DOCUMENT RESUME

ED 029 511 EM 007 246

By-Rosenbaum, J.; And Others
Computer-Based Instruction in Statistical Inference: Final Report. Technical Memorandum (TM Series).
System Development Corp., Santa Monica, Calif. Research and Development Div.
Spons Agency-National Science Foundation, Washington, D.C.
Report No-TM-2914-100-00
Pub Date 30 Oct 67
Note-201p.
EDRS Price MF-$1.00 HC-$10.15
Descriptors-•Computer Assisted Instruction, Computer Based Laboratories, Computer Oriented Programs, •Computer Programs, Concept Formation, Equipment, Interaction, •Number Concepts, Probability Theory, Problem Sets, Programed Tutoring, Programing Languages, •Statistics, Student Attitudes, Teacher Attitudes
Identifiers-PLANIT, •Programming LANguage for Interactive Teaching, System Development Corporation Time Sharing System, TSS

A two-year investigation into the development of computer-assisted instruction (CAI) for the improvement of undergraduate training in statistics was undertaken. The first year was largely devoted to designing PLANIT (Programming LANguage for Interactive Teaching) which reduces, or completely eliminates, the need an author of CAI lessons would otherwise have for professional programing assistance. Four lessons were constructed with PLANIT and presented in the second year to 21 undergraduates on two California campuses. Each student received four to five hours of instruction including PLANIT communication procedures and conventions, elementary probability, descriptive statistics, and computer programing. It was concluded that CAI is best employed in the numerical demonstration of statistical concepts and for statistical laboratory exercise instruction. Now, and for some years to come, only the more routine aspects of tutorial assistance can be provided with CAI; however, this assistance should be beneficial to both students and instructors. Documentation is provided. (GO)

TM-2914/100/00

TECHNICAL MEMORANDUM

(TM Series)

The work reported herein was sponsored by SDC and the National Science Foundation under Grant GY-371.

Final Report

Computer-Based Instruction in Statistical Inference

By

J. Rosenbaum, S. L. Feingold, C. H. Frye, and F. D. Bennik

30 October 1967

SYSTEM
DEVELOPMENT
CORPORATION
2500 COLORADO AVE.
SANTA MONICA
CALIFORNIA
90406

EM007246

ABSTRACT

This report describes a two-year investigation (September 1, 1965-September 1, 1967), conducted by members of the Education and Training Staff, System Development Corporation (SDC), into the development and use of computer-assisted instruction (CAI) for the improvement of undergraduate training in statistics. The target student in the study was the undergraduate social science major enrolled in a first course in statistical inference.

Most of the first year was devoted to the design and implementation of an important software innovation called PLANIT (Programming LANguage for Interactive Teaching) consisting of a CAI author language and an associated computer program that has been operating within SDC's time-sharing system (TSS) since the summer of 1966. PLANIT reduces, or completely eliminates, the CAI lesson author's need for professional programming assistance. The program has been operated in lesson editing, lesson presentation, and on-line calculational modes. The last is particularly useful for statistics instruction employing an expository (numerical illustration of statistical concepts) style and for statistical laboratory exercises.

The second year of the study was devoted to the construction and testing of CAI lesson material. Four experimental lessons were constructed and presented with PLANIT to a total of 21 undergraduates on two California campuses (University of Southern California and San Fernando Valley State College). These students, using campus teletypewriters linked to TSS by DATAPHONE, each received between four and five hours of instruction including PLANIT communication procedures and conventions (INTRO), elementary probability (PROB1), descriptive statistics (DESCRB), and computer programming (PROGM). The field tests (and preparing CAI lessons used there) provided the project staff with ample opportunities to observe both the problems and benefits that accompany CAI. These observations (and impressions) are discussed from the standpoint of students, instructors, and lesson authors.

It is concluded that CAI is most appropriately employed in the numerical demonstration of statistical concepts and for statistical laboratory exercise instruction. Currently (and probably for some years to come) only the more routine aspects of tutorial assistance can be provided with CAI; however, the inclusion of such assistance in expository and exercise lesson material should produce benefits for both students and instructors. A recommendation is made that PLANIT be reprogrammed for less costly operation with computing equipment available at many colleges and universities to provide a CAI utility for these schools.

ACKNOWLEDGMENTS

The following faculty members of the University of California, Los Angeles, gave generously of their time and knowledge, in the early phases of this study, to acquaint the project staff with those parts of the statistics curriculum that are troublesome to beginning students.

Professor Virginia A. Clark, School of Public Health

Professor Theodore R. Husek, School of Education

Professor Morton P. Friedman, Department of Psychology

Professor Bradley Bucher, Department of Psychology

We are also grateful to:

Professor Raymond Berger, Department of Psychology, University of Southern California, Los Angeles, California

Professor Robert E. Dear, Department of Psychology, San Fernando Valley State College, Northridge, California

and the students in their statistics classes for their participation in field tests of computer-assisted instruction. These students and their teachers took time from busy schedules to participate in the study.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

This final report for the study, "Computer-Based Instruction in Statistical Inference," supported by the National Science Foundation (Grant GY-371) and System Development Corporation, describes a two-year investigation (September 1, 1965, to September 1, 1967) into the uses of on-line computing systems for instruction in elementary statistics. The study team, members of the Education and Training Staff of SDC's Research and Development Division, employed the Corporation's time-sharing system in their work. In order to create a variety of instructional sequences (lessons) efficiently, a computer program called PLANIT was developed and used to build four lessons which were then tested in campus trials with undergraduates. Three major parts (II, III, IV) of this report follow the introduction and provide a description of PLANIT and the lessons constructed with that program, study team impressions and observations derived from the campus tests, and finally some conclusions and recommendations for future development and use of this mode of statistics instruction.

A. BACKGROUND

Teachers of theoretical and applied statistics have always recognized the value of including exercises requiring numerical computation on sample data in their students' assignments. Such work provides the instructor, particularly in the elementary courses, an opportunity to assess the student's ability to compute--his understanding of the correspondence between sample data and the symbols in a calculation formula, and most important, his ability to select and implement appropriate techniques of statistical analysis.

Until quite recently, the digital computer was not employed in elementary courses and was used sparingly in advanced courses, primarily for exercises that required significant amounts of computing. Most of this work was and is still done by students in the so-called statistics laboratory, equipped with desk calculators. With the emergence of "on-line" use of computers, more convenient and immediate communication between student and computer became possible. Students in elementary statistics courses at Dartmouth College are taught BASIC [11] a programming language designed for use by students in mathematics and science courses. Twenty, or even thirty, students may simultaneously write short programs while they are seated at teletypewriters linked to a central and time-shared computer. Then these programs may be "called" for use in numerical exercises, e.g., in statistics, tics. Schatzoff [15] developed a number of programs with which advanced statistics students may communicate directly for complex analysis of variance computations. Grubb [5] did some pioneer work of a quite different sort. He too made use of "on-line" communication with a computer in statistics instruction, but not merely for computation. He attempted to actually instruct beginning students in descriptive statistics via the computer. The computational aspect in this instruction was minor; more important was Grubb's use of the computer to "branch" the student through an instructional sequence, as a function of the replies that the student typed in response to numerical problems in descriptive statistics. The computer evaluated the student's replies, sent messages back (via typewriter) telling him whether or not he was correct, and then branched him to new or remedial work accordingly.

One of the authors of this report (C. H. Frye) conceived the idea of an on-line computer-based statistics laboratory in the summer of 1964. By the spring of 1965 his idea was implemented in the form of a computer program called STAT, operating within SDC's time-shared computer system. STAT was an attempt to incorporate in a single program some of the computational capability afforded by Dartmouth's BASIC with the computer-assisted instruction (CAI) provided in Grubb's work. STAT provided richer CAI than the latter, however, with more computer-mediated assistance for the student and more complex problems for him to solve (inferential rather than descriptive statistics). In addition, it automated for the laboratory portion of a statistics course several functions normally performed by the instructor, such as providing data for problems.

The two-year study reported here began in September 1965 with an analysis of the results gathered from student trials of STAT conducted during the spring of 1965. The study objective was to investigate the role of CAI in supplementing the instruction provided in the usual first course in statistical inference for social science undergraduates. Because STAT provided only the laboratory exercise mode of instruction, it was not a sufficiently useful tool for investigating the computer's role in expository and tutorial modes of instruction. STAT was also unsatisfactory from the viewpoint of the CAI author who was not also an expert programmer. Any revision to STAT--and many were required following its trial use by students--called for the intervention of a highly skilled programmer familiar with JOVIAL, the general-purpose language used to program STAT. Assuming the programmer understood the nature of a requested revision (which was not always the case) a day or longer would elapse before STAT was recompiled in revised form and available for further inspection by the author or for trial use by students. This delay is, of course, frustrating in the lesson-building stage and often intolerable if students are tightly scheduled to use CAI materials that require revision. One of the most important results of this study is a CAI system called PLANIT (Programming LANguage for Interactive Teaching) which consists of an author language and a computer program for the preparation, editing, and presentation of lesson material. Subsets of the language are sufficiently simple that they can be mastered quickly by subject-matter experts and teachers. The system operates interpretively so that the results of the author's work are available, in the form to be seen by the student, within seconds after any portion of a lesson has been prescribed or revised.

B. CONTENTS OF THIS REPORT

Part II of this report describes (1) the SDC Time-Sharing System (TSS) in which PLANIT has been operating since June 1966; (2) the evolution from STAT to PLANIT; (3) the PLANIT system itself; and (4) four lessons, employing all three instructional styles (expository, tutorial, laboratory exercise), that have been built using this system.

Part III discusses the staff's experience in preparing for and observing student trial use of these lessons on two California campuses--the University of Southern California, Los Angeles, and San Fernando Valley State College, Northridge.

Part IV contains conclusions and recommendations regarding the future development and more widespread distribution of CAI. In summary, these state that PLANIT is a valuable tool for the construction and presentation of CAI lessons employing the statistical laboratory exercise and expository (i.e., numerical demonstration of statistical concepts) modes of instruction. In order to make this tool available to college faculties and their students, it is recommended that PLANIT be reprogrammed to operate with medium-small computers currently available, or soon to be, on many campuses. It is further recommended that in the interim period, while PLANIT is being reprogrammed, a concurrent activity should be supported for the preparation of lesson material in statistics.

Five appendices are included in this report:

Appendix 1 is for those readers interested in the TSS hardware components used in this study.

Appendix 2 lists the statistical subroutines (options) that are available in the STAT program and provides an illustration of a student's work with STAT.

Appendix 3 provides a rather detailed exposition of the lesson building and in less detail, the computational capabilities of PLANIT. Two "Notes" at the back of this appendix describe the techniques used in PLANIT for "PHONETIC," and "FORMULAS" matching when the student's response to a lesson query is compared with a stored correct answer.

Appendices 4 and 5 document in detail the two most complex lessons, DESCRB and PROGM, constructed with PLANIT. An instructor, contemplating the use of CAI, cannot browse through lessons such as these as he might a textbook. This documentation represents an initial attempt to provide examples of documentation that could spare the instructor many hours at the teletypewriter. The reader who is not interested in this problem, or the attempted solution will find adequate lesson descriptions in the body of the report, Part II.

II. PROJECT SOFTWARE PRODUCTS: PLANIT AND LESSONS

Two kinds of software were produced and tested during this two-year study. The first kind consists of a computer-assisted instruction (CAI) author language and an associated computer program; the combination is called PLANIT (Programming LANguage for Interactive Teaching). PLANIT was then used to generate the second type of software, which consists of four CAI lessons (called INTRO, DESCRB, PROB1 and PROGM), and also to present these lessons to students via teletypewriter.

PLANIT was written in JOVIAL* and compiled for execution in SDC's Time-Sharing System (TSS) to operate under the control of the TSS executive program during its various use modes.

This section presents a brief description of TSS and the computer program called STAT from which PLANIT evolved. It concludes with a description of PLANIT and the four CAI lessons produced thus far.

A. THE SDC TIME-SHARING SYSTEM

The SDC Time-Sharing System became operational at the Santa Monica facility in June, 1963, and has been in continuous use since that date. The term "time-sharing," as used here, means simultaneous access to a computer by many users and programs. Each program, when not active, resides in disc memory (or on magnetic tape) and is activated by the user when he types a LOAD command on his teletypewriter (TTY) or other keyboard device. The TTY may be quite remote from the system's computer in Santa Monica as long as it is linked by a two-way communication channel (e.g., DATAPHONE, TWX, leased line, etc.). As many as 20 or 30 users may have shared access to the computer at any time. When someone logs into the system (i.e., identifies himself as an authorized user) and then types a LOAD command, the requested program--in this case PLANIT--is transferred from disc (or tape) into the computer's magnetic drum memory and execution of instructions in that program can begin. When the program is active in the system, it shuttles back and forth from drum to the computer's high-speed core memory, remaining in the latter for a few hundred milliseconds (or less) at a time. During these short intervals, the computer performs any outstanding tasks that the user may have requested via his TTY. The requests and programs of all users are treated in this fashion, although at times some may be given priority treatment and others may be placed in queues to await service. An individual user such as a student at a remote location is hardly aware that others are "sharing" the computer, since an average of only four or five seconds will elapse between the time he sends a message to the computer via the TTY and the time the computer replies via the same TTY. (In these few seconds, the computer completes a cycle of service for the other users.)

This "conversational" or on-line use of a computer is particularly suited to educational applications. The student may receive almost instantaneous feedback from the computer about the correctness of the answers he submits to questions posed by the computer. In addition, the computer can keep a record of the student's replies and use this "history" to determine an appropriate path (i.e., "branch") for the student to take through an instructional sequence; the sharing aspect of computer use is of interest too, since it permits on-line interaction at relatively reasonable prices; the fixed costs of computer rental or purchase can be divided among the several computer sharing users.

* A high-level programming language and compiler developed at System Development Corporation. It is described in [14].

More details about SDC's approach to time-sharing may be found in [16] and [17]; a schematic picture of the SDC system and a listing of its hardware components is contained in Appendix 1. A more general treatment of on-line and time-shared systems is available in [10] , [13] and [18] .

B. THE EVOLUTION FROM STAT TO PLANIT

1. Description of STAT*

STAT is a computer program that has been operating within SDC's TSS since the fall of 1965. It provides the laboratory part of a first course in applied statistics--the part that teaches the student to apply the theory and technique discussed in the classroom and textbook. In the conventional laboratory situation, the student is asked to work exercises consisting of some sample data and a description of their origin; and he is told to use given statistical techniques, or to select some, for population parameter estimation or statistical tests of hypotheses. He then performs numerical computations on the sample data with the aid of a desk calculator. In addition to preparing problem statements and generating sample data, the lab instructor or his assistant must correct papers, keep records of student performance, and assist the student who gets "stuck." STAT represents an attempt to harness computer power, on behalf of both students and instructors, for this instructional situation.

Once the student types 'LOAD STAT', and the TSS executive program responds that STAT has been loaded into the computer, the student begins a dialogue with a computer-based laboratory course in statistics. STAT is programmed to present a sequence of 25 exercises in a specific order (which can be altered). When STAT causes the text for an exercise to be printed, it prints the message OPTION =?, on the next teletype line, to indicate to the student that he is to choose a number designating an option (actually a subroutine or subprogram). If he types '1', he activates the option for drawing a random sample appropriate to the current exercise. After the sample is generated and typed as a column of numbers (or pairs of numbers for a bivariate or two-sample exercise), the message, OPTION=?, is printed. He may now choose from a variety of some 50 remaining options, which fall into four major categories:

- HELP
- CALCULATION
- LIBRARY
- RESPONSE/EVALUATION

* A detailed description of STAT, together with instructions for its use, is available in Student's Guide to STAT, TM-2910/000/00, System Development Corporation, March 31, 1966.

HELP options provide the student with references to readings in a textbook* for steps leading to solution of the current exercise or for explanation of a LIBRARY option; the desperate student may even get the correct answer to that part of the exercise for which he has submitted an incorrect answer to STAT.

CALCULATION options, as the category name implies, provide the student's computing tools. The most important of these, OPTION 12, computes numerical values for algebraic expressions that the student types (e.g.,"SS2/15" will yield the numerical value of the sum of squares of the data in column 2 of his sample, divided by 15).

LIBRARY options operate on sample data only and are used for automatic calculation of sample statistics, e.g., means, variances, correlations, regression coefficients, and various statistical test criteria or confidence interval limits. For example, if the student types '21', in response to 'OPTION=?', the value for the mean of the generated sample data will be automatically computed and printed. When the student uses the LIBRARY to work an exercise, there is no chance for him to err, except if he chooses an inappropriate LIBRARY option.

RESPONSE/EVALUATION consists of a single option, OPTION 8. Here the student submits the answer resulting from his work for evaluation by STAT; these answers (usually numbers, but occasionallly yes or no) are judged to be correct or incorrect. If the latter is true, the computer message indicating this, includes direction to appropriate readings in the textbook and advice to try again.

STAT normally presents each exercise to the student once or twice. During the first presentation, the LIBRARY is available to the student. If he is able to supply correct answers during this first presentation without using the LIBRARY, he is automatically advanced to the next problem, in programmed order. If he does use the LIBRARY during the first presentation, STAT will present the problem again (with different sample data). However, this time, the LIBRARY is not accessible and he can only use CALCULATION and HELP options to work the exercise. Thus the student is "forced" to demonstrate his ability to work each exercise, throughout the entire sequence without assistance from the LIBRARY.

Progress to a second presentation or to a new exercise is effected automatically by STAT. The student progresses in the sequence, in programmed order, only when he submits correct answers for the current exercises, unless he elects to skip to the next exercise after unsuccessful attempts to answer correctly.

The student may terminate a session at the teletypewriter by choosing OPTION 20. This provides him with a continuation code number to insert via teletype when he next uses STAT; he is then automatically brought to an appropriate continuation point in the exercise sequence. OPTION 20 also provides a performance record

* Hays, W.L. Statistics for Psychologists. Holt, Rinehart and Winston, Inc., 1963.

for the student and/or the instructor, which indicates, for each completed exercise, the amount of time the student worked on the exercise, the number of times incorrect answers were submitted for evaluation, the number of steps to a solution that the student requested from STAT, and some other items.

