQUALLY INTO 2 GROUPS

ALL TIMES ARE IN SECONDS

THE DATA ELEMENT=9191. OR 91 SIGNIFIES THAT NO ACTUAL DATA SUPPLIED

TOTAL PROGRAMMING TIME (INCLUDES DEBUG TIME)

PROBLEM 1	K-LANGUAGE	F-LANGUAGE
	180.	6300.
	60.	70.
	50.	1800.
	240.	300.
	1500.	600.
	1500.	1200.
	130.	4050.
	805.	960.
	1560.	600.
	125.	4200.
	430.	2100.
	1050.	6000.
	330.	2100.
	365.	3300.
	300.	15300.
	180.	9000.
	1020.	5400.

PROBLEM 2	K-LANGUAGE	F-LANGUAGE
	380.	3900.
	50.	600.
	60.	210.
	300.	420.
	10.	1800.
	55.	1440.
	300.	780.
	600.	420.
	480.	1200.
	1080.	1500.
	380.	240.
	360.	600.
	600.	1200.
	300.	900.
	3000.	960.
	1835.	960.
	600.	1200.

MEAN PROGRAMMING TIME (TOTAL) OF EACH PROBLEM-LANGUAGE COMBINATION

	K-LANGUAGE	F-LANGUAGE
PROBLEM 1	581.	3722.
PROBLEM 2	611.	1078.

STANDARD DEVIATION (PROG. TIME) OF EACH PROBLEM LANGUAGE COMBINATION

	K-LANGUAGE	F-LANGUAGE
PROBLEM 1	525.	3793.
PROBLEM 2	735.	832.
S1=	0.5375962E 09	
S2=	0.2674006E 09	
S3=	0.2079664E 09	
S4=	0.1816614E 09	
S5=	0.1526252E 09	
S6=TOTAL SUM OF SQUARES=	0.3849707E 09	
S7=WITHIN-CELLS OF SQUARES=	0.2701955E 09	
S8=ROWS SUMS OF SQUARES=	0.2903624E 08	
S9=COLUMNS SUM OF SQUARES=	0.5534125E 08	
S10=INTERACTION SUM OF SQUARES=	0.3039792E 08	
MSWC=	0.4221904E 07	
PROBLEM F(1,4(IL-1))=	0.6977635E 01	
LANGUAGE F(1,4(IL-1))=	0.1310844E 02	
4(IL-1)=	0.6400000E 02	

SET#2.(WEIGHTED DEBUG TIME, +50% INCORRECT PROGRAM, +20% MINOR ERROR)

RAW DATA

COL.1=GROUP NUMBER COL.2=STUDENT NUMBER WITHIN GROUP
COL.3=K-PROBLEM NUMBER COL.4=K-PROGRAMMING TIME COL.5=K-DEBUG TIME
COL.6=F-PROBLEM NUMBER COL.7=F-PROGRAMMING TIME COL.8=F-KEYPUNCH TIME
COL.9=NUMBER OF DEBUG RUNS COL.10=F-DEBUG TIME COL.11=DEBUG KEYPUNCH TIME
COL.12=WAIT TIME FOR RESULTS

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)
1	1	2	380.	0.	1	5100.	1800.	1	1200.	0.	2700.
1	2	2	50.	0.	1	70.	165.	0	0.	0.	2400.
1	3	2	60.	0.	1	1200.	1200.	2	600.	300.	1300.
1	4	2	300.	0.	1	300.	420.	0	0.	0.	1200.
1	5	2	10.	0.	1	600.	300.	1	0.	0.	900.
1	6	2	55.	0.	1	1200.	900.	0	600.	0.	900.
1	7	2	300.	0.	1	2700.	9191.	1	1350.	300.	1800.
1	8	2	600.	0.	1	960.	300.	0	0.	0.	900.
1	9	2	300.	180.	1	600.	300.	0	0.	0.	1800.
1	10	2	900.	180.	1	600.	1200.	2	3900.	1800.	1300.
1	11	2	200.	180.	1	1200.	600.	2	1500.	300.	1300.
1	12	2	240.	120.	1	600.	600.	2	5400.	180.	7200.
1	13	2	300.	300.	1	1800.	900.	1	1200.	180.	600.
1	14	2	180.	120.	1	600.	900.	4	3000.	1200.	16200.
1	15	2	2700.	300.	1	9000.	1200.	3	10800.	900.	5100.
1	16	2	1800.	35.	1	2700.	900.	2	6300.	900.	18000.
1	17	2	300.	300.	1	3600.	600.	3	1800.	600.	16200.
2	1	1	180.	0.	2	2700.	2700.	2	1200.	600.	7200.
2	2	1	60.	0.	2	600.	900.	0	0.	0.	1500.
2	3	1	50.	0.	2	90.	300.	1	120.	60.	4500.
2	4	1	180.	60.	2	420.	300.	0	0.	0.	600.
2	5	1	1200.	300.	2	600.	600.	91	1200.	9191.	1200.
2	6	1	900.	600.	2	1200.	600.	1	240.	60.	2100.
2	7	1	120.	60.	2	180.	600.	2	600.	1200.	5400.
2	8	1	620.	185.	2	300.	360.	3	120.	60.	1800.
2	9	1	1500.	60.	2	900.	600.	1	300.	0.	1800.
2	10	1	120.	5.	2	300.	300.	2	1200.	300.	3600.
2	11	1	130.	300.	2	180.	600.	1	60.	60.	2700.
2	12	1	900.	150.	2	600.	420.	0	0.	0.	720.
2	13	1	300.	30.	2	900.	600.	1	300.	120.	2700.
2	14	1	315.	50.	2	600.	900.	2	300.	60.	900.
2	15	1	300.	0.	2	900.	1200.	1	60.	120.	720.
2	16	1	120.	60.	2	900.	900.	1	60.	30.	1800.
2	17	1	660.	360.	2	900.	1200.	1	300.	180.	3900.

NUMBER OF STUDENTS= 34
DEVIDED EQUALLY INTO 2 GROUPS

ALL TIMES ARE IN SECONDS

THE DATA ELEMENT=9191. OR 91 SIGNIFIES THAT NO ACTUAL DATA SUPPLIED

TOTAL PROGRAMMING TIME (INCLUDES DEBUG TIME)

PROBLEM 1	K-LANGUAGE	F-LANGUAGE
	180.	6300.
	60.	70.
	50.	1900.
	240.	300.
	1500.	600.
	1500.	1300.
	180.	4050.
	305.	900.
	1560.	600.
	125.	4500.
	430.	2700.
	1050.	6000.
	330.	3000.
	365.	3600.
	300.	19800.
	180.	9000.
	1020.	5400.