Appendix 2 contains some excerpts from the Student's Guide to STAT: a complete list of STAT options, and the mathematical conventions employed in OPTION 12 for computation. A portion of a student's teletypewriter record of work with STAT is also included.

2. Disadvantages of STAT*

It became clear at the outset that the STAT program would not be sufficient for project requirements. First, there was a need to explore the computer's role in expository and in tutorial mode of instructions, whereas STAT provided only laboratory exercise instruction. Second, even for the latter purpose, serious disadvantages to STAT had been observed during student trial use of the program. The project staff analyzed these STAT deficiencies from the standpoint of students, instructors, and authors of CAI sequences.

a. Disadvantages for the Student

The rate at which a student can work with STAT at the TTY is unnecessarily slow for two main reasons. As stated earlier, the LIBRARY options are available to the student only during the first presentation of each exercise. If he uses even one such option during the initial presentation, STAT requires that he work this exercise again (with different sample data) without LIBRARY assistance. Often the students who used STAT would demonstrate their mastery of particular statistical calculations in one exercise, i.e., they obtained correct answers without using a single LIBRARY option, and yet they were effectively denied use of LIBRARY options for the very same calculations in subsequent exercises (if they wished to avoid second presentations). These students could have worked through the exercise sequence more quickly and with less frustration if STAT provided more flexible control over LIBRARY access.

These students would also have been able to work faster if they could have constructed mathematical functions for their own individiual use. (OPTION 12 in STAT does provide numerical evaluation of algebraic expressions and a few functions of sample data arrays, e.g., S1, the function that sums all the data in column 1 of a sample may be used in these expressions.) The

* This discussion is based on an analysis of student trials with the STAT program, which took place at SDC, Santa Monica, California during the spring of 1965. Nine paid volunteers (UCLA graduate students in psychology) used STAT during five 3-hour teletypewriter sessions. A more detailed and complete discussion of the results of these trials is available in C. Frye [4] .

student should also be able to label the functions he defines and call them up for execution whenever (and for as long as) he needs them. In effect, he would then be able to create his own library options.

There was also a need for a richer branching capability than STAT provided. If the student was able to supply correct answers, he was branched to another exercise presentation. If not, he was returned to the current exercise for further work. No matter how many times he input an incorrect answer, the resulting STAT action was identical (unless he gave up). It was clear that effective CAI would require more varied branching as a function of the student's "history" of responses as well as his current response.

Finally, the permissable forms of student response were too restricted; the student could reply only with a number or a single word (often a single letter) to questions posed by STAT. Even in the case of laboratory exercises, much better instruction can result if student replies take the form of entire phrases or mathematical formulas as well as numbers. For tutorial and expository dialogues, such an expansion of possible student answer forms is a necessity.

b. Disadvantages for the Instructor

Some on-line alteration of characteristics of STAT's behavior toward the student can be effected. (There are options reserved to the instructor for this purpose.) The instructor may send information to STAT, via teletype, to alter any combination of the following:

- sample size for an exercise
- level of significance or confidence level values that appear in the text of an exercise statement
- probability distribution parameters (i.e., population parameters) used in generating samples
- order of presentation of exercises in the sequence
- maximum number of presentations for an exercise
- the number of presentations for an exercise during which LIBRARY options are made available to the student

The instructor cannot alter the type of probability distribution used to generate samples for an exercise, the text for an exercise, the criteria for acceptable answers, or the recommended steps to a solution, unless he is proficient in JOVIAL programming. The same applies to augmentation or deletion of LIBRARY options; most critically, he cannot make subsets of the LIBRARY available or unavailable for particular exercises. Indeed, even if the course instructor were proficient in computer programming, using the JOVIAL language, he could not effect these alterations without reprogramming major portions of STAT.

It was clear, after examination of the STAT trial experience, that an expanded capability was required for on-line alteration of STAT's behavior. It was equally clear that such alteration should not (and need not) require the intervention of a professional computer programmer. For example, it was discovered during one trial that STAT was not evaluating the student's answers for a particular exercise correctly. The intervention of a programmer familiar with the JOVIAL language and with the intricacies of STAT's answer evaluation logic was required in order to make this trivial program change. Nearly a day elapsed before STAT was altered and recompiled and this amended version of the program was available for student use. Similar problems arose when there was a need to make minor changes in the text of an exercise statement, or in other messages from STAT, to remove ambiguities. The instructional sequences produced with PLANIT (which is discussed later) will permit an instructor to intervene directly, from his teletype machine, to effect changes in a greatly expanded set of items that affect program behavior and content.

c. Disadvantages for the Lesson Designer

Often, of course, the lesson designer (author of an instructional sequence) is also an instructor. However, an author has an even greater need for the alteration capabilities discussed above. Presumably, an instructor will be working with well-tested programs that will require only modest intervention to remove a few errors, or to tailor behavior to his taste or to the requirements of his students. The author, on the other hand, has a continual need to make changes while he is developing an instructional sequence. In addition, he requires a program capability for initial preparation of instructional materials. While STAT has some on-line alteration features, it has none that permit an author, unskilled in computer programming, to insert new or additional instructional material together with specifications for its presentation to the student.

In order to overcome these disadvantages and to meet the project's requirements for efficient generation (and subsequent alteration) of various styles of CAI, expository and tutorial sequences as well as laboratory exercises, PLANIT was developed.

C. PLANIT* (PROGRAMMING LANGUAGE FOR INTERACTIVE TEACHING)

PLANIT ([2], [3]) is a system consisting of a language, a computer program, and the rules governing the use of both, for building, editing, and controlling the

*This discussion is adapted from a paper by Samuel L. Feingold, "PLANIT, A Flexible Language Designed for Computer-Human Interaction," to be presented at the 1967 Fall Join Computer Conference, Anaheim, California, November 1967.

presentation of CAI sequences. The major portion of the system design was completed by mid-January 1966, following an analysis of the STAT experience, described above, and a survey of existing CAI software.* A first version of PLANIT was released for project use in May, 1966, and the system has been operational within SDC's TSS with periodic improvements since that time. PLANIT is the most important result of this project and represents a major step forward in CAI software. The following sections provide a brief description of the capabilities of the system and an exercise in lesson-building using PLANIT. A more complete description is provided in Appendix 3, "Brief User's Guide to PLANIT."

1. PLANIT Features

The system is described here from the viewpoint of an author of CAI material, e.g., a teacher, who is called a lesson designer (LD).

PLANIT operates in four modes: lesson building, editing, execution, and calculation. Access to the system in any of these modes is always made via a keyboard device such as a TTY. The first two modes permit a teacher to construct and edit lesson frames in various formats and store them in designated sequences for later presentation to the student in the execution mode. The calculation mode is particularly oriented to mathematical subject matter and can be used as a calculation aid for the teacher (when building the lesson) or the student (when performing the lesson). While the student has access only to modes EX (execution) and CALC (calculation), the LD may use all four modes.

- Calculation Capability. PLANIT has an on-line calculation capability that allows either the LD or the student to perform calculations involving trigonometric functions, elementary algebraic functions, and matrix declarations. In addition, the calculation functions of PLANIT can be tied in with the lesson. The LD can request the student to compute some results and can specify that the student's answer be compared with the results of evaluating a previously defined function. The LD can use the CALC mode to define the function when preparing the lesson; during lesson execution, the CALC mode can be used again to "test" the student's answer.

- Criterion Branching. The PLANIT language allows the LD to specify conditions for branching based on the student's performance over any portion of the lesson.

* *IBM's "COURSEWRITER" [9] , G. E./Dartmouth's "BASIC" [11] , Bolt Beranek and Newman's (Cambridge, Mass.) "MENTOR" and University of Illinois' "PLATO" [1] were the most important systems examined. Each of these systems lacked some feature deemed essential for efficient generation, editing and execution of statistics-oriented CAI. See Frye [4] , Chapter VI, for a discussion of desirable features of CAI software.

Conditions for branching may include:

Response latency on any one answer or group of answers.

Number of errors made on any group of questions.

Help received from the CALC mode (functions used or not used).

The actual path taken through the lesson up to this point.

Any combination of the above four points.

- Service Routines. PLANIT also provides the following service functions for evaluating student answers that depart from the anticipated responses specified by the LD:

  PHONETIC Comparison. This routine gives the student credit for his answer even though it is spelled incorrectly as long as it is phonetically equivalent to one of the LD's answers. For example, PLANET and Fonetic are acceptable wrong spellings for PLANIT and PHONETIC.

  KEYWORD Match. This routine instructs PLANIT to search for a set of words in the student's reponse as the KEYWORDS of his answer. Furthermore, if the lesson designer wishes, he may specify that the answer be evaluated as correct only if these keywords appear in a prescribed order.

  FORMULA Equivalent.* This routine will allow the student credit for his answer as long as it is one of a subset of expressions algebraically equivalent to any one of the LD's answers. For example, if one of the LD's anticipated answers is 5/9*(F-32), then (5*F-160)/9 or 5*(F/9)-160/9 or (-32*(5)+5*F)/9 would be equivalent and therefore acceptable.

  The LD can have any combination of these three routines turned off and on during lesson execution (even between actual occurrences of anticipated answers during student interaction with the lesson).

2. Lesson Building with PLANIT

This section explains how a lesson is written in PLANIT. A lesson is composed of a set of frames; frames are composed of groups; groups are composed of lines of information, such as text to be presented, anticipated answers, actions, etc.

*We believe this feature to be unique to PLANIT among CAI systems.

There are five **frame types**: **Problem, Question, Multiple Choice, Decision, and Copy.** The Q (Question) frame is explained and illustrated below. The P, D, M, and C frames are discussed only briefly.* In the frame below, as in practically all PLANIT dialogue, data entered by the user follow an asterisk typed out by PLANIT. All data are entered via teletype.

a. The Q(Question)Frame

```
FRAME 1.ØØ LABEL=*HIST

2. TEXT.
*?

2. SPECIFY QUESTION.
*WHO INVENTED THE ELECTRIC LIGHT?
*

3. ANSWERS.
*A+THOMAS EDISON
*B ALEXANDER BELL
*

4. ACTIONS.
*A F: THATS VERY GOOD B:3
*B R:HE INVENTED THE TELEPHONE, TRY AGAIN...
*-R:
*-C:
```

Explanation of Frame 1

First line. All frames are automatically numbered by the program; the LD provides the labels. Here the LD chooses to label the frame HIST. (If he chooses not to label the frame, he strikes the space bar and the carriage return and passes on to Group 2. Group 1 consists merely of the frame label.) In Group 2: TEXT, the LD is not sure what TEXT means and types in the question mark (?). PLANIT immediately repeats the group number, 2, and elaborates. The LD then types in his question, WHO INVENTED THE ELECTRIC LIGHT? PLANIT returns with an asterisk, waiting for more lines of input. PLANIT has no way of knowing when the LD is through with the question group and so returns with an asterisk for each next line. The LD ends the group by striking the space bar once and then the carriage return key. There is no theoretical limit to the number of lines that can be entered in each group. There is, however, a practical limit of 63 lines for the entire frame.

*See Appendix 3, especially pages 104 to 109 for further details about these **frame types.**

In Group 3: ANSWERS, PIANIT is asking the user to SPECIFY ANSWER. The LD now enters all the anticipated answers, tagging the first one A and the second B, etc. The plus sign (+) next to the A indicates to PIANIT that this is the correct answer. The LD then indicates the end of the group by striking a space bar and carriage return.

In Group 4: ACTIONS, the LD user is requested to SPECIFY ACTIONS to be TAKEN, depending upon which answer the student gives.

There are four types of commands in the action group:

F: What follows is the feedback message that is to be presented to the student. If no message follows F:, PIANIT will choose one randomly from its stored list of feedback messages, according to whether the student's answer was correct or incorrect. The LD can thus enter F: by itself, knowing that the student will not get a monotonous YES or NO when he enters an answer. (Responses are usually one-word messages such as FINE, YES, CORRECT, NO, WRONG.)

R: This command instructs PIANIT to wait for another answer without printing the question again. It can be entered along with an appropriate message such as in the above example. In this case the student receives the feedback: HE INVENTED THE TELEPHONE, TRY AGAIN; and PIANIT then waits for another answer. R: by itself instructs PIANIT to print out a fixed message (WRONG, TRY AGAIN) and wait.

C: This command used alone instructs PIANIT to print out the fixed message: THE CORRECT ANSWER IS, followed by the correct answer (indicated by the plus sign in Group 3). C: can be followed only by another command--F:, R:, B: or a CALC statement. For example, C: COUNT=COUNT*1 or C:X=FACT(3). This puts PIANIT in the CALC mode. COUNT and X are two examples of items in CALC. Here COUNT is to be incremented by one and X is set equal to FACT(3), i.e., the factorial of 3.

B: This instructs PIANIT to branch (B:3 means BRANCH TO FRAME 3). All frames are numbered; they can also be labeled and a branch can be made to any numbered or labeled frame. In addition, B:LSNAM (where LSNAM is the name of another lesson) means that the lesson LSNAM will now be brought into PIANIT and executed as a part of the current lesson; the student will never know the difference. Upon completion of the lesson LSNAM, PIANIT will continue with the next frame in the original lesson. B: by itself causes PIANIT to return to the calling lesson. For example, if during lesson AA the command B:BC is encountered (where BC is a lesson name), lesson BC will be called and run until the command B:, in lesson BC, automatically returns PIANIT to lesson AA. Similarly, B:PRGM means branch to the program whose name is PRGM. When PRGM relinquishes control, execution continues with the next frame in the calling lesson.

The first user input in Group 4 is read as follows: If the student gave answer A (THOMAS EDISON), print out the message: THATS VERY GOOD and branch to Frame 3.

The second input is interpreted as follows: If the student gave answer B, print out: HE INVENTED THE TELEPHONE, TRY AGAIN...and wait for another answer.

The third input, prefaced by a minus sign, indicates an action to be performed if the student's answer did not correspond with either A or B--namely, for any unanticipated response, wait for another answer. (In this case, since nothing appears after the R:, PLANIT then prints out the fixed message: WRONG, TRY AGAIN, and waits for another answer.)

The LD may then select the same or another frame type and continue to build his lesson.

The Q frame is the cornerstone of any interactive tutorial dialogue with the student. Even a casual user of PLANIT quickly masters the use of this frame. The small (but powerful) set of Group 4 action commands, F, R, C, B, permits him to prescribe flexible branching and feedback messages appropriate to the student's response to a question posed in Group 2.

b. <u>The P(Problem) Frame</u>

This frame type is designed especially to provide the "environment" for statistical laboratory exercises. The LD may use as many as nine information groups to specify:

- Probability distribution parameters for generating data in the form of pseudo-random samples, e.g., means, variances, and correlation coefficient for a bivariate Gaussian (normal) distribution. The random data would actually be generated for student use during the execution, for example, of a statistics lesson (the LD can also specify headings and format for the actual printing of the data). The amount of data to be generated may be controlled by the student or the LD according to the latter's prescription.

- Steps to the solution of a problem in which the random data will be used (the student receives one step at a time in response to his request "STEPS").

- Controls over the mathematical functions that shall (or shall not) be made available to the student when he attempts to solve a particular problem.

When this frame is used in conjunction with Q frames (preceding and following), the LD can construct instructional sequences such as that provided by

STAT (without any of the disadvantages of that program described earlier). A preceding Q frame, usually consisting only of Groups 1 and 2, is used to present a textual description of conditions giving rise to the pseudo-random data. The Q frame that follows the P frame will usually contain information in all four groups, including a question that requires the student to perform some computation on the data before he is able to reply; anticipated answers stored as functions (of the unknown and as yet ungenerated sample data); and appropriate action commands and feedback messages. (See Appendix 4, page 140, where these frames are combined in the prescription of a computational exercise for a lesson in descriptive statistics called DESCRB.)

c. <u>The D(Decision) Frame</u>

The action commands (F, R, C,B) in the Q frame permit the LD to specify lesson execution behavior as a function of only those events that occur in connection with the execution of a single frame (including repetitions thereof). The Decision frame affords the LD an opportunity to consider more complex branching decisions (and other actions as well), as a function of the student's performance history, i.e., as a function of events that have taken place across a set of frames. He may use two basic forms in this connection.

The first form, called the pattern form, allows one to inquire if the student took a particular path through the material. For example, let us imagine the student went through Frame 1, answered A or C, and followed it by Frame 5 (in which he responded incorrectly), and then followed that by Frames 10 through 15, where the student was correct. An inquiry concerning whether or not the student went through that pattern exactly with no deviations can actually be written in the language pretty much as it has been stated. For example:

IF 1,AC 5,- 10-15,+

This, then, is the form of the pattern question. If the query is answered affirmatively, one can then use any combination of the three action commands:

F:, C:, and B:.

The second form permits queries about summarized student performance over a set of frames. For example, one may want to know if the student got less than or equal to five right out of Frames 10 through 22 and Frames 30 and 33. This can also be written, almost as stated, in the following manner: IF LQ5 RIGHT 10-22,30,33. Similarly, if these conditions are satisfied, any of the three commands can then be executed. In place of RIGHT, one can substitute WRONG, SEEN, MINUTES, USED. With USED, one can have any function like FACT (factorial), SIN, COS, etc. For example:

IF GQ 5 MINUTES 10-15 B:10

IF SIN USED 6 B:7

means: If the student spent at least five minutes on Frames 10 through 15, then branch to Frame 10. The second IF statement reads: If he used the function SIN in Frame 6, then branch to Frame 7. Also, in place of GQ one can substitute LS for less than, GR for greater than or other relationals such as LQ, NQ, EQ. Finally, one can use IF statements to query the contents of items set in the calculation mode. IF IQ LS 50 B:WORK means: If the item whose name is "IQ" contains a value less than 50, branch to the frame whose label is "WORK."

All these forms can be connected as one large statement via the connectives AND and OR, e.g.:

IF 1,AC 5, - 10-15,+

OR GQ 5 MINUTES 10-15 AND USED SIN 6

AND IQ LS 50 B:WORK

d. The M(Multiple Choice) and C(Copy) Frames

The M frame is almost identical in all respects to the Q frame except that it causes a presentation to the student of a "tagged" list of answers specified by the LD (the correct answer designators, of course, are omitted from the presented list). The student, instead of constructing a response in reply to an M frame message, selects and types a literal answer tag. The M frame is very useful for cases in which elaborate forms of student response are required (often too time-consuming to type) or where the set of "Correct" constructed responses may include too many variations, with respect to content and format, around one basic form.

The C frame is mainly a lesson building and editing convenience. Where the LD needs another version (with some minor differences) of a previously constructed frame, he can have it copied by calling for a C frame in the lesson building mode, designating which frame is to be "copied," and effecting minor alterations thereafter.

e. The CALC Feature and Expository Lessons

We have mentioned the on-line calculation capability of PLANIT; Appendix 3, Pages 92 to 97, contains a partial list of the variety of mathematical statements or expressions that may be employed by students (or other users) for on-line numerical work. These statements and expressions can also be "stored" by the LD in D frames and Q frames to provide valuable numerical demonstrations of statistical concepts when those frames are executed for the student.

Consider the following portion of a TTY record, Figure 1, obtained during a demonstration of the fact that care is required in interpreting observed sample values of the (Pearson product moment) correlation coefficient. The replies of a mythical "student" are underlined in this record.

```
                         CORRELATION AND INDEPENDENCE

  IT IS OFTEN (INCORRECTLY) CONCLUDED THAT A CORRELATION VALUE CLOSE TO ZERO
IMPLIES A LACK OF RELATIONSHIP BETWEEN TWO VARIABLES.  THE FOLLOWING SAMPLING
EXPERIMENT INDICATES THIS IS NOT ALWAYS THE CASE.

  CONSIDER THE FOLLOWING SAMPLE OF SIZE 'N' FOR THE VARIABLES, X AND Y; PLEASE
CHOOSE A VALUE FOR 'N' (BETWEEN 1Ø and 3Ø).

   N = 1Ø

   YOUR SAMPLE AND THE CORRELATION COEFFICIENT FOR THIS SAMPLE, 'RHO,' FOLLOW:

                  X              Y
                 ---            ---
         1-     5.7357         0.5204
         2-     0.4461         0.4314
         3-     2.2314         0.7895
         4-     5.3921         0.7777
         5-     4.6548         0.9984
         6-     1.4223         0.9890
         7-     5.7579         0.5013
         8-     4.2327         0.8871
         9-     4.3651         0.9403
        10-     1.4889         0.9967

   RHO = -Ø.142

    HOWEVER Y IS PERFECTLY PREDICTABLE FROM X: THE COMPUTER USED THE ABSOLUTE
VALUE OF SIN(X) IN TABULATING THE VALUES OF Y SHOWN ABOVE FOR EACH RANDOM
OBSERVATION ON X.

THE CLOSER 'N' IS TO 3Ø, THE MORE LIKELY ARE SMALL VALUES FOR 'RHO'.  WANT TO
TRY THIS AGAIN?  YOU MAY VARY 'N' IF YOU LIKE.

*YES
```

FIGURE 1.* Correlation and Independence

*Upon repetition of this "experiment" for N = 20 and N = 30, RHO values of 0.167 and -0.099, respectively, were obtained. The "population" correlation is actually zero for these two variables; this can be proven mathematically, i.e., if X is uniformly distributed on the interval $[0,2\pi]$ and $Y = |\sin(X)|$, then X and Y are statistically independent (uncorrelated) variables (even though a knowledge of X is sufficient to predict Y with certainty).

Such demonstrations, in which the student is free to vary a critical parameter, e.g., sample size, give the student (and his instructor) accelerated experience in stochastic phenomena, without the computational drudgery. In addition, the preparation of such CAI sequences is relatively easy with PLANIT; the lesson used for Figure 1 was built in one hour, including debugging time. It consists of six frames: four Q frames and one each of the P and D type. The hour also included time to label the lesson as CORR and store it in the TSS disc memory, from which it could be retrieved for immediate execution by any authorized user.

The next section describes four lessons that have been subjected to trial use by students at two college campuses. The trial results are discussed in Part III.

D. LESSONS

The project staff used PLANIT to construct four lessons, INTRO, DESCRB,* PROB1 and PROGM*. Each is described below with brief statements about lesson prerequisites, objectives, content and style; excerpts from TTY records of various students work are included, at the end of each description, to illustrate content and style.

1. INTRO

INTRO is a CAI lesson of approximately one hour's duration.** It presents the essential teletypewriter communication procedures and certain symbol conventions that the student must understand and use with PLANIT-administered instruction.

Prerequisites

It is recommended (but not essential) that students be acquainted with the nature of time-shared computer systems and their use in CAI applications before beginning INTRO. They should have seen a demonstration on operating the teletypewriter. For some portions of the lesson, a mathematics background equivalent to that provided in one year of high school algebra is assumed.

Objectives

The primary objective is to prepare the student for effective communication, via teletypewriter, with PLANIT operating in the lesson execution mode. A portion of the lesson is devoted to preparation applicable in all types of subsequent lessons. However, most of INTRO is devoted to a subset of the more specialized PLANIT conventions and procedures required for mathematical or statistical subject matter.

*More detailed descriptions of DESCRB and PROGM may be found in Appendices 4 and 5.

**Based on a sample of 14 college undergraduates, the median time for completion of INTRO is 73 minutes with a range of 56 to 87 minutes.

Content.

The sequence of topics in INTRO does not appear entirely in the order used to describe them here. The location and role of certain teletypewriter keys is pointed out and the student is given practice in their use: the "RETURN" key for sending typed messages to the computer, "RUBOUT" for "erasing" single characters, and upper case "2" (i.e., quotation mark key) for cancelling all of the characters that have been typed on a single line. The student is also told that the computer will present an asterisk when it is ready to receive his reponse, which may take one of two forms, multiple choice or constructed; the latter may be numerical or literal. Finally, the student is introduced to the location of the keys for symbols used in typing arithmetic expressions, e.g., *, -, /, +, for multiplication, subtraction, division, and addition, respectively, and is given practice in their use. Also covered is the use of parentheses to avoid ambiguities in arithmetic expressions and to set off arguments for a particular function, SQRT(X), for the square root of the number X. (Other functions are introduced in a lesson called DESCRB, designed to follow INTRO. DESCRB is described below.)

The student is taught to distinguish between a response, on his part, consisting of a mathematical expression, and the numerical evaluation of that expression. For the latter, he must preface his expression with "←" to enter PLANIT's CALC mode; he uses "↑" to return to the lesson execution mode with his numerical results, so that they may be judged correct or incorrect. The lesson contains ten practice problems requiring entry to, and return from, the CALC mode.

Styles Employed

The instructional style employed in this lesson consists, for the most part, of presentation of short expository passages followed immediately by questions or exercises designed to test the student's understanding of these passages. Initial failures to comprehend, as determined from his answers, are treated by giving the student the correct answer and advancing him in the lesson. Repeated failure results in a stepping back to material of lesser difficulty, and if the student still cannot supply correct answers, he is advised to seek outside help. The student has no control over the presentation sequence except near the end of the lesson, where he may elect to do only a subset of the exercises offered him.

Illustrations

Figures 2, 3, and 4 on the following pages are excerpts from a student's teletypewriter record of his work with INTRO. The student's replies are underlined.

```
                    = LESSON INTRO =

                 **THE 'RETURN' KEY**

IND THE 'RETURN' KEY ON YOUR TELETYPE -- IT SHOULD BE ON THE RIGHT
 SIDE OF THE KEYBOARD.  IF YOU DON'T SEE IT, GET HELP.  ONCE YOU
 HAVE FOUND IT, TYPE IN THE WORD 'YES' AND STRIKE THE 'RETURN' KEY.
 (DON'T WORRY ABOUT SPELLING.)

YES
OOD.


            **ALWAYS WAIT FOR THE ASTERISK**

OU WILL NOTICE THAT AN ASTERISK PRECEDES YOUR ANSWER. BEFORE YOU
 CAN SEND A MESSAGE YOU MUST RECEIVE AN ASTERISK FROM THE COMPUTER.
 THE ASTERISK TELLS YOU TO 'GO AHEAD.'  AFTER YOU HAVE TYPED IN
 YOUR ANSWER OR INSTRUCTION, HIT THE 'RETURN' KEY TO RELEASE THIS
 INFORMATION INTO THE COMPUTER.  DO YOU UNDERSTAND?  YES/NO

*NO
N'T KID ME... ANSWERING 'NO' INDICATES THAT YOU UNDERSTAND.


        **  'DOUBLE QUOTES' CANCELS A LINE    **

TYPING IN INFORMATION OR ANSWERS 'FILLS A BUCKET' IN A SENSE. TO
 'SPILL THIS BUCKET' OR RELEASE THE INFORMATION, THE 'RETURN' KEY IS
 USED. IF, HOWEVER, YOU HAVE FILLED THIS 'BUCKET' AND DECIDE THAT YOU
 DON'T WANT TO RELEASE THE INFORMATION, HITTING THE 'DOUBLE QUOTATION
 MARK' KEY (UPPER CASE 2) WILL CANCEL THE COMPLETE LINE OF INFORMATION
 JUST TYPED. THE CARRIAGE WILL RETURN AUTOMATICALLY GIVING YOU A
 CLEAN LINE; COMMENCE WITH YOUR CORRECTED LINE IMMEDIATELY SINCE NO
 NEW ASTERISK WILL BE GIVEN TO YOU.

 FOR PRACTICE, COPY THESE LETTERS INCORRECTLY, USE THE DOUBLE QUOTES
 ON THE UPPER CASE '2', AND THEN TYPE THEM AGAIN CORRECTLY:

        PQHNF

*PQWNF
WRONG, TRY AGAIN.

*PQHNF"
PQHNF
YOU TYPE WELL....
```

FIGURE 2. INTRO Excerpt

(This illustration is taken from the very beginning of the lesson.)

```
                    **MATHEMATICAL COMPUTATIONS**

IT IS OFTEN DESIRABLE TO PERFORM PARTS OF A NUMERICAL PROBLEM OR TO
  KNOW THE VALUE OF AN ALGEBRAIC EXPRESSION BEFORE TYPING IN THE
  FINAL ANSWER.

  TO MEET THESE NEEDS, WE HAVE MADE AVAILABLE A FEATURE CALLED THE
  'CALCULATION MODE'. TO GO INTO THIS MODE AFTER YOU HAVE RECEIVED AN
  ASTERISK FROM THE COMPUTER, YOU MUST TYPE THE HORIZONTAL ARROW '←'
  ON THE UPPER CASE 'O'.

  AT THIS POINT YOU MAY DO AS MUCH CALCULATING AS YOU DESIRE, HOWEVER,
  YOU CANNOT ENTER A COMPUTED ANSWER TO THE LAST QUESTION WHILE IN
  THIS MODE.

  TO ENTER THE COMPUTED ANSWER, YOU MUST USE THE VERTICAL ARROW '↑'
  AND TYPE YOUR ANSWER IMMEDIATELY TO THE RIGHT.  THIS ARROW,'↑' ON
  THE UPPER CASE 'N' TAKES YOU OUT OF THE CALCULATION MODE.

  HOW DO YOU ENTER THE 'CALC' MODE? (A OR B, BELOW?)

A TYPE IN A VERTICAL '↑' ARROW.
B TYPE IN A HORIZONTAL '←' ARROW.

*←

*↑
ENTER YOUR ANSWER.

*TYPE IN AVERTICAL ARROW

CHOOSE ONE OF THE ABOVE LETTERS.
*A
NO, YOU WOULD TYPE THE HORIZONTAL '←' ARROW.
```

FIGURE 3. INTRO Excerpt

(Note the student's struggle to find the correct mode of response to a multiple choice question.)

```
EVALUATE THE EXPRESSION '6**7'?

   IF YOU HAPPEN TO GET STUCK IN THE CALCULATION MODE, GET SOME HELP.
   WHEN YOU HAVE FINISHED YOUR COMPUTATION, COMPARE YOUR ANSWER AND
   TYPE IN THE APPROPRIATE LETTER.

A 279936
B 1679616
C 10077696
D 60466176

*←6*6*6*6*6*6*6
279936

*↑A
EXCELLENT
```

FIGURE 4. INTRO Excerpt

(The symbol '**' is used for raising a number to a power but this student chose to raise 6 to the 7th power through multiplication.)

2. DESCRB

DESCRB* is a CAI lesson on descriptive statistics designed for undergraduates in beginning courses on statistical inference. A student will require from 1 to 2 hours to complete his work with DESCRB.

Prerequisites

It is recommended that the student complete the lesson INTRO and read Chapters 3-5 in Guilford [6] prior to taking DESCRB. In addition, the student will need to refer to two handouts (containing a frequency table, a histogram, and a polygon shown on pages 122 and 123, Appendix 4) in order to answer certain questions posed by DESCRB.

Objectives

The objectives of DESCRB are (1) to review certain concepts of descriptive statistics and (2) to give the student supervised practice in the use of PLANIT's CALC mode. He uses this mode to do statistical exercises.

*See Appendix 4 for more details on DESCRB.

Content

Topics included in the descriptive statistics instruction are: grouping of data, frequency distributions, histograms, frequency polygons, measures of central tendency and variability, and computational techniques. In addition, the conventions for using CALC are presented and several CALC functions (e.g., summing, summing squares, etc.) are provided to simplify the student's computational tasks in the exercises.

Styles Employed

The style primarily consists of tutorial sequences followed by practice exercises. Several criterion questions are also included to determine whether a particular student needs the available remedial or repeated instruction.

Illustration

Figure 5 shows a sample of teletypewriter dialogue between a student and the lesson where the student is being shown the effect of calculating deviations of scores from some point other than the mean. Notice that the student is asked to supply that point.

The typed messages prefixed with an asterisk are responses supplied by the student. Underlining has been added to designate his responses in these sample records.

```
     LET'S GENERATE ANOTHER COLUMN OF SCORES AND A COLUMN OF
 DEVIATIONS FROM THE MEAN.

        SCORES    DEVIATIONS

ITEM    COL. 1    COL. 2
  1-     82        26.8750
  2-     26       -29.1249
  3-     94        38.8750
  4-      3       -52.1249
  5-     62         6.8750
  6-     28       -27.1249
  7-     93        37.8750
  8-     53        -2.1249

     THE DEFINITION OF THE VARIANCE IS; 'THE SUM OF THE SQUARED
DEVIATIONS FROM THE MEAN, DIVIDED BY N'.  SO IN ORDER TO COMPUTE
THE VARIANCE OF THE ABOVE SCORES WE SQUARE EACH VALUE IN COLUMN 2,
SUM THE SQUARES, AND DIVIDE BY THE NUMBER OF VALUES IN COLUMN 2.

COMPUTE THE VARIANCE OF THE SCORES IN COLUMN 1.  (REMEMBER THE
FUNCTIONS YOU LEARNED IN YOUR FIRST LESSON.)

* •SS(2)/8
1002.6066

*↑1002.6066
 VERY, VERY GOOD.


WHAT IS THE STANDARD DEVIATION OF THE SCORES?

* •SQRT(SS(2)/8)
31.6640

*↑31.664
 VERY GOOD. YOU RING THE BELL.

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
I'LL MARK THIS PLACE BECAUSE YOU WILL HAVE OCCASION TO LOOK BACK
AT YOUR ANSWERS ABOVE.
```

FIGURE 5. An Excerpt from a Student's Dialogue with DESCRB (Sheet 1)

```
     NOW LET'S SEE WHAT HAPPENS WHEN THE DEVIATIONS ARE TAKEN
FROM SOME POINT OTHER THAN THE MEAN.  TYPE IN A NUMBER BETWEEN
5 AND 15, I WILL ADD THIS VALUE TO THE MEAN AND COMPUTE A NEW
COLUMN OF DEVIATIONS.

* 12
O.K.  HERE ARE YOUR NEW DEVIATIONS
        SCORES     DEVIATIONS

ITEM     COL. 1    COL. 2
 1-      82         14.8750
 2-      26        -41.1249
 3-      94         26.8750
 4-       3        -64.1249
 5-      62         -5.1249
 6-      28        -39.1249
 7-      93         25.8750
 8-      53        -14.1249


     NOW APPLY THE VARIANCE FORMULA TO THE VALUES IN COLUMN 2 AND
TYPE IN YOUR ANSWER.

*-SS(2)/8
1146.6053

*↑114.6053
 NO, TRY AGAIN.

*↑1146.6053
 VERY GOOD.


     COMPARE THIS 'VARIANCE' WITH THE VARIANCE YOU GOT WHEN THE
DEVIATIONS WERE TAKEN FROM THE MEAN.

TYPE IN THE LETTER ASSOCIATED WITH THE TRUE STATEMENT.

A  THE VARIANCE IS SMALLER WHEN THE DEVIATIONS ARE TAKEN FROM THE MEAN.
B  THE VARIANCE IS LARGER WHEN THE DEVIATIONS ARE TAKEN FROM THE MEAN.
C  THE VARIANCE IS NOT CHANGED BY CHANGING THE DEVIATIONS.

*A
 THAT'S RIGHT.


     RECALL THAT 'VARIANCE' WAS DEFINED AS THE SUM OF SQUARED
DEVIATIONS ABOUT THE MEAN DIVIDED BY THE NUMBER OF OBSERVATIONS
IN THE SAMPLE.  YOU HAVE JUST SEEN A DEMONSTRATION OF ANOTHER
IMPORTANT PROPERTY OF THE SAMPLE MEAN, SOMETIMES CALLED THE
'LEAST SQUARES' PROPERTY.


LEAST SQUARES PROPERTY:  THE SUM OF SQUARED DEVIATIONS FOR A
SAMPLE IS MINIMIZED (I.E., LEAST) WHEN THE DEVIATIONS ARE MEASURED
FROM THE SAMPLE MEAN.  IF YOU SUBSTITUTE ANY OTHER VALUE FOR THE
SAMPLE MEAN THE SUM OF SQUARED DEVIATIONS WILL INCREASE.
```

FIGURE 5. (Sheet 2)

3. PROB1

PROB1 is designed as the first in a planned series of CAI lessons for review of the basic ideas in elementary probability theory. (Thus far it is the only lesson in the series that has been prepared.) This lesson steers a middle course between an abstract mathematical presentation and an intuitive treatment of the probabilistic concept of event from the standpoint of elementary ideas about sets. Additional lessons in the series are planned that will take up such notions as independence, conditional probability, and Bayes' theorem.