PROBLEM 2	K-LANGUAGE	F-LANGUAGE
	380.	3900.
	50.	600.
	60.	210.
	300.	420.
	10.	1800.
	55.	1440.
	300.	780.
	600.	420.
	480.	1200.
	1080.	1500.
	380.	240.
	360.	600.
	600.	1200.
	300.	900.
	3000.	960.
	1835.	960.
	600.	1200.

MEAN PROGRAMMING TIME (TOTAL) OF EACH PROBLEM-LANGUAGE COMBINATION

	K-LANGUAGE	F-LANGUAGE
PROBLEM 1	581.	4146.
PROBLEM 2	611.	1078.

STANDARD DEVIATION (PROG. TIME) OF EACH PROBLEM LANGUAGE COMBINATION

	K-LANGUAGE	F-LANGUAGE
PROBLEM 1	525.	4601.
PROBLEM 2	735.	832.
S1=	0.7094961E 09	
S2=	0.3240517E 09	
S3=	0.2440553E 09	
S4=	0.2141693E 09	
S5=	0.1749611E 09	
S6=TOTAL SUM OF SQUARES=	0.5345349E 09	
S7=WITHIN-CELLS OF SQUARES=	0.3954443E 09	
S8=ROWS SUMS OF SQUARES=	0.3920829E 08	
S9=COLUMNS SUM OF SQUARES=	0.6909427E 08	
S10=INTERACTION SUM OF SQUARES=	0.4073310E 08	
MSWC=	0.6022568E 07	
PROBLEM F(1,4(I-1))=	0.6510227E 01	
LANGUAGE F(1,4(I-1))=	0.1147256E 02	
4(I-1)=	0.6400000E 02	

APPENDIX C
REFERENCE MANUAL

APPENDIX C

The following two pages are the two sides of the K-M reference card given to students during the initial lecture

C2

APPENDIX D
McCracken Problems
vs.
Corresponding K-M Programs

- combine subprograms II.
 gives passing adjustable dimension
 through a subprogram.
14. Given single variables A, B, X, and L, write
 a SUBROUTINE subprogram to compute
 S, and T from

$$R = \sqrt{A + BX + X^L}$$

$$S = \cos(2\pi X + A) \cdot e^{BX}$$

$$T = \left(\frac{A + BX}{2}\right)^{L+1} - \left(\frac{A + BX}{2}\right)^{L-1}$$

15. Identify any errors in the following:

a. `COM B(2, 10)`

PROBLEM 14, PAGE 195

SUBROUTINE RST.

$$R = \sqrt{A + BX + X^L},$$

$$S = \cos(2\pi X + A) e^{BX}, \text{ AND}$$

$$T = \left\{ \frac{A + BX}{2} \right\}^{L+1} - \left\{ \frac{A + BX}{2} \right\}^{L-1}.$$

RETURN.

$$X2 = \frac{-\sqrt{b^2 - 4ac}}{2a}$$

WRITE: a, b, c, X1, X2

✓ 5. READ: a, b, c, x
Evaluate:

$$r = \frac{b \cdot c}{12} \left[6x^2 \left(1 - \frac{x}{a} \right) + b^2 \left(1 - \frac{x}{a} \right)^3 \right]$$

WRITE: a, b, c, x, r

*6. READ: a, e, h, p
Evaluate:

$$x = \frac{h \cdot p}{a}$$

(sc
actu
grar
sol
act
unc
runni
is encc
to w
stre
to

PROBLEM 5, PAGE 19

READ A,B,C,X.

$$R = \frac{BC}{12} \left[6X^2 \left\{ 1 - \frac{X}{A} \right\} + B^2 \left\{ 1 - \frac{X}{A} \right\}^3 \right]$$

PRINT A,B,C,X,R. FINISH.

10
th
de-
8. READ: ET, ES, RG, ROPT, RIN
Evaluate:

$$F = \frac{1}{1 + \left(\frac{RG}{ROPT}\right)^2} \cdot \frac{1}{1 - \frac{\left(\frac{ET}{ES}\right)^2 \left(1 + \frac{RG}{RIN}\right)^2}$$

WRITE: ET, ES, RG, ROPT, RIN, and F
9. Add appropriate READ, WRITE, and FOR-
AT statements to the segments
wrote for Exercise 5.

PROBLEM 8, PAGE 19

SPECIAL VARIABLES ET,ES,RG,ROPT,RIN.
READ ET,ES,RG,ROPT,RIN.

$$F = \frac{1}{1 + \left(\frac{RG}{ROPT}\right)^2} \cdot \frac{1}{1 - \frac{\left(\frac{ET}{ES}\right)^2 \left(1 + \frac{RG}{RIN}\right)^2}$$

PRINT ET,ES,RG,ROPT,RIN,F. FINISH.

- (c) Replace each element in the third row by the sum of the corresponding elements from the first and second rows, using a loop.
4. A two-dimensional array named XYZ3 contains four rows and three columns. Write separate program segments to accomplish the following:
- Replace all the elements in the fourth row by zeros.
 - If the product of the first element in the first row, the second element in the second row, and the third element in the third row is less than 10^{-5} in absolute value, place a zero in DET.
 - Replace each element in the second column by the average of the corresponding elements in the first and third columns.
- *5. A three-dimensional array named PUPILS contains information about the pupil population of a certain school district, organized as follows: first subscript distinguishes the school; second subscript distinguishes the sex; third subscript distinguishes the grade. For example, PUPILS(1,1,1) would be the number of first graders in the first school district who are girls.

PROBLEM 4, PAGE 89

DIMENSION XYZ3=(4,3),DET=1.

FROM j=1 TO 3 XYZ3_{4,j} = 0 .

IF $|(XYZ3_{1,1})(XYZ3_{2,2})(XYZ3_{3,3})| < 10^{-5}$

THEN DET=0 .

FROM i=1 TO 4 $XYZ3_{1,2} = \frac{XYZ3_{1,1} + XYZ3_{1,3}}{2}$.