Nine undergraduates have used preliminary versions of various segments of PROB1. This experience indicates that the lesson described here could be completed in $1\frac{1}{2}$ to $2\frac{1}{2}$ hours by students enrolled in a beginning statistics course.

Prerequisites

It is assumed the student has had two or three hours of classroom discussion or lectures devoted to such concepts as sample space (universal set), simple events (points in the sample space), composite events (sets of points) and the Boolean operations of union, intersection and complementation on composite events. It is further assumed that he has completed readings on probability in his statistics textbook.* The student also should have completed the CAI lesson INTRO before beginning PROB1.

Objectives

PROB1 provides a review of the concept of event, which is usually treated too hastily for most students in a beginning statistics course. It also attempts to strengthen the student's ability to translate from a verbal description of an event to its representation as a set of "points," i.e., a set of possible outcomes of a random experiment.

Content

"Simple" (also called "elementary") and "composite" events are defined and illustrated in coin-and dice-tossing experiments. Then more abstract representations of these two types of events are presented and used in place of verbal descriptions for possible outcomes of random experiments. Elementary ideas from set theory are introduced, such as methods of defining sets, the universal set, the null set, and unions and intersections of sets. Toward the end of PROB1, probability assignment is treated in the elementary case where each point in a sample space is assigned equal probability, (i.e., is "equally likely"). The student is then asked to determine the probability of occurrence of various composite events by counting the "points" or "simple events" that are included in these composite events.

*For example, Chapter 3 in the text, Elementary Statistics by Paul G. Hoel, John Wiley and Sons, New York, 1966.

The lesson concludes by introducing the concept of mutually exclusive events, the addition theorem for probability, and applications of both.

Styles Employed

PROB1 uses a tutorial style. Short descriptive or expository passages are followed by questions to the student. These require replies consisting of a word or two, or more frequently, a number. The student who replies incorrectly is usually given a hint designed to suggest the correct answer and another opportunity to reply. If he continues to "miss" he is given the correct answer and advanced in the lesson. At certain points in the lesson, if his performance has been extremely poor, he is advised to "sign off" and seek outside help.

The lesson is entirely computer administered with one exception. The student is asked some questions which refer to Venn Diagram representations of sets. The required diagrams are shown below.

Illustrations

Figures 6 and 7 are the aforementioned Venn diagrams. Figures 8 and 9 provide brief excerpts from a student's dialogue with PROB1. The latter are teletypewriter records where only the student's replies are underlined.

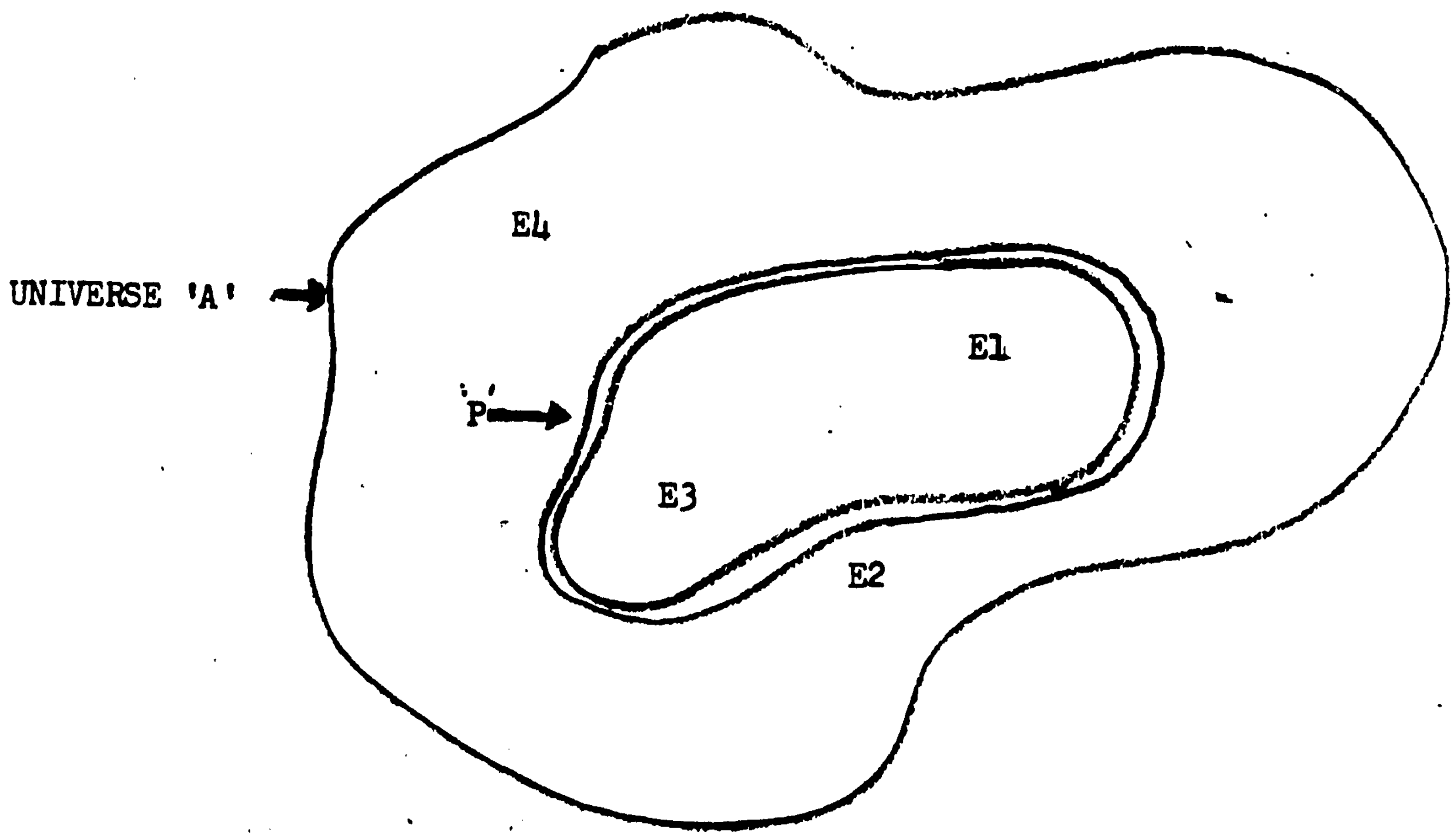


FIGURE 6. Venn Diagram of Universe 'A' with Set 'P'

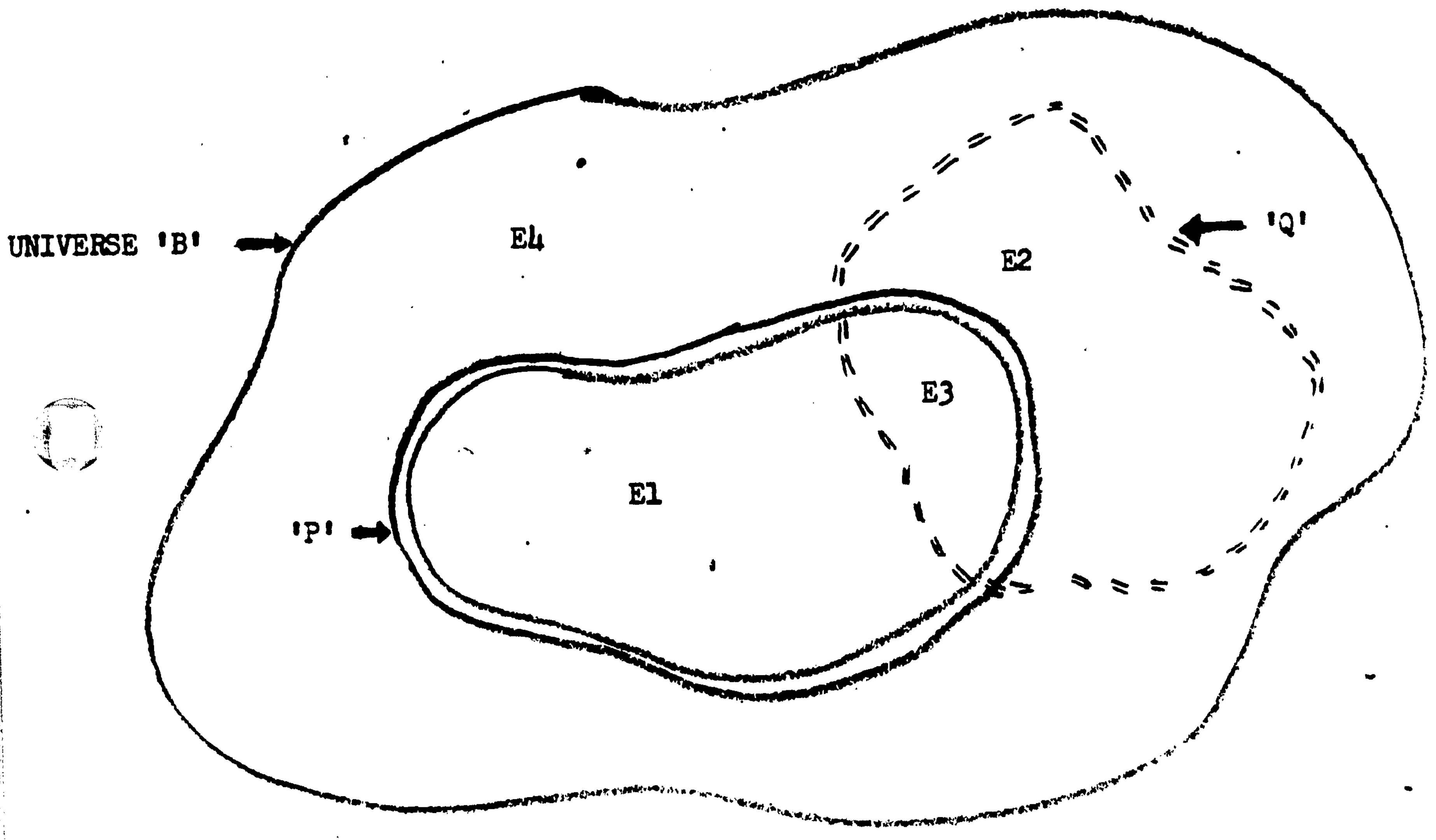


FIGURE 7. Venn Diagram of Universe 'B' with Sets 'P' and 'Q'

```
            LESSON PROB1--ELEMENTARY PROBABILITY

PROBABILITY IS THAT BRANCH OF MATHEMATICS WHICH TREATS OF EVENTS
  AND NUMBERS ASSOCIATED WITH THESE EVENTS,CALLED PROBABILITIES.
  IN PROB1 WE WILL CONCENTRATE ON THE CONCEPT OF EVENT. LATER ON,
  IN PROB2, WE WILL CONSIDER PROBABILITIES.

TO THE PROBABILIST ALL EVENTS FALL INTO ONE OF TWO CATEGORIES:

             1. SIMPLE EVENTS
             2. COMPOSITE EVENTS


        SIMPLE EVENTS-THE SAMPLE SPACE

IF YOU TOSS A COIN, AS A PROBABILIST, YOU ASSUME THAT ONE OF TWO
  OUTCOMES WILL RESULT,HEADS OR TAILS,AND YOU WOULD CALL EACH OF
  THESE TWO DIFFERENT RESULTS A 'SIMPLE EVENT'.

IF YOU WERE TO TOSS A SINGLE COIN TWICE, HOW MANY DIFFERENT OUT-
  COMES ARE POSSIBLE RESULTS OF YOUR RANDOM EXPERIMENT?

*2
NO. HOW MANY DIFFERENT RESULTS IN ALL , ARE POSSIBLE
 IF YOU CONSIDER,E.G., 'HEADS,HEADS' A SINGLE POSSIBLE OUTCOME,
'TAILS,HEADS' ,ANOTHER, ETC.---TRY AGAIN.

*3
VERY GOOD. SUPPOSE HOWEVER YOU WISH TO DISTINGUISH 'HEADS ,TAILS'
FROM 'TAILS,HEADS'---HOW MANY DIFFERENT RESULTS IN ALL THEN?

*4
GOOD.


SUPPOSE YOU TOSS A COIN THREE TIMES? HOW MANY DIFFERENT RESULTS ARE
POSSIBLE FOR THREE TOSSES?

*6
NO--FOR TWO TOSSES WE HAD FOUR DISTINCT POSSIBLE OUTCOMES:

  H H -
  H T -
  T H -
  T T -

THE RESULT OF THE THIRD TOSS WAS LEFT OUT ABOVE.THINK AND TRY AGAIN.

*8
CHECK.
```

FIGURE 8. PROB1 Excerpt

```
<<THE POOL SHARK>>

A POOL SHARK CLAIMS HE CAN TELL THE WEIGHT OF A CUE WITHOUT INSPECT-
ING THE LABEL. YOU DECIDE TO TEST HIS ABILITY USING THREE CUES
  WHICH WEIGH 2 LB., 2 LB. 3 OZ., AND 2 LB. 5 OZ., RESPECTIVELY.
  YOU ASK HIM TO PLACE THE LIGHTEST CUE IN RACK NUMBER 1, THE NEXT
  HEAVIEST IN RACK NUMBER 2, AND THE HEAVIEST IN RACK NUMBER 3 AFTER
  MASKING THE LABELS.

FOR THIS EXPERIMENT, HOW MANY POSSIBLE OUTCOMES ARE THERE?

*9
NO, TRY AGAIN.  (USE NUMBERS.)

*6
YES.

HOW MANY OF THESE POSSIBLE OUTCOMES OR ELEMENTARY EVENTS MAKE UP THE
  COMPOSITE EVENT, 'AT LEAST TWO CUES ARE CORRECTLY ASSIGNED TO THE
  RACKS'?

*3
WRONG, TRY AGAIN.

*2
NO.  THE CORRECT ANSWER IS ONE.

IS THIS EVENT IDENTICAL TO THE EVENT 'ALL THREE CUES ARE CORRECTLY
  ASSIGNED?

YES
CORRECT.

SUPPOSE THAT THE COMPOSITE EVENT IS THAT ONE AND ONLY ONE CUE IS
  CORRECTLY PLACED; HOW MANY ELEMENTARY EVENTS MAKE UP THIS COM-
  POSITE EVENT?

*3
YES.

HOW MANY ELEMENTARY EVENTS MAKE UP THE EVENT, 'AT LEAST ONE CUE IS
  CORRECTLY PLACED'?  IF YOU NEED HELP, SAY 'I DON'T KNOW.'

*I DONT KNOW
ALL RIGHT, LET'S DEVELOP IT.
```

FIGURE 9. PROB1 Excerpt

(This problem is presented to the student in the last third of the lesson and is designed to test whether he has assimilated the important ideas in PROB1.)

4. PROGM

PROGM* is a CAI lesson designed to introduce students in elementary statistics courses to computer programming concepts and the use of the TINT** programming language as an aid to problem solving. Three levels of computer programs are involved in the presentation of PROGM: first, a time-sharing system (TSS) program that supervises all activities on the computer for as many as 25 simultaneous users; secondly, the PLANIT instructional system program (under the control of TSS) that supervises the administration of the lesson and collection of student records; and finally, the lesson itself, which is comprised of instructional frames and the TINT programming system. TINT does not normally require the supervision of PLANIT; it will operate directly under TSS. However, the fact that PLANIT can also supervise the operation of TINT allows the student to do programming exercises in this language while he is being taught to use it by PROGM. No student has yet had an opportunity to interact with the entire lesson; however, a crude extrapolated estimate of the time required to complete this lesson is five to fifteen hours.

Prerequisites

It is assumed in PROGM that the students are receiving concurrently or have previously received instruction in elementary statistics. Statistical problems are used for the programming exercises. These include computation of sample means, variances, standard deviations, sums of pair products and the Pearson product moment correlation coefficient. PROGM contains a brief review of the computational aspects of these topics for the student who needs to refresh his memory; the review is not an adequate replacement for prior statistics instruction. However, if the student knows how to interpret a computational formula for the Pearson correlation coefficient, then he will usually have the necessary statistical prerequisities.

Objectives

The objective of PROGM is to provide enough instruction and experience in the use of TINT so that the student will be capable of using it for elementary statistical computations. He must learn how to write correct TINT statements for accepting data, doing summations, expressing formulas, and communicating results. A correlation programming problem is used to test whether the student has achieved this capability, since it encompasses most of the programming techniques required in computation and data processing for commonly used statistical procedures.

*See Appendix 5 for more details about PROGM and illustrations of its use.

**TINT (Teletype INTerpreter) is a programming system for on-line use [12]. The user may type program statements while seated at a teletypewriter which communicates with a time-shared computer. He can call for immediate execution of these statements or save his program by issuing commands from a teletypewriter.

Content

PROGM is divided into fifteen topics. (Appendix 5 discusses the content for each topic.) Briefly, the concepts that are covered include:

- item names, their declaration forms and meanings,
- composition of a statement and role of the equal sign within it,
- conditional expressions, both simple and compound,
- sequence control through branch commands,
- the construction and application of loops,
- arithmetic conventions, and
- TINT command conventions.

An orientation precedes the actual lesson material for those who signify a need for it, and programming exercises are introduced within the lesson, increasing in frequency and complexity toward the end. The correlation problem is the final exercises.

Styles Employed

The pedagogic styles in PROGM are expository, tutorial, and exercise. An effort was made to minimize the expository material because of the inadequacies of the teletype printer for displaying large amounts of text. Therefore, typed tutorial dialogues and TINT exercises dominate the instruction.

Normally, PROGM assumes the initiative during the dialogue with the student. Depending on the student's response to the question or message from PROGM, PLANIT selects the frame to be executed next. However, there are five exceptions to this. If the student types one of the following messages:

- ← REVIEW
- ← EXPLAIN
- ← EXAMPLE
- ← TINT
- ← QUIT

in place of an "expected" response, he can seize the initiative temporarily. This permits him to review topics, to request explanations and examples of the use of TINT commands, to access the TINT programming system (an abbreviated version) for use in programming exercises, or to signify that he wishes to terminate an instructional session. The student may return to the normal lesson sequence when his request has been satisfied, except for the request to terminate the session. (A more detailed discussion of these student lesson control features is contained in pages 167-172 of Appendix 5.)

Illustration

The teletypewriter record sample in Figure 10 shows a student's first attempt to use the TINT language and interpreter in an exercise. The student usually makes his teletypewriter response following an asterisk when PLANIT is controlling the instructional sequence. However, when the student is actually employing the TINT interpreter, he types his messages following the TINT signals:

"*ENTER COMMAND," or

"*1.00"

where the latter designates a label for a line (the first in this case) of program code.

```
   WOULD YOU LIKE TO DIGRESS FOR AWHILE AND ACTUALLY TRY THIS PROGRAM
IN TINT (IT WILL MAINLY INVOLVE SOME EXTRA TYPING)?

*OK


   OK.  YOU WILL NEED TO KNOW THREE TINT COMMANDS...

     ?START
     ?EXECUTE
     ?READY

TINT COMMANDS ARE ALL PREFACED WITH A QUESTION MARK (?).  YOU WILL
RECEIVE A MESSAGE, *ENTER COMMAND.  ENTER...  ?START.  TINT WILL
NUMBER THE LINES FOR YOU AS YOU TYPE THE PROGRAM.  WHEN YOU HAVE THE
PROGRAM TYPED AND TINT IS ASKING FOR ANOTHER LINE, TYPE...  ?EXECUTE.
YOU WILL THEN BE OPERATING YOUR OWN PROGRAM.  YOU MAY RUN IT AS OFTEN
AS YOU WISH BY SIMPLY TYPING THE COMMAND, ?EXECUTE, AGAIN.
YOU MUST RECOGNIZE THAT YOU WILL NOT BE OPERATING PLANIT THEN, BUT
YOU WILL BE OPERATING TINT.  WHEN YOU ARE READY TO RETURN TO PLANIT
AGAIN, TYPE...  ?READY.

   THE FIRST PART OF YOUR SHEET SHOULD LOOK LIKE THIS:

*ENTER COMMAND
?START

*  1.00 REAL NUM;
*  2.00 READ NUM;
*  3.00 IF NUM LS 0;
*  4.00 NUM=-NUM;
*  5.00 PRINT NUM;
*  6.00 ?EXECUTE

     NUM = ?

I AM GOING TO GET TINT FOR YOU NOW.  DON'T FORGET TO TYPE... ?READY
WHEN YOU WANT TO RETURN.  THIS WILL TAKE A FEW SECONDS.  STANDBY.
```

NOTE: About five to fifteen seconds elapse here while the TINT interpreter is "loaded into PLANIT" from TSS disc memory and then the student begins to prepare a TINT program. TINT signals the student to begin with the message, "*ENTER COMMAND."

FIGURE 10. PROGM Excerpt (Sheet 1)

(Prior to this excerpt the student has been shown the essential statements in a program which computes the absolute value of a number.)

```
*ENTER COMMAND
?START

*   1.00 REAL NUM;
*   2.00 READ NUM;
*   3.00 IF NUM LS 0;
*   4.00 NUM=-NUM;
*   5.00 PRINT NUM;
*   6.00 ?EXECUTE

      NUM = ? 735
+.735000000E+003
*EXECUTION COMPLETE


*ENTER COMMAND
?EXECUTE

      NUM = ? -399
+ .399000000E+003
*EXECUTION COMPLETE


*ENTER COMMAND
?READY
```

Note: With the command "?READY" control over instruction is returned to PLANIT and normal dialogue resumes.

```
   YOU PROBABLY NOTICED A DIFFERENCE IN THE FORM USED TO PRINT YOUR
NUMBER.  IT WAS PRINTED IN SCIENTIFIC NOTATION.  LATER I WILL SHOW
YOU HOW TO MAKE IT PRINT IN OTHER FORMS.  DID YOUR PROGRAM WORK AS YOU
EXPECTED IT TO (OTHER THAN THE NUMBER FORMAT)?

*YES
FINE.  YOU'RE A TINT PROGRAMMER ALREADY.


   YOU SAW LINE NUMBERS ATTACHED TO THE STATEMENTS IN A COMPUTER
PROGRAM.  THEY ARE NOT PART OF ANY STATEMENT AND HAVE NO FUNCTION IN
THE OPERATION OF THE PROGRAM.  THEY ARE USED FOR EDITING.  WE WILL
IGNORE THEM FOR NOW.
```

FIGURE 10. (Sheet 2)

III. CAMPUS TRIALS

A. DESCRIPTION OF TRIAL CONDITIONS

Two field trials of lesson materials produced with PLANIT were conducted. The first took place during July 1966 on the campus of the University of Southern California (USC) where a teletypewriter station was installed and linked by DATAPHONE to the TSS computer in SDC's Santa Monica facility. Seven undergraduate students, the entire summer session enrollment in Psychology 274* participated in shakedown tests of INTRO, DESCRB, and portions of PROBl. The students, with the exception of one, who worked alone, worked at the teletypewriter in pairs during each of four instructional sessions; the sessions varied in duration from 40 minutes to 1½ hours and occurred at one-week intervals. An SDC project representative was always on hand to establish DATAPHONE communications, load the required programs, and intercede when the students asked for assistance. The students who worked in pairs were allowed to consult with each other before deciding on a response to questions posed by the computer and alternated as typists, one student performing for the first half and the other for the second half of a session. (The use of pairs of students rather than individuals was not deliberate but necessary because of scheduling conflicts.) None of these students had any prior computer experience, and before the tests began they were introduced to time-shared computer systems and CAI in a one-hour lecture; they were also assured that their performance in these test sessions would not affect course grades. The lessons were frequently revised by SDC authors after each test session to remove discovered deficiencies before the next session (usually 24 hours later). Thus, the last student to "see" a particular lesson often worked with a much improved version. No attempt was made to pre-test or post-test these students on any of the material covered in their CAI instruction. The object in these tests was to check out PLANIT (which had been released as an operational program for author use only six weeks before the tests began) and to remove "bugs" from the prepared lesson material.

The second test series was conducted between March 27 and May 31, 1967, at San Fernando Valley State College (SFVSC), Northridge, California. The test conditions were similar to those in the USC trials except that the number of

*Psychology 274 is a first course in statistics required for psychology majors but also attended by other students. The students in these trials were juniors and seniors majoring in sociology, education and pre-dentistry, in addition to psychology. Dr. Ray Berger, who has taught this course for several years, was the instructor and used the text by Guilford [6] .

students participating* was increased to 14, and all students worked individually at the single teletypewriter installed on campus in Northridge. PLANIT was considerably improved and augmented in the interim between the two test periods making it possible to write and present more elaborate lessons such as PROGM. Each of the 14 students was scheduled for one hour of instruction at the teletypewriter per week for eight weeks, but only 75% of the time when students were present was devoted to actual instruction.** Again, an SDC representative was present to assist the student; a log was kept in which impressions and observations were recorded by the representative. An improved version of INTRO was used to introduce these students to the use of the teletypewriter and PLANIT communication conventions and procedures. Then the students began computer programming instruction via the lesson PROGM. There was no attempt to measure any gain in knowledge through pre- and post-testing since this was a shakedown test of PROGM. (However, only two of the 14 students had any prior computer programming experience or training). PROGM, unlike the lessons DESCRB and PROB1, was not designed to augment conventional classroom instruction or textbook readings. It represents an attempt by the SDC authors to provide a completely self-contained CAI sequence for on-line computer programming.

These two tests of CAI provided the SDC staff with ample opportunities to observe both the problems and benefits that accompany CAI. The observations and impressions gathered by SDC monitors during these trial runs with PLANIT are discussed below from the standpoint of students, instructors, and authors of CAI lessons; the roles of instructor and author with respect to CAI lessons were assumed entirely by the project staff who report their own experiences in these roles.

B. PROBLEMS ENCOUNTERED

1. By the Student

By far the most serious problems encountered by students during their use of PLANIT (and STAT) stemmed from failure to appreciate the meaning of certain messages that would appear on the teletypewriter. For example, some students sent their messages to the time-shared system executive program (TSS Exec) rather

*These students were volunteers enrolled in Professor Robert Dear's section of Psychology 320, Statistical Methods in Psychology, a first course in statistics for undergraduates majoring in psychology. Dr. Dear used Hoel [8] as the course textbook. Satisfactory performance on a qualifying examination including algebra, descriptive statistics, and elementary uses of the normal probability distribution was a condition for enrollment in Psychology 320.

**Students were actually available for CAI for 97 hours during the eight-week period. Of these hours, 15 were lost due to problems involving software, hardware components of the computer system, or the DATAPHONE communications link and an additional eight hours could not be used because the required lesson material was not ready.

than to PLANIT (or STAT) by inadvertently typing the exclamation point '!' (instead of 1), as the initial character in a numerical expression. In these cases, instead of an expected numerical reply from the computer the student often saw '$?', a message from TSS Exec signifying that the student's message could not be interpreted. When the student repeated his message, typed correctly this time, he failed to realize he was still communicating with TSS Exec, not with STAT or PLANIT, and received the same reply from the puzzled computer '$?'. (In order to reestablish communication with STAT or PLANIT the student would have had to preface his repeated message with the double quotes, ", symbol. Even those students who were aware of this procedure often failed to employ it because they did not realize the nature of their difficulty.) SDC monitors interceded in such cases so that the student could continue with his instruction.

A communication problem of a different sort arose for one student who typed 'SQ5T(5)' instead of "SQRT(5)" while communicating with PLANIT's CALC mode. He wanted to obtain the numerical value of the square root of 5 but the computer replied, 'ENTER VALUES (COMMAS BETWEEN) FOR: SQ5,T. Surprised by this unexpected reply the student inspected his original message and noticed that he had typed the numeral 5 in place of R; he then retyped his request correctly. The computer still did not reply with the desired numerical result, but requested a value for 'T'. An SDC monitor had to intercede, to explain that the student's original message was being interpreted as a request to evaluate the triple product of an item called "SQ5' with an item called 'T' and the number '5', and was calling for information needed for this purpose. The monitor also explained how to "cancel" the effect of the original mistyped message to CALC.

The frequent user of a time-shared system and programs operating within it learns from experience and training to cope with communication problems like these and others. Many of the students' communications difficulties could be anticipated by the project staff, and were, but it was decided not to expend more than an hour or two of each student's limited time to instruct him in communication conventions and procedures. In other circumstances, particularly where the student is to make fuller use of the CALC mode, he should have about five hours of introductory instruction including many CALC exercises.

Many students would have enjoyed a greater measure of control over the instructional sequences. For example they could not effect control in most lesson presentations to review a topic already encountered. (This review feature was included only in the lesson PROGM.) Also, through careless typing of a response, the student might set off a remedial chain of instruction that he did not require and could not stop. In other cases he could not interrupt such remedial sequences, when they were longer than necessary for his individual requirements.

Improved preparatory instruction in CAI communication conventions and procedures, more interactive experience with TSS, redesign of the teletypewriter keyboard, more cleverness in the design of lessons, and more detailed feedback messages from the computer cannot completely eliminate these problems, but will markedly reduce their frequency of occurrence.

2. <u>By the Instructor</u>

The regular instructors of the students who participated in the field tests remained almost entirely ignorant of what their students were doing at the teletypewriter stations. Many of them could not gain access to the teletypewriters without displacing one of their students. They were loathe to do this. Some instructors did try the lesson material in the evenings when the teletypewriter was free, but could not spare as many hours as would be required to fully explore a lesson, especially not lengthy lessons, like PROGM. Documentation of the CAI instructional sequence, shown in Appendices 4 and 5 for DESCRB and PROGM was not available during the field trials. Such descriptions of CAI lessons are essential for the instructor who may wish to "browse" through a lesson, as he might a textbook, before deciding whether to use it for his classes. Also, some of the lessons "advise" the student to seek help outside the computer when he has failed to master a part of the lesson. Instructors should be prepared for such visits from students, and the documentation of lessons would be essential for this purpose as well as others. In addition, in future trials (or in actual CAI operations) instructors should be free of a portion of their regular teaching chores, to have time to become at least as well oriented as their students to CAI.

3. <u>By the Lesson Authors</u>

The preparation of CAI lesson material, while greatly facilitated by PLANIT, proved to be a time-consuming task, particularly for the tutorial type of dialogue. A good student could complete a lesson like PROB1, which is almost entirely tutorial, in an hour. The author(s) of PROB1 invested approximately 200 hours in its preparation. Much of this time is consumed in building the sets of anticipated student answers, to the questions posed by the author, and planning the various paths through the lesson frames as a function of the students' answers. If a tutorial dialogue is to provide truly individualized instruction for a heterogeneous class (with respect to levels of ability and preparation in prerequisites), 200 hours is probably only a lower bound for the time an author will spend building a "one-hour" tutorial. It is clear that the tutorial form of CAI represents a heavy investment of skilled individuals' time and should be attempted only for fundamental, troublesome topics and lessons where a large number of students can profit from the instruction.

By contrast, the CAI sequences for numerical exposition of statistical properties (e.g., the properties of the sample mean in DESCRB, see page 33) or for statistical laboratory exercises (as in STAT) can be prepared in far less time; project experience indicates that some 36 hours of author time would be sufficient to prepare material of this type that would occupy a student for one hour.

C. APPARENT BENEFITS FOR THE STUDENT

1. <u>The student pays attention</u>. Throughout the PLANIT field trials in sessions at the teletypewriter that usually lasted for an hour, the student kept his

attention fixed on the learning task. This is remarkable considering the primitive conditions that the student faced, e.g., occasional TSS failure, a noisy teletypewriter. (No doubt the novelty of CAI was partially responsible for this phenomenon, but the same effect was noticed in the longer, three-hour sessions, when students used the STAT program. There, this high degree of student attentiveness was still visible even during the last of five three-hour sessions.*) In addition, the students all began their work promptly with the first computer-generated message and stayed with their learning task until the lesson was completed or the SDC monitor interceded to remind them of a class they had to attend. (Some observers of the CAI scene have commented on the negative aspects of this phenomenon, e.g., the "transfixed" student may lose contact with his fellow students, the library or professors.) There is little question that some of the students who participated in these trials would have difficulty maintaining equivalent attentiveness and concentration with the same subject matter in other learning situations, e.g., while reading a textbook or attending a lecture.

2. The student appreciates privacy and feels "freer to fail." In CAI, the student attempts to answer questions or to work exercises presented to him by the computer without the reluctance and hesitancy he often exhibits in the conventional classroom setting. He has no fears that he is wasting class time or that his peers or the instructor are "grading" him when he replies to a computer-presented question, though he may be less than sure his answer is correct. This is not to say the student finds "being wrong" a pleasant experience in CAI, but it is apparently less uncomfortable for him and less threatening than in the classroom. This advantage of CAI was noticed for some students with apparent low levels of self-confidence who were observed by an SDC monitor both in the classroom and during CAI field tests.

3. The student gets to use more varied techniques. The calculation capability in STAT or in PLANIT's CALC mode makes it possible for students in beginning statistics courses to use a greater variety of techniques. For example, non-parametric techniques are often preferable** to such parametric "work horses" as Student's-T Test for comparing two populations or the effect of two treatments. However, almost all non-parametric tests require that the sample data be ranked according to magnitude before the test criterion can be computed. The task of ranking sample data even in samples of quite modest size, say 20 to 30 observations, is time consuming and discouraging (particularly when much of the data lies in a narrow interval). The ranking function available in

*However, it should be noted that these UCLA graduate students who participated in STAT tests were paid volunteers.

**Hodges and Lehmann recommend that beginning students be taught the non-parametric Wilcoxon (sometimes called Mann-Whitney) test in place of Student's-T Test. See p. VIII in the Preface of [7].

programs like STAT and in PLANIT's CALC mode provided some students (those that tested STAT) their initial experience with the actual use of non-parametric technique for samples of realistic size.

4. Concepts that are "difficult" for beginning students can be demonstrated numerically. There are interesting statistical concepts and properties that can be demonstrated numerically for beginning, non-mathematically oriented students of statistics where a mathematical demonstration or proof would be inappropriate or easily forgotten. For example, with DESCRB the students participated in numerical demonstrations of two properties of the sample mean: that sample deviations sum to zero and the sum of these squared deviations is minimized, when deviations are measured from the sample mean. (The student was allowed to choose a number(s) within a restricted range and the two sums were automatically computed by PLANIT with the student's number(s) replacing the sample mean. The student was then asked to compare these results with those of preceding computations where the sample mean was used.) Many other concepts could have received such numerical treatment had there been more time for the SDC authors (or faculty at USC or SFVSC) to prepare the required CAI lesson material as well as auxiliary notes* which would explain and amplify the numerical results for the student.

5. CAI can play a useful supplementary role for conventional instruction. It was assumed that students would be introduced to such topics as elementary probability or descriptive statistics in the classroom and in text readings.* The tutorial instruction provided by PROB1 and DESCRB was designed to remove deficiencies in the student's understanding of this material or to revisit certain topics, rather hastily covered in class, in a more leisurely fashion. There was evidence that this supplementary role was welcomed by both students and instructors. This was especially true for PROB1, which proved to be very popular with students in the PLANIT field trials. Yet PROB1 can hardly be considered a "finished" lesson. It needs to be polished and extended. We believe its popularity stems from the need for supplementary probability instruction for which too little time** (two to four lectures, depending on the instructor) can be spared in most beginning courses in statistics. CAI, properly designed and tested, could offer the student such instruction in varying amounts according to his needs.

*There is little point in using CAI merely to have the teletypewriter present the students with large amounts of introductory and expository text or the recorded form of a professor's lecture.