FINISH.

ed L

$Dx_1 \dots - X(I)$
1 ... 49

Write a program segment to perform this calculation.

13. Suppose we have a one-dimensional array named Y that contains 32 elements; these are to be regarded as the 32 ordinates of an experimental curve at equally spaced abscissas. Assuming that a value has already been given to H, compute the integral of the curve represented approximately by the Y values from

these three
ordinates
sional

$$TRAP = \frac{H}{2} (Y_1 + 2Y_2 + 2Y_3 + \dots + 2Y_{31} + Y_{32})$$

both
to
two-
the fol-

14. A two-dimensional array named AMATR

4G

PROBLEM 13, PAGE 90

SPECIAL VARIABLE TRAP.

$$TRAP = \frac{H}{2} (Y_1 + 2 \sum_{i=2}^{31} Y_i + Y_{32}) \cdot FINISH.$$

- corresponding to
through the four given p
ment of the differences is some
ent from that in Stirling's formula, however.
- *17. Given two one-dimensional arrays named A and B, of seven elements each, suppose that the seven elements of A are punched on one card and the seven elements of B are punched on another card. Each element value is punched in 10 columns in a form suitable for reading with an F10.0
25. Rete
that
sion
num
Cha
num
- field descriptor. Write a program to read the cards, then compute and print the value ANORM from
- $$ANORM = \sqrt{\sum_{i=1}^7 a_i b_i}$$
- Use a 1PE20.7 field specification for ANORM.
18. Using the assumptions of Exercise 17, write a program to read the data cards and then carry out the following procedure. If every $a_i > b_i$, for $i = 1, 2, \dots, 7$, print an integer 1; if this condition is not satisfied, print a zero.
- *19. Rewrite the program segment for Exercise 11 to use double precision variables.
20. Rewrite the program segment for Exercise 16 to use double precision variables.

PROBLEM 18, PAGE 91 (ALT. INTERP.)

READ A₁ FROM 1=1 TO 7.

READ B₁ FROM 1=1 TO 7.

X=1. FROM 1=1 TO 7 IF A₁ ≤ B₁ THEN

X=0. PRINT X {1}. FINISH.

49 elements from

$$DX(I) = X(I+1) - X(I)$$

$$I = 1, 2, \dots, 49$$

name
the ele
YS.
This
etc

3. A two-dimensional array named AMATR contains 10 rows and 10 columns. A one-dimensional array named DIAG contains 10 elements. Write a program segment to compute the elements of DIAG from

$$DIAG(I) = AMATR(I, I)$$

$$I = 1, 2, \dots, 10$$

*10. A
rows
array
15 e
a

A one-dimensional array named M contains 20 integers. Write a program segment using a loop statement to recompute the elements of M by multiplying each element by 2.

PROBLEM 3, PAGE 115

```
DIMENSION AMATR(10,10), DIAG(10).

FOR I=1,2,...,10  DIAG(I)=AMATR(I,I).

FINISH.
```

named X
 program
 compute
 named

9. Two one-dimensional arrays named X and Y contain 50 elements each. A variable named XS is known to be equal to one of the elements in X. If $XS = X_i$, place Y_i in YS.

This kind of table search has a wide variety of applications, such as finding a value in a table of electric utility rates from a rate code or finding the numerical code corresponding to an alphabetic name.

*10. A two-dimensional array A contains 15 rows and 5 columns. A one-dimensional array B contains 15 elements.

ATR
 one-
 .tains
 ent to

PROBLEM 9, PAGE 115

DIMENSION X=50, Y=50, XS=1, YS=1.

FROM 1=1 TO 50 IF $XS=X_1$ THEN $YS=Y_1$.

FINISH.

A_j

and
 10er of
 en by
 eger
 nts
 10

1. be viewed as multiplication of a matrix and a vector.

11. Three two-dimensional arrays A, B, and C have 15 rows and 15 columns each. Given the arrays A and B, compute the elements of C from

$$C_{ij} = \sum_{k=1}^{15} A_{ik} B_{kj} \quad i, j = 1, 2, \dots, 15$$

This is matrix multiplication.

12. A two-dimensional array named RST has 20 rows and 20 columns. Compute the product of the main diagonal of RST and the DPRO array.

ave
 the

PROBLEM 11, PAGE 115

DIMENSION A=(15,15), B=(15,15), C=(15,15).

$$C_{1,j} = \sum_{k=1}^{15} A_{1k} B_{kj} \quad \text{FOR } j=1,2,\dots,15$$

AND j=1,2,...,15. FINISH.

$$\sqrt{Y^4 + \sqrt{Y}}$$

$$V =$$

$$\frac{\sin^2 Y + \sqrt{1 + \sqrt{Y}}}{\sqrt{1 + \sqrt{Y}}} + 3 \sin^2 Y$$

Define a statement function to compute

$$SLG(A) = 2.549 \log \left(A + A^2 + \frac{1}{A} \right)$$

Then use the function to compute

$$R = X + \log X + 2.549 \log \left(A + A^2 + \frac{1}{A} \right)$$

$$S = \cos X + 2.549 \log \left(1 + X + (1 + X)^2 + \frac{1}{1 + X} \right)$$

$$T = 2.549 \log \left[(A - B)^3 + (A - B)^6 + \frac{1}{(A - B)^3} \right]$$

$$U = [B(I) + 6]^2 + 2.549 \log \left[\frac{1}{B(I)} + \frac{1}{B(I)^2} + B(I) \right]$$

7. W_i
p_i

of
A, C
8. R_p

- *3. Define a logical statement function to compute the "exclusive or" of two logical variables. The exclusive or of the inputs

Problem 2, page 194

$$\text{FUNCTION } SLG(A) = 2.549 \text{ LOG}(A + A^2 + \frac{1}{A})$$

$$R = X + \text{LOG}(X) + SLG(A).$$

$$S = \text{COS}(X) + SLG(1 + X).$$

$$T = SLG \left[(A - B)^3 \right].$$

$$U = [B_1 + 6]^2 + SLG\left(\frac{1}{B_1}\right).$$

FINISH.

the point
nd XIMAG lie.
de $\sqrt{2}$ with its cor-
nate axes.

aten.
arger, place
nd then com-
nd the largest

2. In the following exercises you are to draw a flowchart and write a complete program, including input and output. You may use F10.0 field specifications for all input and 1PE15.6 for all output.

be posi-
hat is

n to be
Subtract
as neces-
ess than 2π ;
HETA.

set SIGNS
itive, set
t signs.

of three
ocal maxi
id $Y2 > Y3$;
o, otherwise

= 0, set
herwise

statement
er to state-
o statement

atement 250;

* (a) Read the value of ANNERN; print ANNERN and compute and print TAX according to the following table:

ANNERN (annual earnings)	TAX
Less than \$2000	Zero
\$2000 or more but less than \$5000	2% of the amount over \$2000
\$5000 or more	\$60 plus 5% of the amount over \$5000

(b) GROSS is an employee's earnings for the year; DEPEND is the number of dependents he claims. Multiply DEPEND by 675.00, subtract the product from GROSS, and place the difference in TAXABL. However, if this difference is negative, place zero in TAXABL.