**There is a tendency to introduce a greater amount and more modern treatment of probability into the beginning statistics course. This is visible in the textbooks that were used by these students, Guilford [6] at USC, and Hoel [8] at SFVSC; earlier editions of these texts devoted fewer pages to probability.

6. Magnet effect. The single teletypewriter stations that were installed at USC and SFVSC drew students who were not participating in the PLANIT lesson trials. Some students were merely curious and wished to observe a "computerized tutor" in action. Other students who had some programming experience, were eager to use the on-line programming capability that was available through time-sharing. At SFVSC several such students "mysteriously" appeared each day at the close of PLANIT trials to inquire whether they could interact with TSS. They were given a TINT programming manual [12] , a few hints about "logging in", and permission to use TSS during low-load evening periods. They made rapid progress and, on their own, worked through many of the exercises provided in the manual.

D. INSTRUCTOR AND STUDENT ATTITUDES

1. The Student's Attitude

No systematic attempt was made to measure the students' attitude at either campus during PLANIT field trials. However, it is quite clear that when the computer has something to say that is of interest to the student, he is willing and eager to pay attention. The students came to the trials on their own time without compensation, other than that provided by their CAI experience, and their attendance record was excellent. Often their experience was a frustrating one (about 15% of the time was lost during these test sessions due to some failure of TSS or the communication link between the college and the computer), but the USC students returned for further CAI tests with no exceptions. At SFVSC there was a falling off in attendance as the final examination period approached. Prior to that, only three of sixteen student volunteers failed to appear regularly.* The students were too polite to complain to their SDC monitors but did complain to their instructors and other "outsiders" about frustrations stemming from the lack of reliability in remote TSS operations, the too rapid pace of the lesson PROGM, and too little orientation to TSS operations.

2. The Instructor's Attitude

Both at USC and SFVSC a cordial reception was given to CAI by faculty members. Dr. Ray Berger, USC instructor of the class that provided students for the first PLANIT tests, expressed a desire to use STAT and any other available PLANIT lesson material with his future classes. One of his colleagues, Dr. Daniel Davis, expressed similar interest in the fall semester following the USC trials. (Some of Dr. Davis' students used an improved version of PROB1 before the teletypewriter station at USC was removed in September 1966; both Dr. Davis and these students expressed strong interest in additional CAI material for probability and statistics which, unfortunately, was not available.)

*Of these three students, two dropped out early because of an accident and family illness, respectively. The third student had a very spotty record of attendance at the trials (and in his regular statistics class).

At SFVSC, Dr. Richard Docter, chairman of the Psychology Department, and three of his colleagues, Professors Smith, O'Connell and Dear, were most interested in the catalysis that CAI might provide for curriculum innovation and reorganization. For example, Dr. Dear saw advantages that could accrue to his students in a course on design and analysis of experiments from an on-line matrix arithmetic capability. (Such a capability could be provided through expansion of PLANIT's CALC mode or through special subroutines that might be written and executed with TINT.) Dr. O'Connell was interested in exploring the opportunities in CAI for closer ties between courses in experimental psychology and the design and analysis of experiments course. A computer-based statistics laboratory could serve both areas of instruction.*

In summary, the faculty members who were encountered during these student trials were quite willing to learn about CAI and to experiment with it as authors and instructors. However, the resources of the project were inadequate and insufficient (also the professors did not have enough time when they were free from regular duties during the project) to give them opportunities for either activity. The discussion that follows includes recommendations for greather involvement of teachers in the development and application of CAI.

IV. CONCLUSIONS AND RECOMMENDATIONS

This study provided the staff with valuable experience of several types: design and implementation of PLANIT; the use of a time-sharing system (TSS), together with PLANIT, to build and present CAI lessons for a variety of topics in varied styles; and observation of some 150 hours of student interaction with these lessons. This experience is translated in this section into conclusions and recommendations for (1) development of a less costly (to operate) PLANIT and its distribution to colleges and universities, and (2) given such distribution, appropriate and feasible applications of CAI in the statistics instruction of undergraduates majoring in the social sciences.

A. DISTRIBUTION OF PLANIT TO COLLEGES AND UNIVERSITIES

PLANIT proved to be a most valuable assistant on this project from the time it was released for use in the SDC TSS. In the hands of experienced computer programmers, it cut the programming time required to generate lessons at least by one-half over comparable programming with a general-purpose language such as JOVIAL. It permitted temporary and part-time assistants, who had no programming skill, to generate and edit lesson material within two weeks of their introduction to PLANIT. The enthusiasm for this CAI author language was by no means restricted

*Discussion of these ideas by SDC staff and these SFVSC faculty members resulted in an informal proposal, "The Computer and Innovation in the Undergraduate Curriculum," which has been submitted to the Office of Computer Activities, National Science Foundation.

to the project staff. Numerous requests came from other works in CAI (e.g., at the University of Michigan, at the Irvine campus of the University of California, etc.) for access to PIANIT which could not be provided. The SDC TSS was already oversubscribed and "shares" were not available even for requestors who could afford the $10 per hour SDC TSS charge. Other requestors could not tolerate this cost were shares available. Both barriers can be overcome for a large class of users through reprogramming of PIANIT to operate with relatively inexpensive computing equipment which is currently or soon to be installed on many campuses. Preliminary design figures indicate that a medium-small central processor (e.g., IBM 360/40H) could be used to operate a redesigned PIANIT quite satisfactorily for 50 to 100 users, at an hourly operating cost of one to two dollars per teletype station including the teletype rental fees. The redesign of PIANIT for such equipment (with large disc memory) would result in further economies and convenience. For example, the program could be recast into modules permitting students to employ a reduced module containing only the CALC (on-line computation) and EX (lesson presentation) modes. Lesson authors would use a larger module (requiring more of core memory) incorporating all of the modes, as in the currently existing PIANIT, including lesson building and editing as well as EX and CALC.

It is therefore recommended that such a redesign and reprogramming effort be supported by the Foundation to provide a large number of colleges and universities with what might be termed an instructors' computer utility.*

B. APPROPRIATE AND FEASIBLE CAI FOR STATISTICS

We believe that computer-student dialogues for the laboratory type of instruction and the numerical demonstration of concepts represent the most appropriate forms of CAI for statistics at this time. It is feasible to provide only the more routine aspects of tutorial assistance with CAI (at present and probably for many years to come); a restricted use of tutorial dialogue is therefore recommended.

1. The Computer-Based Statistics Laboratory

There is little question that the student will enjoy the profit from the computational convenience of such a laboratory and the immediate feedback about the correctness of his submitted answers. The student can be introduced to the teletype and PIANIT calculation and communication procedures in approximately the same amount of time currently devoted to his instruction in the use of desk calculators. The instructor would be spared the drudgery of providing data for exercises, computing correct answers, and keeping records of student progress and performance. The cost of developing such instruction and its maintenance

*SDC Proposal 68-195, Instructors' Computer Utility, discusses the design and implementation of this utility, and will be submitted to the Office of Computer Activities, National Science Foundation, in December 1967.

is modest. We estimate that a subject matter expert, working full-time, could build a statistics laboratory instructional sequence, e.g., an improved version of STAT without the disadvantages discussed earlier in this report on pages 15-17 with PLANIT in three to six months; this would provide a full semester of laboratory work on the basis of one or two hours at the teletype per student per week for 16 weeks. The cost of operating this "laboratory," at the estimated rate of one or two dollars per teletype hour, could be recouped from student fees quite comparable to those charged for the use of college chemistry laboratories.

2. Numerical Exposition of Statistical Concepts

Much has been written about the benefits to be derived from Monte Carlo (sampling) experiments in teaching statistics, especially to students whose mathematical training is limited. Virtually every introductory text suggests that the student toss dice or coins and record the frequencies he observes for various outcomes. The computer not only offers the advantages of faster "tossing," automatic recording and tallying of results, virtually painless calculation of theoretical frequencies (probabilities) for comparison with empirical results, but perhaps most important of all, it can easily "produce" coins whose probabilities for coming up heads can be any desired number from zero to one. In other words, the student (or his instructor) can vary parameters (e.g., sample size, population mean) and explore the effect of such variations on both the samples that are generated and theoretical probability distributions. It is ironic that the very impact that such numerical demonstrations of probabilistic (and statistical) concepts would have on normal classroom instruction is what prevented, to a great extent, the experimentation with such CAI sequences in this project. The student must be prepared in advance for such demonstrations through specially prepared "notes" or texts and in class discussions if he is to profit fully for them. There was not sufficient time and personnel to integrate the CAI experiments with the classroom instruction of the students who participated in the PLANIT lesson field trials. It is interesting to note that the example of a numerical demonstration for correlation shown on page 25 was quite instructive even to several experienced users of statistics at SDC. It is this type of CAI that will most likely produce significant increases in the student's understanding of concepts and in his ability to think "statistically." Fortunately, as in the case of the laboratory exercises, the preparation of such expository demonstrations is not very time-consuming for the knowledgeable (in subject matter) CAI author. A demonstration that could occupy a student for 10 or 15 minutes can be constructed by a "non-programmer" statistician (e.g., one of the authors of this report, J. Rosenbaum) in four or five hours, from initial concept to "debugged" lesson.

It is therefore recommended that studies leading to the development of both laboratory exercise and numerical demonstration sequences be supported by the Foundation. This activity can and should proceed concurrently with the redevelopment of PLANIT.* The PLANIT language specifications are firm so that any lesson material prepared and tested using the current SDC TSS facility would be executable with the redeveloped PLANIT system as well. Such lesson materials would then be ready for use in the various colleges and universities when the redeveloped PLANIT is made available to them. Not only would these lessons be useful for the instruction of students, but they would serve to exemplify lesson-building techniques and PLANIT's capabilities for the faculties at these institutions.

3. Restricted Tutorial Instruction

Perhaps the most valuable feature of PLANIT (as a language for specification of CAI lessons) is the power it affords the lesson designer for prescription of communication between student and computer. This feature is exploited in several ways in preparing and executing lessons of the two types discussed above. In demonstrations and laboratory exercises, much of the content of messages sent to the student is numerical and computer-generated from specifications (e.g., sample size and population parameters for pseudo-random data, the definition of a mathematical function, etc.) stored by the lesson designer in lesson frames, or supplied by the student from his teletype, or combinations of both. The pedagogic value of such lessons is considerably enhanced if prestored verbal messages, designed to provide tutorial services, are also presented to the student from time to time. The student can then be interrogated by the machine to determine if he wishes to continue with a demonstration, whether he is ready to go on to an exercise involving a concept just demonstrated, or whether he wishes to review some parts of available lesson material, and depending on his response, appropriate computer behavior can ensue; the student can be enlisted into guided active participation in a numerical demonstration by asking him to supply or to vary some required parameter values; the student can seize the initiative in his dialogue with the computer by typing a request for help with an exercise and he can be furnished hints on how to proceed toward a correct solution; the student's attention can be directed to some subtleties in an exercise or demonstration after he has had sufficient exposure to the content therein. These are all examples of tutorial duties performed by an instructor when he is in a position (all too seldom in most colleges and universities) to interact with the individual student in the educational process. The deeper forms of tutorial assistance, however, require "intelligent behavior" that machines cannot yet provide (e.g., inspection of a series of calculations to determine where and why the student is going astray) or that can be only rudely approximated at great expense** (dialogue between the student and the computer to detect and explain a non sequitur in his faulty chain of reasoning.

*An informal proposal, "The Computer and Innovation in the Undergraduate Curriculum," describes such an activity and has been submitted to the National Science Foundation.

**See page 47, for a discussion of the lesson author effort required to produce a tutorial CAI lesson, PROB1.

We therefore recommend that only the more routine aspects of tutorial assistance be attempted with CAI, and mainly in lessons that are expository or of the laboratory exercise type.

V. PROJECT STAFF

Dr. Harry F. Silberman, Manager, Education and Training Staff, was Principal Investigator for NSF Grant GY-371 and supervised the full time effort of these staff members for the entire term of the study:

Mr. Joseph Rosenbaum, Project Director
Mr. Samuel L. Feingold, Senior Programming Analyst
Dr. Charles H. Frye, Human Factors Scientist

During the second year of the project, they were assisted by personnel on loan from the Defense Systems Division of the Corporation:

Mr. Alfred K. Butler, Human Factors Scientist
Mr. Fred D. Bennik, Human Factors Scientist

Two student associates, Mr. John C. Hawkins, University of Utah, and Mr. Bobby R. Brown, Pennsylvania State University, worked full time on the study during the summer of 1966 in residence in Santa Monica.

The project employed two paid consultants:

Wilfrid J. Dixon, Professor of Biostatistics and Professor of Preventive Medicine, University of California (Los Angeles)
Dr. Frank B. Baker, Department of Educational Psychology and Wisconsin Research and Development Center for Cognitive Learning, The University of Wisconsin, Madison, Wisconsin.

VI. PRESENTATIONS, PUBLICATIONS, AND PROJECT DOCUMENTATION

A. PRESENTATIONS

The PLANIT system was described and demonstrated for National Science Foundation representatives on two occasions:

- 18 October 1966, Dr. Lavon Richardson, at System Development Corporation, Falls Church, Virginia

- 1 March 1967, Drs. Glen Ingram and Arthur Melmed, at System Development Corporation, Santa Monica, California.

Similar presentations were made to various professional groups and organizations. Most notable are:

- 8 August 1966, Summer Study Group on CAI, U. S. Naval Academy, Annapolis, Maryland
- 18-19 August 1966, American Statistical Association (126th annual meeting), Los Angeles, California
- 16-17 August 1966, EDUCOM (Task Force on Educational Systems and Technology). University of Colorado, Boulder, Colorado
- 26-27 September 1966, Office of Naval Research CAI Interest Group (3rd semiannual meeting), System Development Corporation, Santa Monica, California
- 30-31 March 1967, Workshop on CAI Languages, Learning Research and Development Center, University of Pittsburgh, Pittsburgh, Pennsylvania

B. PUBLICATIONS

- Feingold, S. L., "PLANIT--A Flexible Language Designed for Computer-Human Interaction," AFIPS Conference Proceedings, Volume 31. Fall Joint Computer Conference, November 14-16, Anaheim, California. Washington, D. C.: Thompson Books. 1967. pp. 545-552.
- Rosenbaum, J. "Dialogues for Elementary Statistics Instruction via Computer," Computer Science and Statistics: Modern Instrument of Technical and Management Progress, A. F. Goodman and N. R. Mann, editors. New York, N. Y.: Academic Press (scheduled for publication late 1968).

C. PROJECT DOCUMENTATION

- Frye, C. H. and Rosenbaum, J. Student's Guide to STAT. SDC document TM-2910/000/00. March 31, 1966. 172 pp.
- Frye, C. H. and Feingold, S. L. User's Guide to PLANIT: Programming LANguage for Interactive Teaching. SDC document TM-3055/000/01. October 17, 1966. 214 pp.

In addition, three progress reports were submitted to the National Science Foundation during the project:

- H. F. Silberman and J. Rosenbaum. "Semiannual Progress Report, Computer-Based Instruction in Statistical Inference, 1 April 1966," TM-2914/001/00, System Development Corporation, May 2, 1966.

- H. F. Silberman and J. Rosenbaum. "Semiannual Progress Report, Computer-Based Instruction in Statistical Inference, 8 September 1966," TM-2914/002/00, System Development Corporation, September 19, 1966.

- H. F. Silberman. "Semiannual Progress Report, Computer-Based Instruction in Statistical Inference, 13 March 1967," TM-2914/003/00, System Development Corporation, March 13, 1967.

REFERENCES

1. Bitzer, D.L. and Easley, J.A. "PLATO: A Computer-Controlled Teaching System," Symposium on Computer Augmentation of Human Reasoning. Washington, D.C.: Spartan Books. M.A. Sass and W.D. Wilkinson (Eds.) 1965. p. 89-103.

2. Feingold, S.L. "PLANIT - A Flexible Language Designed for Computer-Human Interaction, " AFIPS Conference Proceedings, Volume 31. Fall Joint Computer Conference, November 14-16, Anaheim, California. Washington, D.C.: Thompson Books. 1967. p. 545-552.

3. Feingold, S.L. and Frye, C.H. User's Guide to PLANIT. Tech Memo TM-3055/000/01. Santa Monica: System Development Corporation. October 1966.

4. Frye, C.H. Teaching Statistical Inference by Computer: Problem Simulation for Hypothesis Testing and Evaluation (thesis for the degree of Ph.D.). Michigan State University: 1967.

5. Grubb, R.E. and Selfridge, L.D. "The Computer Tutoring of Statistics," Computers and Automation, 13(1964), p. 20-26.

6. Guilford, J.P. Fundamental Statistics in Psychology and Education. New York: McGraw-Hill Book Co. Fourth Edition 1965.

7. Hodges, J.L., Jr. and Lehmann, E.L. Basic Concepts of Probability and Statistics. San Francisco: Holden-Day, Inc. 1964.

8. Hoel, P.G. Elementary Statistics. New York: John Wiley and Sons. Second Edition 1966.

9. International Business Machines 1500 Operating System, Computer-Assisted Instruction - COURSEWRITER II Manual, FORM CAI-4036-1.

10. Karplus, W.J. On-Line Computing: Time-Shared Man-Computer Systems. New York: McGraw-Hill Book Co. 1967.

11. Kemeny, J.G. and Kurtz, T.E. BASIC Programming. New York: John Wiley and Sons, Inc. 1967.

12. Kennedy, P.R. The TINT User's Guide. Technical Memorandum TM-1933/000/03. Santa Monica: System Development Corporation. July 1965.

13. Parkhill, D. F. The Challenge of the Computer Utility. Reading, Mass.: Addison-Wesley Publishing Co. 1966.

REFERENCES (continued)

14. Perstein, M. H. Grammar and Lexicon for Basic JOVIAL. Technical Memorandum TM-555/005/00. Santa Monica: System Development Corporation. May 1966.

15. Schatzoff, M. "Console Oriented Model Building," ACM 20th National Conference Proceedings. New York: ACM Publication P-65. August 1965, P. 354-366.