TRANSFER OF CONTROL
49

PROBLEM 2B, PAGE 49

SPECIAL VARIABLES GROSS, DEPEND, TAXABLE.

TAXABLE=GROSS-(675)DEPEND.

IF TAXABLE<0 THEN TAXABLE=0.

FINISH.

D11

get out.
 solve this g an
 variable run it. convert to re
 then divide by the result as the
 independent variable.

(e) Y is to be computed as a function of X
 according to the formula

$$Y = \sqrt{1+X} + \frac{\cos 2X}{1+\sqrt{X}}$$

for a number of equally spaced values of
 X. Three numbers are to be read from a
 card: XINIT, XINC, and XFIN. XINIT, we
 assume, is less than XFIN; XINC is posi-
 tive. Y is to be computed and printed
 initially for X = XINIT. Then X is to be
 incremented by XINC, and Y is to be
 computed and printed for this new value
 of X, and so on, until Y has been com-
 puted for the largest value of X not ex-
 ceeding XFIN. (The phrase "the largest
 value of X not exceeding XFIN" lets us
 ignore the problem presented in the last
 two exercises. However, this formulation
 does mean that if the data is set up with
 the intention of terminating the process
 with X exactly equal to XFIN it may not
 do so.)

S.
 squ
 LIN
 yo

B:

(C)
 .
 h

1. In the following exercises the emphasis is on
 trying to devise decision processes . . .
 than on computations. Draw a flow

A GUIDE TO PROGRAMMING

PROBLEM 2E, PAGE 50

SPECIAL VARIABLES XINIT, XINC, XFIN .

READ XINIT, XINC, XFIN.

FROM X=XINIT BY XINC UNTIL X>XFIN

PRINT Y= $\sqrt{1+X} + \frac{\cos(2X)}{1+\sqrt{X}}$.

FINISH.

APPENDIX E
K-M Examples
(Programs as Compiled and Executed)

APPENDIX E

The following 20 pages are the K-M example set give to students during the initial lecture.

RM examples

To make this an executable program, you need only type additional statements to read in or compute, or assign values for n, β, x , etc. An answer will be printed out only if the integral converges for the parameters input. Else the appropriate message of non-convergence will be output.

$$\theta_{\alpha\beta} = \frac{\int_0^{\sqrt{y}} \int_0^{\sqrt{y}} \frac{e^{-\alpha x}}{y+n} \sum_{j,j=1}^n \left[\cos y + \frac{\sin^{1+j} x}{(1+j)x} \right] \frac{\log_2 x + \tan^{-1} \frac{x}{y}}{y^{n-1}} + \prod_{k=1}^{n^2-\alpha^2} \left[\frac{\beta^{k+1}}{k} + (x+ky)^{-k} \right] dx dy .$$

* → FROM I=² BY 2.5 UNTIL I > 60 PRINT I².3 .

FINISH.

90010 FROM I=2 BY 2.5 UNTIL I GREATER THAN OR EQUAL TO 60 PRINT I².
2)
90020 FINISH

97777 X14=3.14159265
X15=8.7182818
FORMAT (E14.8)
X57=2.
Q1=(Q=2.5)/ABS(F(Q))
GOTO 90001
90002 X57=X57+2.5
90001 IF((X57-(60.)))90004,90003,90003
90004 P1=X57
90005 FORMAT (F10.2,4X)
WRITE 2,90005,P1
GOTO 90002
90003 CONTINUE
END

* Note: error in typing
corrected by backspacing
and overtyping with
correct character

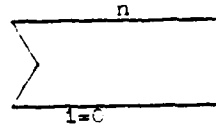
2.00
4.50
7.00
9.50
12.00
14.50
17.00
19.50
22.00
24.50
27.00
29.50
32.00
34.50
37.00
39.50
42.00
44.50
47.00
49.50
52.00
54.50
57.00
59.50

note termination point

MAXIMUM N=25. *note that max min without a fixed end program will not know the value of n except at run time.*

READ n.

$A_i = 1$ FROM $i=0$ TO n. READ X. $P =$



$A_i X^i$

PRINT X,P. FINISH.

S0010 MAXIMUM N=25

S0020 READ N

S0030 A SUB (I)=1 FROM I=0 TO N

S0040 READ X

S0050 P=SUM WITHIN (N,I=0) OF (A SUB (I)*X RAISED TO (I))

S0060 PRINT X,P

S0070 FINISH

```

          DIMENSION X21(6026)
          X14=3.14159265
          X15=2.7182818
07777    FORMAT (E14.8)
          READ 1,97777,X72
          .57=0.
          Z1=(0+1.)/ABS7(0)
          GOTO 90001
          X57=X57+1.
          90002  IF((X72-X57)*01)90003,90004,90004
          90001  X21(I1=X57+1.)=X57
          GOTO 90002
          90003  READ 1,97777,X67
          X47= SUM(X57=0., SUM7(X57,X72),U=X21(I1=X57+1.)*X67*(X57))
          P1=X47
          P2=X47
          WRITE 2,97777,P1-P2
          END
    
```

Note that there is an input buffer which stores values for n and X

INPUT 12 12
 .12000000+02 .11583561+15
 ↑ ↑
 x p

$\int_0^x y \, dx$ TO COMPUTE $z = \int_0^x y \, dx$ AND PRINT FORMAT 1, Y, Z.

FORMAT 1 THE INTEGRAL FROM 0 TO X IS XXX.XXX APPROXIMATELY . FINISH.