16. Schwartz, J. I.; Coffman, E. G.; and Weissman, C. "A General-Purpose Time-Sharing System," AFIPS Conference Proceedings, Volume 25. SJCC 1964. P. 397-411.

17. Schwartz, J. I. and Weissman, C. The SDC Time-Sharing System Revisited. SP-2876. Santa Monica: System Development Corporation. August 1967.

18. Ziegler, J.R. Time-Sharing Data Processing Systems. Englewood Cliffs, N. J.: Prentice-Hall. 1967.

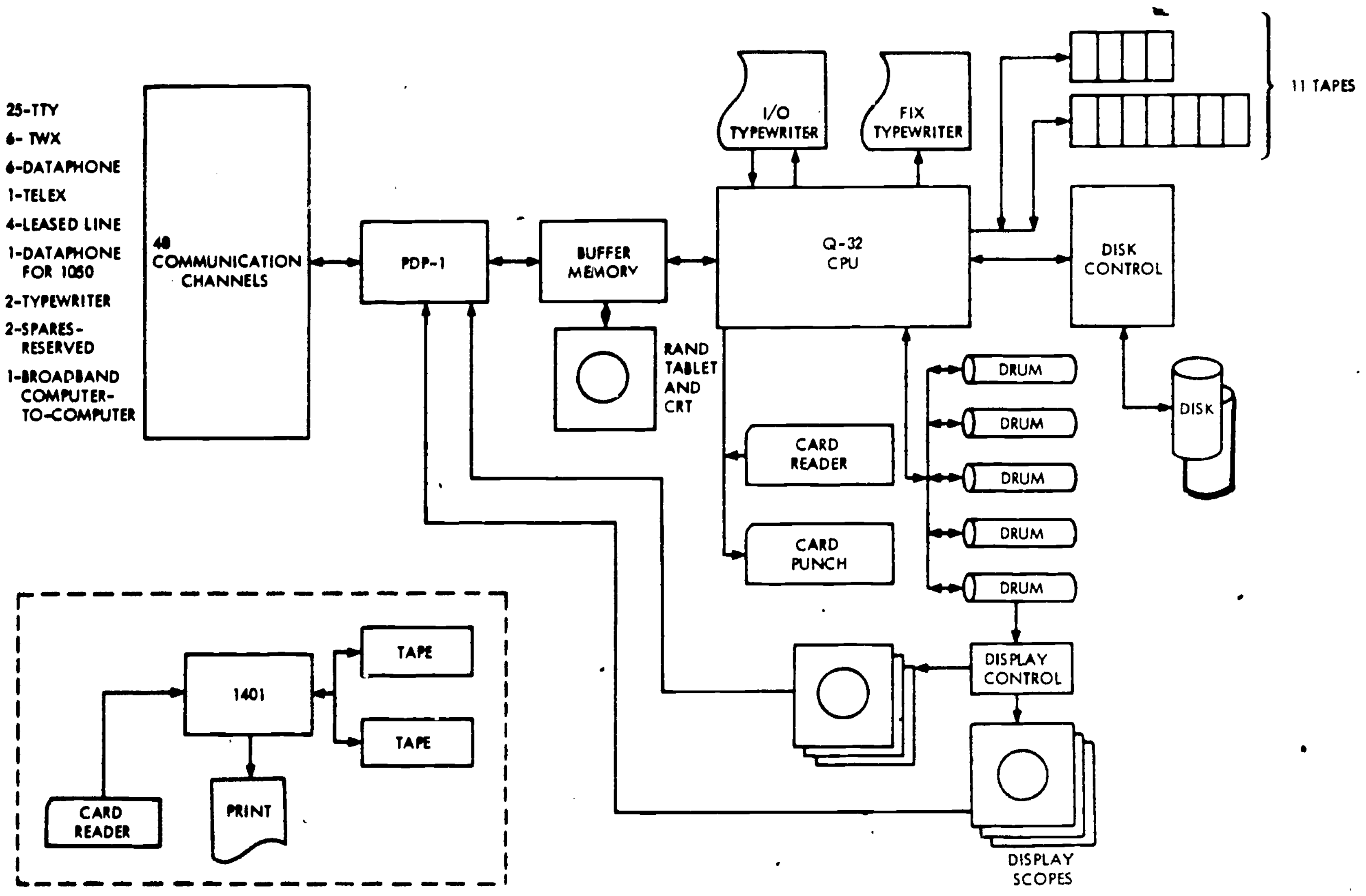
APPENDIX 1

THE SDC TIME-SHARING SYSTEM (TSS):
EQUIPMENT LIST AND CONFIGURATION

TSS EQUIPMENT

COMPONENT	NUMBER	CAPACITY/SPEED	TOTAL
AN/FSQ-32 COMPUTER			
Main Core Memory	4	16,384 words	65,536 words
. Cycle time 2.5 μsec.			
. 48-bit word			
Input Core Memory (Buffer)	1	16,384 words	16,384 words
. Cycle time 2.5 μsec.			
Drum	5	139,264 words	696,320 words
. Access time 11.5 ms.			
. Word transfer rate 2.75 μsec.			
Disc File	2	4,194,304 words	8,388,608 words
. Access time 225 ms.			
. Word transfer rate 12 μsec.			
Tape Drives (729-IV)	11	112 1/2 ips	
Card Reader (714)	1	250 cpm	
Card Punch	1	100 cpm	
Typewriter	2	100 wpm	
ASSOCIATED COMPUTERS (ON/OFF-LINE)			
PDP-1			
. Shares input core memory of Q-32			32K words
. Cycle time 5 μsec.			
. 18-bit word main core memory	1	4K words	4K words
1401-D			
. Core memory	1	4K char.	4K char.
. Printer	1	600 lpm	
. Tape drives (729-IV)	2	112 1/2 ips	
. Card reader (Uptime)	1	850 cpm	
I/O DEVICES			
Teletypes and Typewriters			
. Model 33 Teletypes	34	100 wpm	
. TWX data sets (remote users)	6	100 wpm	
. Soroban typewriters	3	90 wpm	
. Telex data sets	1	100 wpm	
. Data-Phone sets	6	100 wpm	
. IBM 1051-1052	1	150 wpm	
. Communications testboard	1		
Display Consoles	6	2K char. max. (per console)	
. Light pens			
. Vector-generator capability			
. Graphic tablet	1	5K points/sec.	
. Keyboard	2	100 wpm	

TSS CONFIGURATION



APPENDIX 2

STAT OPTIONS, COMPUTATION CONVENTIONS, AND EXAMPLE OF STUDENT'S WORK

Student's Reference Sheet for STAT-OPTIONS

OPTION=?

DRAW SAMPLE	1.	SAMPLE data	
	2.	SUMS	
	3.	Sums of SQUARES	
	4.	Sum of PRODUCTS	
CALCULATION	5.	Sum of DIFFERENCES	
	6.	Sum of SQUARED DIFFERENCES	
	7.	Sums of N^{th} POWERS	
	12.	CALCULATION ASSISTANCE program Legal symbols: Sj SSj SCP SMD SSD RX Fn mCn Lj ** * + - / ( ) []	
	9.	ANSWER to most recently MISSED QUESTION (Use as last resort)	
HELP	10.	HELP, OPTIONS 21 to 66	
	11.	STEPS in SOLUTION	
YOUR ANSWERS	8.	Student RESPONSE and EVALUATION	
BEGIN LIBRARY	21.	MEAN	P.161*
	22.	BIASED VARIANCE and standard deviation	P. 177-8
	23.	UNBIASED VARIANCE and standard deviation	P. 207
	24.	STANDARD ERROR of the MEAN	P. 202
ESTIMATES	25.	POOLED STAND. ERROR, for differ. of sample means	P. 320
	26.	MEAN of DIFFERENCE scores	P. 335
	27.	BIASED VARIANCE of DIFFERENCES	P. 335
	28.	UNBIASED VARIANCE of DIFFERENCES	P. 335
	31.	CONFIDENCE LIMITS for MEAN	P. 312
SAMPLE STATISTICS BASED ON THE T-DISTRIBUTION	32.	CONFIDENCE LIMITS for DIFFERENCE of MEANS	P. 321
	33.	CONF. LIM. for DIFF. of MEANS, CORRELATED case	P. 334-5
	34.	Criterion for testing that RHO (correlation coefficient) equals zero	P. 533
	35.	T-statistic for HYPOTH. TEST of MEAN	P. 311
	36.	T-statistic for HYPOTH. TEST OF DIFFER. of MEANS	P. 317
	37.	T-statistic, TEST of DIFF. of MEANS, CORRELATED case	P. 334-5

NOTE: Always wait for =? before typing a message.

*These are page references to the textbook, Statistics for Psychologists by William L. Hays, Holt, Rinehart and Winston, New York, 1963.

OPTION=? (continued)

	42.	MANN-WHITNEY TWO GROUP SIGNIF. probability	P. 633
	43.	WILCOXON PAIRED OBSERVATION SIGNIF. probability	P. 635
NON-PARAMETRIC	44.	SPEARMAN RANK CORRELATION coefficient	P. 643
	45.	FISHER EXACT TEST	P. 598
STATISTICS BASED ON CHI-SQUARE DISTRIBUTION	51.	GOODNESS-OF-FIT test statistic	P. 586
	52.	CONFIDENCE LIMITS for VARIANCE	P. 345
	53.	CHI-SQUARE test of INDEPENDENCE	P. 589
	54.	CONTINGENCY coefficient	P. 606
	55.	PRODUCT MOMENT CORRELATION coefficient	P. 505
DEPENDENCE MEASURES	56.	Criterion for testing that RHO (correlation coefficient) equals zero (same as OPTION 34)	P. 533
	57.	T-statistic for CORRELATION TEST	P. 529
	58.	PHI-COEFFICIENT	P. 604
	59.	SPEARMAN RANK CORRELATION	P. 643
	60.	T-statistic for RANK CORRELATION test (approx.)	P. 646
	61.	X on Y Slope, INTERCEPT and S.E. of prediction	P. 504-5
	62.	Y on X Slope, INTERCEPT and S.E. of prediction	P. 504-5
REGRESSION	63.	X on Y S.E. of BETA and T-value for BETA	P. 521
	64.	Y on X S.E. of BETA and T-value for BETA	P. 521
	65.	X on Y Prediction and CONFIDENCE INTERVAL	P.504-5, P.522
<u>LIBRARY ENDS</u>	66.	Y on X Prediction and CONFIDENCE INTERVAL	P.504-5, P.522
	13.	BINOMIAL, $\sum_{S} \binom{N}{S} P^{S}(1-P)^{N-S}$	
PROBABILITY DISTRIBUTIONS	14.	HYPERGEOMETRIC, $\sum_{D} \binom{R}{D}\binom{N-R}{S-D} \Big/ \binom{N}{S}$	
	15.	MANN-WHITNEY	level of significance for specified minimum rank sum and total number of ranks (i.e., sample size)
	16.	WILCOXON paired observation	
<u>TO TERMINATE</u>	20.	LESSON TERMINATION, records and continuation code	

NOTE: Strike CARRIAGE RETURN button to enter typed messages into the computer. To cancel a complete message, strike " (upper case 2) before striking the CARRIAGE RETURN. Strike the RUBOUT key to eliminate the last character you have typed; continuing to strike the RUBOUT key to eliminate the preceding characters, one at a time.

Conventions for OPTION 12, CALCULATION ASSISTANCE

The following symbols may be employed in OPTION 12 to compose expressions or statements for numerical evaluation. The expressions must appear on a single line and are typed following the STAT message ST=?. Where X and Y appear below the student would type explicit decimal numbers or expressions composed of symbols and decimal numbers; m and n are used in place of X and Y where symbols can only operate on integers or on expressions which take on integer values.

SYMBOL	ST=?	INTERPRETATION
+	X+Y	add Y and X
-	X-Y	subtract Y from X
	-Y	negative of (i.e. opposite of) Y
/	X/Y	X divided by Y
*	X*Y	X times Y
**	X**Y	X raised to power Y
(,)	(---)*X	the expression within parentheses times X
[,]	[---]*X	the expression within brackets times X
R	RX	square root of X
F	Fn	n! i.e. n factorial; n must be no greater than 170 (e.g., F4=24=4·3·2·1)
C	mCn	the number of combinations that can be obtained from m things taken n at a time; mCn=Fm/[Fn*F(m-n)]
L	Lm	the value <u>previously</u> computed by STAT in line m; Lm may be used in the same manner as X, Y, m, n, are in ST=?

Your most frequent use of OPTION 12 will be for calculations on sample data obtained in OPTION 1. Such calculations are more conveniently done with the following symbols which may be typed by the student, in the expressions submitted in ST=?, as if they were explicit numbers.

These symbols may replace explicit numbers in OPTION 12:

SYMBOL	DEFINITION
S1	sum of data in column 1
S2	sum of data in column 2
SS1	sum of squares of data in column 1
SS2	sum of squares of data in column 2
SCP	sum of products of data in the same row of columns 1 and 2
SMD	sum of differences of data in the same row of columns 1 and 2
SSD	sum of squared differences

An additional = sign may be typed by the student (in addition to that printed by the computer) to indicate where evaluation is to begin. This allows comments or labels to be inserted for reference only, e.g.,

```
LINE 1.
ST=? MEAN=S1/1Ø        An additional equal sign follows
                       the label MEAN.

L1=     43.ØØØØ
```

Try END as your last reply to ST=? to exit from OPTION 12.

If you reenter OPTION 12 during the same problem, line numbers continue from where you left off.

EXCERPT FROM STUDENT'S WORK WITH STAT

(Note: References which are printed by STAT as in the case of OPTION 11 below, are for the textbook by Hays, Statistics for Psychologists. The underlined messages were typed by the student.)

```
PROBLEM  1.07

   IN A FIELD TEST OF A NEW SERUM A LARGE SAMPLE
WAS DIVIDED INTO TWO TREATMENTS.  TREATMENT 1 RECE-
IVED THE SERUM, TREATMENT 2 DID NOT.  THE DATA
RECORDS THE NUMBER OF CASES IN EACH CATEGORY FOR THE
ENTIRE SAMPLE.
   TEST THE PROPOSITION THAT THE SERUM WAS EFFECTIVE.
   USE THE 0.05 LEVEL OF SIGNIFICANCE.

 OPTION=? 1

THE SAMPLE VALUES FOLLOW:
TREATMENT--HEALTHY--INFECTED
    1-     219        3
    2-     186        1

 OPTION=? 11

SOLVE FOR EACH QUANTITY SPECIFIED BY THE STEPS.
STEP 1
 FISHER EXACT TEST: P598, S17.9

 OPTION=? 12

CALCULATION ASSISTANCE.

LINE 1 .

ST=? 409C222
L1 =       1.16814991E+121

ST=? [(405C219)*(4C3)]/L1
L2 =                0.2928

ST=? [(405C218)*(4C4)]/L1
L3 =                0.0857

ST=? L2+L3
L4 =                0.3785

ST=? END
```

(Continued on Page 71.)

(Continued from Page 70.)

```
 OPTION=? 8

EVALUATION
ANSWER QUESTIONS AS DIRECTED OR TYPE 'BACK' OR 'STUCK.'
ENTER THE SIGNIFICANCE LEVEL OBTAINED FROM YOUR TEST.
(IF YOU USED A TABLE, GIVE NEAREST APPROXIMATION.)
       P=? .3785
PROBABILITY INCORRECT.
CORRECT BEFORE PROCEEDING.
 FISHER EXACT TEST: P598, S17.9

 OPTION=? 45

FISHER EXACT PROBABILITY

ENTER THE NUMBER OF 'SPECIALS' OCCURING IN THE EXPERIMENTAL GROUP.

     NUM=? 3
ENTER THE TOTAL NUMBER OF 'SPECIALS.'
     NUM=? 4
ENTER THE EXPERIMENTAL GROUP TOTAL.
     NUM=? 222
  PROBABILITY =             Ø.9143

 OPTION=? 9

-ANSWER- **** ~             Ø.9143

 OPTION=? 8

EVALUATION
ANSWER QUESTIONS AS DIRECTED OR TYPE 'BACK' OR 'STUCK.'
ENTER THE SIGNIFICANCE LEVEL OBTAINED FROM YOUR TEST.
(IF YOU USED A TABLE, GIVE NEAREST APPROXIMATION.)
      P=? .9143
NOW YOU ARE CORRECT.
```

Note: Because the student used LIBRARY OPTION 45 he is forced to work the problem again, with new data; this time the LIBRARY OPTIONS are not available to him.

(Continued on Page 72.)