90010 FROM Y=1 TO 6 COMPUTE Z=INTEGRAL WITHIN (Y,X=0) OF (X) AND
 PRINT FORMAT 1,Y,Z

90020 FORMAT 1 THE INTEGRAL FROM 0 TO X IS XXX.XXX APPROXIMATELY 1
 90030 FINISH

Note: in typing integrals
 the integrand should be
 outside the $\int$

97777 X14=3.14159265
 X15=2.7182818
 FORMAT (E14.8)
 X101=1.
 Q1=(Q=1.)/ABSF(Q)
 GOTO 90001
 90002 X101=X101+1.
 90001 IF((6.-X101)*Q1)90003,90004,90004
 90004 X102=XINT(X100=0.,XINT1(X100,X101),W=X100)
 P1=X101
 P2=X102
 WRITE 2,60001,P1,P2
 GOTO 90002
 90003 CONTINUE
 60001 FORMAT (0024H THE INTEGRAL FROM 0 TO ,11.0004H IS ,F7.3,0017H 4
 APPROXIMATELY)
 END

but the upper
 + lower limits within
 the symbol are

THE INTEGRAL FROM 0 TO 1 IS	.500	APPROXIMATELY
THE INTEGRAL FROM 0 TO 2 IS	2.000	APPROXIMATELY
THE INTEGRAL FROM 0 TO 3 IS	4.500	APPROXIMATELY
THE INTEGRAL FROM 0 TO 4 IS	8.000	APPROXIMATELY
THE INTEGRAL FROM 0 TO 5 IS	12.500	APPROXIMATELY
THE INTEGRAL FROM 0 TO 6 IS	18.000	APPROXIMATELY

ATOM=5. PRINT ATOM. FINISH.

ERROR; SHOULD HAVE Declared. SPECIAL VARIABLES ATOM.
in DIMENSION ATOM = 1

S0010 A*T*O*M=5
S0020 PRINT A*T*O*M
S0030 FINISH

```

X14=3.14159265
X15=2.7182818
97777  FORMAT (F14.8)
X21=X63*X46*X44=5.
P1=X21*X63*X46*X44
WR11E 2.97777,P1
END

```

XX

FOR X = -5 BY .1 TO 3 COMPUTE $Q = \frac{10}{4} C_1 X^4$ AND IF $|Q| \leq .5$ THEN PRINT X, Q.

FINISH

50010 READ C SUP (1) FROM 121 TO 10 **EXECUTION 1 OF 2**

50020 FROM X=-5 BY .1 TO 3 COMPUTE Q=SUM:WITHIN (10,I=1) OF(C: SUP (1)*X RAISED TO (1)) AND IF ABS(Q) LESS THAN OR EQUAL TO .5 THEN
PRINT X,Q

50030 FINISH

```

          DIMENSION X23(10011)
          X14=3.14159265
          X15=2.7182818
97777    FORMAT (E14.8)
          X57=1.
          Q1=(Q=1.)/ABSF(Q)
          GOTO 90001
90002    X57=X57+1.
90003    IF((10.-X57)*Q1)90003,90004,90004
90004    READ 1,97777,X23(11=X57+1.)
          GOTO 90002
90005    X67=-5.
          Q2=(Q=.1)/ABSF(Q)
          GOTO 90005
90006    X67=X67+.1
90007    IF((3.-X67)*Q2)90007,90010,90010
90010    X50= SUM(X57=1., SUM1(X57,10.),W=X23(11=X57+1.)*X67*(X57))
          IF(ABSF(X50) -(R1=(.5)))90011,90011,90012
90011    P1=X67
          P2=X50
          WRITE 2,97777,P1,P2
90012    GOTO 90006
90007    CONTINUE
          END

```

INPUT 2 7 3 9 1 6 7 5 55 51 ← input data typed in at program request

- .40000000-00	.35547382-00
- .30000000-00	.21859741-01
- .20000000-00	-.12963620-00
- .10000000-00	-.13210451-00
- .23719621-08	-.47439244-08
- .99999997-01	.27391680-00

X
 Q

```

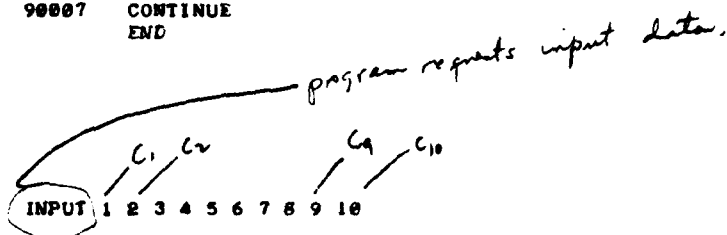
50010 READ C SUB (I) FROM I=1 TO 10 EXECUTION 2 OF 2
50020 FROM X=-5 BY .1 TO 3 COMPUTE Q=SUM WITHIN (10,I=1) OF (C SUB(-
I)*X RAISED TO (I)) AND IF ABS(Q) LESS THAN OR EQUAL TO .5 THEN
PRINT X,Q
50030 FINISH

```

```

          DIMENSION X23(1011)
          X14=3.14159265
          X15=2.7182818
97777    FORMAT (F14.8)
          X57=1.
          Q1=(Q=1.)/ABSF(Q)
          GOTO 90001
90002    X57=X57+1.
90003    IF((10.-X57)*Q1)90003,90004,90004
90004    READ 1,97777,X23(11=X57+1.)
          GOTO 90002
90005    X67=-5.
          Q2=(Q=-1)/ABSF(Q)
          GOTO 90005
90006    X67=X67+.1
90007    IF((3.-X67)*Q2)90007,90010,90010
90010    X50= SUM(X57=1., SUM1(X57,10.),W=X23(11=X57+1.)*X67*(X57))
          IF(ABS(X50) - (R1=.5))90011,90011,90012
90011    P1=X67
          P2=X50
          WRITE 2,97777,P1,P2
90012    GOTO 90006
90007    CONTINUE
          END

```



<u>X</u>	<u>Q</u>
-.60000000+00	.38971699-00
-.50000000+00	.28125000-00
-.40000000-00	.19623936-00
-.30000000-00	.12249988-00
-.20000000-00	.61111211-01
-.10000000-00	.17355371-01
-.23719621-00	.11252409-16
.99999997-01	.23456788-01
.19999999-00	.11249945-00
.29999999-00	.31221596-00

← Note.

FROM X=3 BY 0.5 TO 13 PRINT Q=3X²+7X³-19 . FINISH.

S0010 FROM X=3 BY 0.5 TO 13 PRINT Q=3*X RAISED TO (2)+7*X
RAISED TO (3)-19
S0020 FINISH

```

X14=3.14159265
X15=2.7182818
97777 FORMAT (F14.8)
Xc7=3.
Q1=(Q-0.5)/ABSF(Q)
G010 90001
90002 X67=X67+0.5
90001 IF((13.-X67)*Q1)90003,90004,90004
90004 P1=X50=3.*X67*(2.)+7.*X67*(3.)-19.
WRITE 2,97777,P1
GOTO 90002
90003 CONTINUE
END

```

.19699999+03
.31787499+03
.47699999+03
.67962499+03
.93099999+03
.12363749+04
.16009999+04
~~.20301209+04~~
.25289999+04
.31028749+04
.37569999+04
.44966249+04
.53269999+04
.62533749+04
.72809999+04
.84151249+04
.96609999+04
.11023874+05
.12508999+05
.14121624+05
.15866999+05

Output

MAXIMUM $n=30$.

READ n .

$A_1=1$ FROM $i=0$ TO n . READ x .

$$P(x) = \sum_{i=0}^n A_i x^i.$$

PRINT P , n . PRINT FORMAT 1, n, x, P .

FORMAT 1 THE POLYNOMIAL OF DEGREE x , ARGUMENT xx, xx , = xxx, xxx .

FROM $i=0$ TO n PRINT 1, A_i .

FINISH.

S0010 MAXIMUM N=30

S0020 REAL N

S0030 A SUB (I)=1 FROM I=0 TO N

S0040 READ X

S0050 P=SUM WITHIN (N, I=0) OF (A SUB (I)*X**I)

S0060 PRINT P, N

S0070 PRINT FORMAT 1, N, X, P

S0080 FORMAT 1 THE POLYNOMIAL OF DEGREE x , ARGUMENT xx, xx , = xxx, xxx

S0090 FROM I=0 TO N PRINT 1, A SUB (I)

S0100 FINISH

```

        DIMENSION X21(10031)
        X14=3.14159265
        X15=2.7182818
97777  FORMAT (E14.8)
        READ 1,97777,X72
        X57=0.
        Q1=(Q=1.)/ABSF(Q)
        GOTO 90001
9 0002  X57=X57+1.
9 0001  IF((X72-X57)*Q1)90003,90004,90004
90004  X21(I1=X57+1.)=X57
        GOTO 90002
9 0003  READ 1,97777,X67
        X47= SUM(X57=0., SUM1(X57,X72), W=X21(I1=X57+1.)*X67*(X57))
        P1=X47
        P2=X72
        WRITE 2,97777,P1,P2
        P1=X72
        P2=X67
        P3=X47
        WRITE 2,60001,P1,P2,P3
6 0001  FORMAT (0025H THE POLYNOMIAL OF DEGREE ,I1,0012H , ARGUMENT ,F5.
        2,0005H , = ,F7.3,0002H )
        X57=0.
        Q2=(Q=1.)/ABSF(Q)
        GOTO 90005
9 0006  X57=X57+1.
9 0005  IF((X72-X57)*Q2)90007,90010,90010
90010  P1=X57
        P2=X21(I1=X57+1.)
90011  FORMAT (18.6X,18.6X)
        WRITE 2,90011,P1,P2
        GOTO 90006
9 0007  CONTINUE
        END

```

INPUT 5 5

```

-18554999+85 .50000000+01
THE POLYNOMIAL OF DEGREE 5 , ARGUMENT 5.001. = 1.185+05
0 0
1 1
2 2
3 3
4 4
5 5

```

MAXIMUM $n=20$.

PRINT $X_1=1$ FROM $i=1$ TO 10. {LENGTH OF X IS KNOWN }

READ n . FROM $i=1$ TO n PRINT $Y_1=1$. {THE LENGTH OF Y IS FIXED BY n }

PRINT $Z = j^2$ FROM $j=1$ UNTIL $Z \geq 94$. { STOP WHEN $Z \geq 94$ }

PRINT n .
FINISH.

S0010 MAXIMUM N=20

S0020 PRINT X SUB (I)=1 FROM I=1 TO 10

S0030

READ N

S0040 FROM I=1 TO N PRINT X SUB (I)=1

S0050

PRINT Z=J RAISED TO (2) FROM J=1 UNTIL Z GREATER THAN OR EQUAL TO 94

S0060

PRINT N

S0070 FINISH

Execution on next page

```

      DIMENSION X67(8011),X78(8021)
      X14=3.14159265
      X15=2.7182818
97777  FORMAT (E14.8)
      X57=1.
      Q1=(Q=1.)/ABSF(Q)
      GOTO 98001
98002  X57=X57+1.
98001  IF((10.-X57)*Q1)98003,98004,98004
98004  P1=X67(I1=X57+1.)=X57
      WRITE 2,97777,P1
      GOTO 98002
98003  READ 1,97777,X72
      X57=1.
      Q2=(Q=1.)/ABSF(Q)
      GOTO 98005
98006  X57=X57+1.
98005  IF((X72-X57)*Q2)98007,98010,98010
98010  P1=X78(I1=X57+1.)=X57
      WRITE 2,97777,P1
      GOTO 98006
98007  X68=1.
      Q3=(Q=1.)/ABSF(Q)
      GOTO 98011
98012  X68=X68+1.
98011  IF((X71-(94.))98014,98013,98013
98014  P1=X71=X68*(2.)
      WRITE 2,97777,P1
      GOTO 98012
98013  P1=X72
      WRITE 2,97777,P1
      END

```

.10000000+01
 .20000000+01
 .30000000+01
 .40000000+01
 .50000000+01
 .60000000+01
 .70000000+01
 .80000000+01
 .90000000+01
 .10000000+02
 .11000000+02
 .12000000+02
 .13000000+02
 .14000000+02
 .15000000+02



.10000000+01
 .39999999+01
 .89999999+01
 .15999999+02
 .24999999+02
 .35999999+02
 .46999999+02
 .63999999+02
 .80999999+02
 .99999999+02
 .15000000+02



TYPING ERROR CORRECTED → WITH OVERTYPE

FOR I=1,2,...,10 PRINT I(2).
 FOR J=5(10)55 PRINT J(2).
 PRINT LABEL ALPHA,BETA,GAMMA,ZETA.
 FINISH.

S0010 FOR I=1,2,...,10 PRINT I(2)
 S0020 FOR J=5(10)55 PRINT J(2)
 S0030 PRINT LABEL ALPHA,BETA,GAMMA,ZETA
 S0040 FINISH

X14=3.14159265
 X15=2.7182818
 97777 FORMAT (E14.8)
 X57=1.
 Q1=(Q=2.-(1.))/ABS(F(Q))
 GOTO 90001
 90002 X57=X57+2.-(1.)
 90001 IF((10.-X57)*Q1)90003,90004,90004
 90004 P1=X57
 90005 FORMAT (18.6X)
 WRITE 2,90005,P1
 GOTO 90002
 90003 X60=5.
 Q2=(Q=10.)/ABS(F(Q))
 GOTO 90006
 90007 X60=X60+10.
 90006 IF((55.-X60)*Q2)90010,90011,90011
 90011 P1=X60
 90012 FORMAT (18.6X)
 WRITE 2,90012,P1
 GOTO 90007
 90010 CONTINUE
 90013 FORMAT (4X,0005HALPHA,5X,5X,0004HPETA,5X,4X,0005HGAMMA,5X,5X,00
 04HZETA)
 WRITE 2,90013
 END

OUTPUT

1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
5			
15			
25			
35			
45			
55			
ALPHA	BETA	GAMMA	ZETA

```

DO STATEMENT 5 FROM Y=1 TO 11.
PRINT Y (2) . STATEMENT 5. FINISH.

```

```

50010 LOOP STATEMENT 5 FROM Y=1 TO 11
50020 PRINT Y(2)
50030 STATEMENT 5
50040 FINISH

```

```

X14=3.14159265
X15=2.7182818
97777 FORMAT (E14.8)
X70=1.
Q1=(Q-1.)/ABSF(Q)
GOTO 90001
90002 X70=X70+1.
90001 IF((11.-X70)*Q1)90003,90004,90004
90004 P1=X70
90005 FORMAT (18.6X)
WRITE 2,90005,P1
GOTO 90002
90003 CONTINUE
70005 CONTINUE
END

```

```

1
2
3
4
5
6
7
8
9
10
11

```

PRINT Z=

$$\frac{\sin \theta}{\cos \theta}$$

$$\sqrt{\frac{1}{(\cos \theta)^2}}$$

FROM $\theta = .1\pi$ BY .05 TO $.45\pi$.

FINISH.

S0010 PRINT Z=((SIN(THETA))/(COS(THETA)))*SQRT(((1)/((COS(THETA))
RAISED TO (2))))

FROM THETA=.1*PI BY .05 TO .45*PI
S0020 FINISH

```

          X14=3.14159265
          X15=2.7182818
97777    FORMAT (E14.8)
          X33=.1*X14
          Q1=(Q=.05)/ABSF(Q)
          GOTO 90001
90002    X33=X33+.05
90003    IF((.45*X14-X33)*Q1)90003,90004,90004
90004    P1=X71=((SINF(X33))/(COSF(X33)))*SQRT(((1.)/((COSF(X33))^(2.))
          ))
          WRITE 2,97777,P1
          GOTO 90002
90003    CONTINUE
          END

```

.34164297-00
.40790803-00
.48018216-00
.55987554+00
.64870550+00
.74879297+00
.86879701+00
.99410211+00
.11470632+01
.13274886+01
.15430055+01
.18041134+01
.21254097+01
.25277090+01
.30416688+01
.37135437+01
.46164769+01
.58710427+01
.76879986+01
.10463164+02
.14993454+02
.23250644+02

OUTPUT

$R=15-\ln(e^{15})+1$ AND $D=9-\log(10^9)+3$.
 PRINT R,D. FINISH.

NOTE FACT THAT
 COMPUTER INTERPRETED
 $9-\log(10^9)+3$
 CORRECTLY!

S0010 R=15-LN (E RAISED TO (15))+1 AND D=9-LOG(10 RAISED TO (9))+3

S0020 PRINT R,D

S0030 FINISH

97777 X14=3.14159265
 X15=2.7182818
 FORMAT (E14.8)
 X51=15.-LOGF(X15*(15.))+1.
 X24=9.-CLOGF(10.*(9.))+3.
 P1=X51
 P2=X24
 WRITE 2,97777,P1,P2
 END

.10000001+01 .29999999+01

$P_1 = .99$ FROM $i=1$ TO 100.

P_1 IS THE RELIABILITY INDEX FOR COMPONENT i

$Q = \prod_{i=1}^{100} P_i$ (PRODUCT FUNCTION) PRINT Q.

$R = 100Q$.

R = THE TOTAL DEVICE RELIABILITY FOR 100 COMPONENTS

PRINT FORMAT 1, R.

FORMAT 1 THE DEVICE IS xx.xxxx PER CENT RELIABLE.

FINISH.

50010 P SUB (I) = .99 FROM I=1 TO 100

50020

Q = PRODUCT WITHIN (100, I=1) OF (P SUB (I))

50030 PRINT Q

50040 R = 100 * Q

← NOTE EXPLICIT MULTIPLICATION

50050

PRINT FORMAT 1, R

50060 FORMAT 1 THE DEVICE IS XX.XXXX PER CENT RELIABLE
50070 FINISH

```

          DIMENSION X47(0101)
          X14=3.14159265
          X15=2.7182818
97777    FORMAT (E14.8)
          X57=1.
          Q1=(Q=1.)/ABSF(Q)
          GO10 90001
90002    X57=X57+1.
90003    IF((100.-X57)*Q1)90003,90004,90004
90004    X47(I1=X57+1.)=.99
          GO10 90002
90003    X50= PROD(X57=1., PROD1(X57,100.),6=X47(I1=X57+1.))
          P1=X50
          WRITE 2,97777,P1
          X51=100.*X50
          P1=X51
          WRITE 2,60001,P1
60001    FORMAT (0015)THE DEVICE IS ,F7.4,0010H PER CENT(RELIABLE)
          END

```

.36603233-00
THE DEVICE IS .366002 PER CENT RELIABLE

FROM I=1 TO 11 AND K=99 TO 102 PRINT 1 {2} ,k {3} .

{NOTE THAT OUTER LOOP IS EXERCISED FIRST }

FINISH.

S0010 FROM I=1 TO 11 AND K=99 TO 102 PRINT I(2),K(3)

S0020
FINISH

```

X14=3.14159265
X15=2.7182818
97777  FORMAT (E14.8)
X61=99.
Q1=(Q=1.)/ABSF(Q)
GOTO 90001
90002  X61=X61+1.
90003  IF((102.-X61)+Q1)90003,90004,90004
90004  X57=1.
      Q2=(Q=1.)/ABSF(Q)
      GOTO 90005
90006  X57=X57+1.
90007  IF((11.-X57)+Q2)90007,90010,90010
90010  P1=X57
      P2=X61
90011  FORMAT (I6,6X,I6,6X)
      WRITE 8,90011,P1,P2
      GOTO 90006
90007  GOTO 90002
90003  CONTINUE
END

```

I		K		THEN THIS	
I	K	I	K	I	K
1	99	1	101		
2	99	2	101		
3	99	3	101		
4	99	4	101		
5	99	5	101		
6	99	6	101		
7	99	7	101		
8	99	8	101		
9	99	9	101		
10	99	10	101		
11	99	11	101		
1	100	1	102		
2	100	2	102		
3	100	3	102		
4	100	4	102		
5	100	5	102		
6	100	6	102		
7	100	7	102		
8	100	8	102		
9	100	9	102		
10	100	10	102		
11	100	11	102		

THIS WAS PRINTED
OUT FIRST

FROM N=1 BY .66 TO 20
 PRINT FORMAT 1 , N, TRUNCATE (N).

FORMAT 1 FOR N=XX.XXXX THE TRUNCATE IS XX.
 FINISH.

S0010 FROM N=1 BY .66 TO 20 PRINT FORMAT 1,N,TRUNCATE (N)

S0020 FORMAT 1 FOR N=XX.XXXX THE TRUNCATE IS XX
 S0030 FINISH

```

      X14=3.14159265
      X15=2.7182818
97777  FORMAT (E14.8)
      X45=1.
      Q1=(Q=-.66)/ABSF(Q)
      GOTO 98001
98002  X45=X45+.66
98001  IF((20.-X45)*Q1)98003,98004,98004
98004  PI=X45
      P2=XINTG(X45)
      WRITE 2,60001,PI,P2
      GOTO 98002
98003  CONTINUE
6 0001  FORMAT (0000HFOR N=,F7.4,0019H THE TRUNCATE IS,12)
      END

```

```

FOR N= 1.0000 THE TRUNCATE IS 1
FOR N= 1.6600 THE TRUNCATE IS 1
FOR N= 2.3200 THE TRUNCATE IS 2
FOR N= 2.9800 THE TRUNCATE IS 2
FOR N= 3.6400 THE TRUNCATE IS 3
FOR N= 4.3000 THE TRUNCATE IS 4
FOR N= 4.9600 THE TRUNCATE IS 4
FOR N= 5.6200 THE TRUNCATE IS 5
FOR N= 6.2800 THE TRUNCATE IS 6
FOR N= 6.9400 THE TRUNCATE IS 6
FOR N= 7.6000 THE TRUNCATE IS 7
FOR N= 8.2600 THE TRUNCATE IS 8
FOR N= 8.9200 THE TRUNCATE IS 8
FOR N= 9.5800 THE TRUNCATE IS 9
FOR N= -1024+02 THE TRUNCATE IS 10
FOR N= -1009+02 THE TRUNCATE IS 10
FOR N= -1155+02 THE TRUNCATE IS 11
FOR N= -1221+02 THE TRUNCATE IS 12
FOR N= -1287+02 THE TRUNCATE IS 12
FOR N= -1353+02 THE TRUNCATE IS 13
FOR N= -1419+02 THE TRUNCATE IS 14
FOR N= -1485+02 THE TRUNCATE IS 14
FOR N= -1551+02 THE TRUNCATE IS 15
FOR N= -1617+02 THE TRUNCATE IS 16
FOR N= -1683+02 THE TRUNCATE IS 16
FOR N= -1749+02 THE TRUNCATE IS 17
FOR N= -1815+02 THE TRUNCATE IS 18
FOR N= -1881+02 THE TRUNCATE IS 18
FOR N= -1947+02 THE TRUNCATE IS 19

```

FROM $\theta=.1$ BY .2 TO 3 COMPUTE $\alpha=\cos(\theta)$ AND $\gamma=\cos^{-1}(\alpha)$
AND PRINT $\theta, \gamma, \theta-\gamma$.

FINISH.

50010 FROM THETA=.1 BY .2 TO 3 COMPUTE ALPHA=COS(THETA) AND GAMMA=ARCCOS(ALPHA) AND PRINT THETA,GAMMA,THETA-GAMMA
50020 FINISH

```

          X14=3.14159265
          X15=2.7182818
- 97777  FORMAT (E14.8)
          X33=.1
          Q1=(Q=.2)/ABSF(Q)
          GOTO 98001
98002  X33=X33+.2
98001  IF((3.-X33)*Q1)98003,98004,98004
98004  X55=COSF(X33)
          X103=ARCCOSF(X55)
          P1=X33
          P2=X103
          P3=X33-X103
          WRITE 2,97777,P1,P2,P3
          GOTO 98002
98003  CONTINUE
          END

```

.10000000-00	.10000000-00	-.12587406-00
.30000000-00	.30000000-00	-.20934757-00
.50000000-00	.50000000-00	-.11932570-00
.70000000-00	.70000000-00	-.23374102-00
.90000000-00	.90000000-00	-.22700987-00
.11000000+01	.11000000+01	-.34342519-00
.13000000+01	.13000000+01	-.29103830-00
.14999999+01	.15000000+01	-.36088749-00
.16999999+01	.17000000+01	-.27357600-00
.19000000+01	.19000000+01	-.30867983-00
.20999999+01	.21000000+01	-.27939677-00
.22999999+01	.23000000+01	-.19790604-00
.24999999+01	.24999999+01	.14901161-07
.26999999+01	.26999997+01	.20570587-06
.28999999+01	.28999977+01	.22012973-05

*MISSION
of
Rome Air Development Center*

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C³I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.