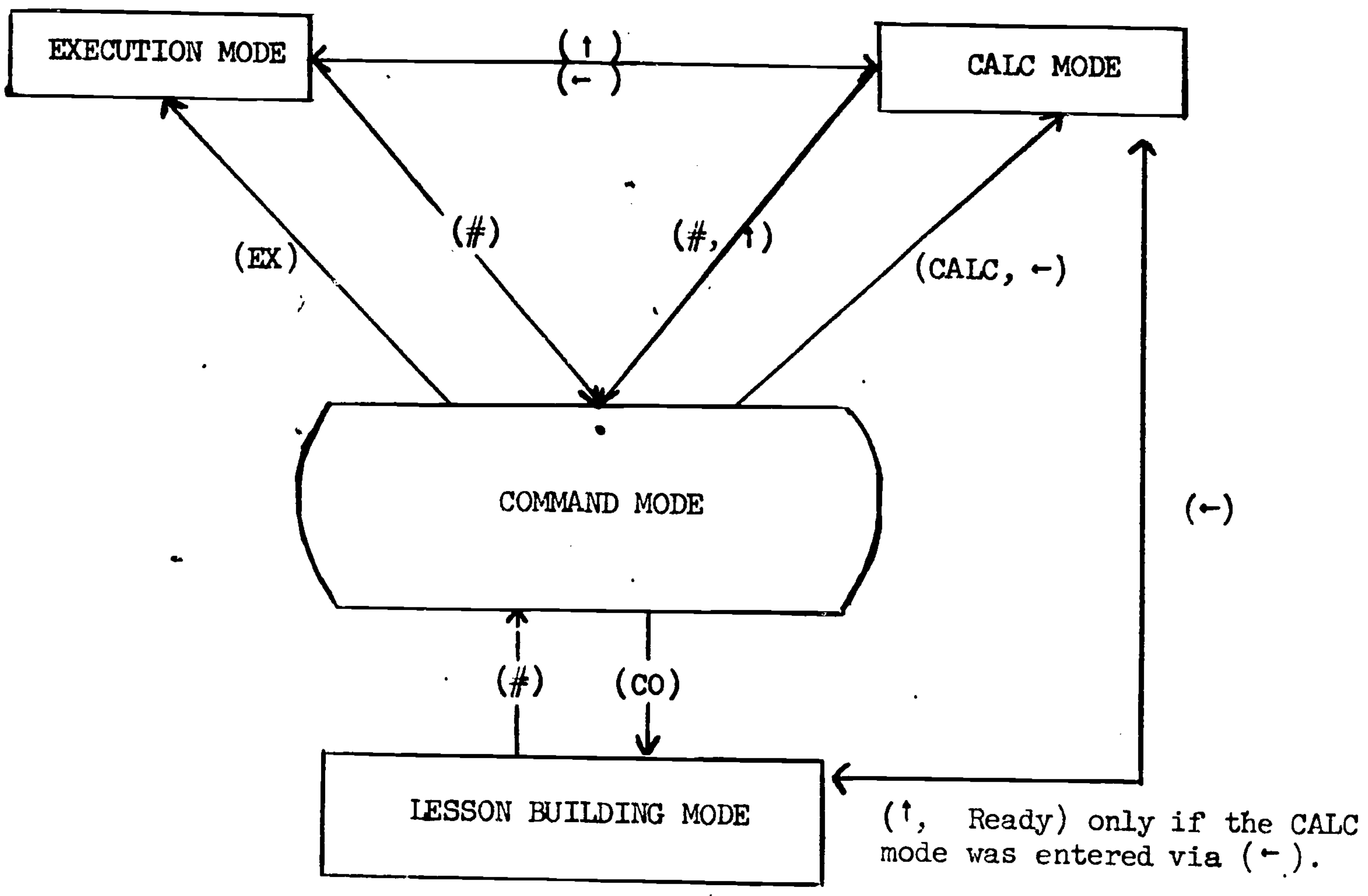
(Contined from Page 71.)

```
PROBLEM  2.07

   IN A FIELD TEST OF A NEW SERUM A LARGE SAMPLE
WAS DIVIDED INTO TWO TREATMENTS.  TREATMENT 1 RECE-
IVED THE SERUM, TREATMENT 2 DID NOT.  THE DATA
RECORDS THE NUMBER OF CASES IN EACH CATEGORY FOR THE
ENTIRE SAMPLE.
   TEST THE PROPOSITION THAT THE SERUM WAS EFFECTIVE.
   USE THE 0.05 LEVEL OF SIGNIFICANCE.

 OPTION=? 1

THE SAMPLE VALUES FOLLOW:
TREATMENT--HEALTHY--INFECTED
    1-     195        4
    2-     199        6

 OPTION=? 12

CALCULATION ASSISTANCE.

LINE 1 .
ST=? [(394C195)*(10C4)]/404C199
L1  =                   0.2127

ST=? [(394C196)*(10C3)]/404C199
L2  =                   0.1234

ST=? [(394C197)*(10C2)]/404C199
L3  =                   0.0465

ST=? [(394C198)*(10C1)]/404C199
L4  =                   0.0103

ST=? [(394C199)*(10C0)]/404C199
L5  =                   0.0010

ST=? L1+L2+L3+L4+L5
L6  =                   0.3940
ST=? END

 OPTION=? 8

EVALUATION
ANSWER QUESTIONS AS DIRECTED OR TYPE 'BACK' OR 'STUCK.'
ENTER THE SIGNIFICANCE LEVEL OBTAINED FROM YOUR TEST.
(IF YOU USED A TABLE, GIVE NEAREST APPROXIMATION.)
        P=? .394
GOOD WORK.  NO ERRORS.
```

APPENDIX 3

A BRIEF USER'S GUIDE TO PLANIT

TABLE OF CONTENTS



#	The hash mark is used to enter the COMMAND mode (PLANIT prints UNRECOGNIZABLE CHARACTER if the student attempts to use it).
CO	The CO command is used to enter the Lesson Building mode to COmmence or COntinue building lessons.
EX	The EX command is used to enter the EXecution mode.
↑	The up arrow is used to return to the mode from which the CALC mode was entered.
←	The left arrow is used by the student, or the LD playing student, to enter the CALC mode.
CALC	The CALC command is used by the LD to enter the CALC mode.

Figure 1. Mode Selection Commands

(Q) The Question Frame

The question frame is extremely flexible and would probably be used mostly for general interaction with the student.

This frame can be used in two basic ways: (1) Constructed response; and (2) Dichotomous answer (POS/NEG).

A new user should look at the constructed response type first. This version of the Q-frame may be the only one a LD needs for his lessons.

Constructed Response

(Remember, all information not prefaced by an asterisk is typed out by PLANIT. All information prefaced by the asterisk is typed in by the LD.)

Simple use of the Q-frame. The LD types in Q, PLANIT responds with:

```
FRAME 1.ØØ LABEL=*HIST

2.  TEXT.
*?
2.  SPECIFY QUESTION.
*WHO INVENTED THE ELECTRIC LIGHT?
*

3.  ANSWERS.
*A+THOMAS EDISON
*B ALEXANDER BELL
*

4.  ACTIONS.
*A F:THATS VERY GOOD B:3
*B R:HE INVENTED THE TELEPHONE, TRY AGAIN...
*-R:
*
```

Explanation of Frame 1

All frames are automatically numbered. Here the LD chooses to label the frame HIST. (If he chose not to label the frame, he would merely strike the space bar followed by the carriage return and pass on to Group 2.)

2. TEST. The LD was not sure what TEXT stood for and typed in the question mark (?). PLANIT immediately elaborates with 2. SPECIFY QUESTION. The LD then types in his question "Who invented the electric light?" PLANIT returns with an asterisk waiting for more lines of input. PLANIT has no way of knowing when the LD is through with the question group and so returns with an asterisk for each

next line. The LD ends the group by striking the space bar once and then the carriage return key. There is no theoretical limit to the number of lines that can be entered in each group. We do, however, have a practical limit of 60 lines for the entire frame.

3. ANSWERS. After the LD enters the space and carriage return, PLANIT returns with 3. ANSWERS. A stroke of the ? would have produced the elaboration 3. SPECIFY ANSWER. The LD now enters all the anticipated answers he will accept. In doing so, he tags the first one A and the second one B. Note the + next to the tag A, this indicates to PLANIT that this is the correct answer. Having finished entering answers, the LD indicates this by a space and carriage return (CR) and we go on to the action group.

4. ACTIONS. Stands for SPECIFY ACTIONS; the LD could have discovered this with a stroke of the ? key. In this group we specify that we want to do next depending upon which answer the student gives. Action specifications for any given tagged answer must be confined to one line.

There are four types of commands in the action group: F:, R:, C:, B:.

F: means that what follows is the feedback message to be presented to the student. If no message follows F:, then PLANIT will choose one from its feedback table. PLANIT has a list of correct and incorrect feedback message and will select one from the appropriate list (randomly) according to whether the answer associated with this action was correct or incorrect. This allows the LD to enter F: by itself, knowing the student will not always get "YES" or "NO" feedback every time he enters an answer. The response will normally be one-word messages, e.g., YES, FINE, CORRECT, SO, or NOT TRUE, NO, WRONG, etc.

R: operates the same as F:, except that R: also instructs PLANIT to wait for another answer (i.e., do not print out the question, just wait for another answer). Finally, R: by itself means print out the fixed message, "WRONG, TRY AGAIN" (then wait for another answer).

C: by itself means print out the fixed message "THE CORRECT ANSWER IS" followed by the correct answer as indicated by the plus (+) in Group 3. C: can only be followed by another command, e.g., F:, R:, C:, B: or a CALC statement. For example C: CONT=CONT+1. This puts PLANIT in the CALC mode at that point. CONT is an item in CALC; here the LD would be incrementing CONT by one (see CALC 2.3.1 and 2.4).

B: means branch (e.g., B:3 means branch to Frame 3). All frames are numbered. They can also be labeled, and a branch can be made to any numbered or labeled frame. In addition, "B:LSNAM" where LSNAM is the name of another lesson, means that the lesson LSNAM will now be brought into PLANIT and executed (just like another frame; the student will never know the difference). Upon completion of the lesson LSNAM, PLANIT will continue with the next frame in the original lesson.

B: by itself with no label or number following it means return to the calling lesson (i.e., if lesson AA called on lesson BC "B:BC", then PLANIT would return from lesson BC when PLANIT comes across "B:"). If CNT is some item and contains a number (put there in CALC perhaps), then B:CNT means branch to the frame number contained in CNT; e.g., if LINK contained the value 20, then B:LINK means branch to Frame 20 and execute as per the instructions in Frame 20. Frames mentioned in a branch need not exist unless that branch is executed during the running of the lesson. If it is executed and no frame exists, then PLANIT will interrupt the lesson and print out an error message stating the illegal branch, the frame, group, and line number where it was requested.

The first line in Group 4 is read as: If the student gave Answer A (i.e., THOMAS EDISON), then print out the message: THATS VERY GOOD and then branch on to Frame 3.

The second line is interpreted as: If the student gave Answer B, then print out: HE INVENTED THE TELEPHONE, TRY AGAIN... then wait for another answer.

The third line indicates an action to be performed if the student's answer did not correspond with either A or B; i.e., for any unanticipated answer (actions prefaced by the minus sign), repeat the frame. (In this case note that nothing appears after the R:. This means print out the fixed message: WRONG, TRY AGAIN and wait for another answer.)

The LD terminated the frame by striking the space bar and the carriage return key (CR).

Remember, the space bar and CR alone on any line terminates the group. The dollar sign "$" and CR alone on any line terminates the frame. If, for example, the LD (in the middle of Group 3) decided to end the frame and go onto the next frame, he would only have to enter the $ and CR alone on a line and PLANIT would respond with:

```
P/Q/M/D/C.
*Q

FRAME 2.ØØ LABEL=*MATH

2. TEXT.
*LETS SEE WHAT YOU REMEMBER ABOUT TEMPERATURE.  USING F FOR DEGREES
*FAHRENHEIT AND C FOR DEGREES CENTIGRADE, WRITE THE FORMULA FOR
*CONVERTING FROM DEGREES FAHRENHEIT TO DEGREES CENTIGRADE.\
* HINT:  F=9*C/5+32  CONVERTS FROM CENTIGRADE TO FAHRENHEIT.
*

3. ANSWERS.
*Ø FORMULAS ON
*A+C=(5/9)*(F-32)
*B F=9*C/5+32
*C C=(5/9)*F-32
*

4. ACTIONS.
*A F:   B:7
*B R:YOUR ANSWER IS THE SAME AS THE ONE I GAVE YOU, TRY AGAIN...
*A F: NOW YOU'VE GOT IT.  B:15
*B R:YOU'RE STILL CONVERTING FROM CENTIGRADE TO FAHRENHEIT, TRY AGAIN...
*BC C:  F:NOTE THE DIFFERENCE.  B:OUT
*-R:
*-C:
*
```

Explanation of Frame 2:

The LD labels this frame MATH.
2. TEXT. The LD enters his question. Notice the back slash "\" after CENTIGRADE on the 3rd line. This instructs PLANIT to skip a line after printing out centigrade. To a student taking the lesson, the question would come out as:

```
LETS SEE . . . . . . . . . . . . . . . . . DEGREES
FAHRENHEIT . . . . . . . . . . . . . . . . FOR
CONVERTING . . . . . . . . . . . . . . . . CENTIGRADES

HINT:  F=9*C/5+32. . . . . . . . . . . . . FAHRENHEIT
```

* (Where the asterisk tells the student that it's ready for his answer.) The "\" can be used anywhere in Group 2. Two in a row "\\" means skip two lines, etc.

3. ANSWERS. This group illustrates the algebraic matching ability of PLANIT (turned on by the expression: Ø FORMULAS ON). The student can type in any equivalent algebraic form of the correct answer and get full credit for it; e.g., C=(F-32)*(5/9), C=5*(F-32)/9, C=(5*F-160)/9, etc., are all equivalent and therefore acceptable forms. If FORMULAS is not turned on, or is turned off by the expression: Ø FORMULAS OFF, then only the exact form as typed in by the LD would be looked for in the matching.
We do not wish to mislead the user; this is not true symbol manipulation. We merely employ a technique which (in part) includes performing algebra on the student's answer. For details on the technique see Note 1.

Note: (5/9)*(F-32)=C is not considered an equivalent form to PLANIT. Naturally the matching technique is not restricted to correct answer alone but works for all anticipated answers in Group 3.

4. ACTIONS. This group illustrates repeated use of a frame.
The <u>FIRST</u> time through this frame, if the student's answer corresponded to:

A--he would receive a (randomly selected) feedback message followed by a branch to Frame 7.

B--he would receive the feedback message "YOUR ANSWER IS THE SAME AS THE ONE I GAVE YOU, TRY AGAIN...".
(Remember the R: means wait for another answer.)

C--he would receive: NOTE THE DIFFERENCE. THE CORRECT ANSWER IS: C=(5/9)*(F-32) followed by a branch to the frame whose label is OUT.

If his answer did not correspond to either A, B, or C (unanticipated response) he would have received: WRONG, TRY AGAIN.

The <u>SECOND</u> time through the frame, if the student's answer corresponded to:

A--he receives: NOW YOU'VE GOT IT. following by a branch to Frame 15. This action is performed regardless whether or not his first answer corresponded to A. In other words, the n<u>th</u> repetition of a lettered (or numbered) action, counting from the top of Group 4, is performed the n<u>th</u> time through the frame (provided, of course, that his answer corresponds to that lettered or numbered action). Again the action chosen is dependent only upon the student's answer and the number of times he has been through the frame and <u>not</u> upon what answer he gave before in this frame.

B--he receives: YOU'RE STILL CONVERTING FROM CENTIGRADE TO FAHRENHEIT. TRY AGAIN....

C--he receives the same as he would have gotten the first time through the frame had his answer corresponded to C then.

If no match (2nd unanticipated answer) he receives:
THE CORRECT ANSWER IS: C=(5/9)*(F-32). PLANIT would then go on to the next frame in the sequence.

The THIRD (or more) time through the frame, if the student's answer corresponded to:

A, same for A the second time through.

B, same for C any time through the frame.

C, same as before.

Unanticipated answer, same as second time through.

Note: F: C: R: B: appearing on one line are done in the order shown. If the same command appears more than once on the same line, the priority is from left to right. If "B:" appears more than once on any line, the left-most branch will always be executed.

If there is no Group 3, then all commands in Group 4 will be executed, and there will be no pause for the student's answer.

Specifications for FORMULAS, KEYWORD, and PHONETIC*

A. What they do:
FORMULAS OFF implies that the student's answer is to be regarded as a standard English reply (i.e., word, number, phrase or sentence).

FORMULAS ON implies that the student's answer is to be regarded as an algebraic formula where commutative, associative, and distributive rules hold. Names are interpreted as parameters of the formula to which arbitrary numbers could be assigend.

*For details on techniques used in FORMULAS and PHONETIC, see Note 1 and Note 2 respectively.

KEYWORD ON provides a means for disregarding everything in the student's answer except that which occurs in the LD answer to be matched. The minimal acceptable reply is that which matches the LD's answer. This reply may be a part of a longer reply.

KEYWORD OFF implies an exact match word-for-word.

PHONETIC ON implies that all words in both the LD's answer and the student's reply will be reduced to a phonetic encoding and the encoded messages will be compared - not the original messages.

PHONETIC OFF disallows that function.

B. How to use them:

Each of the three is controlled in exactly the same way. They can be turned on or off in CALC or in any statement on which CALC operates. Once turned on, they will be automatically turned off again when advancing from one frame to the next <u>unless</u> the SET operator is included. If they are SET on, they will remain on until that status is changed by a subsequent declaration of the status of that function.

The chart below shows several uses:

	In CALC	In Answer Group	In Action Groups
Temporarily ON	KEYWORD ON	∅ KEYWORD ON or 1 KEYWORD ON	C: KEYWORD ON
Set ON	SET KEYWORD ON	∅ SET KEYWORD ON or 1 SET KEYWORD ON	C: SET KEYWORD ON
OFF	KEYWORD OFF	∅ KEYWORD OFF or 1 KEYWORD OFF	C: KEYWORD OFF

Note: In the answer group, the ∅ implies that the action will be taken before the student responds, while 1 or greater implies that the action will be taken at the position found in the process of answer-matching.

C. How they work:

Their function will be noticed only in question-type frames for answers with letter tags. The following chart shows the various combinations.

	FORMULAS ON	FORMULAS OFF
KEYWORD* ON	Answers will be treated as algebraic formulas. Order is unimportant to match, at least the keywords must be found.	Words appearing in the answer must appear somewhere, properly ordered, in the student's reply.
KEYWORD OFF	Answers will be treated as algebraic formulas. Order is unimportant. Every word in the answer must occur in the reply and no others.	Everything in the answer must appear in the same way in the student's reply. Only leading and trailing blanks are ignored.

In each case, PHONETIC ON simply phonetically encodes the words.

D. Operational definitions:

A word is a string of letters with a blank before and after it, i.e., EDISON.

With FORMULAS OFF, any string of characters set apart by blanks are handled as unit "words."

With FORMULAS ON, the word is a string of letters that may or may not have digits attached on the right, i.e., MEAN SUM1 RHO12

E. Restrictions and special conditions:

1. With PHONETIC ON, only the first sixteen letters of each word will be encoded.
2. With FORMULAS ON and PHONETIC OFF, all words must be distinguishable within the first eight characters; i.e., FREQUENCY would match with FREQUENCIES. No such restriction holds with FORMULAS OFF.
3. With FORMULAS ON, punctuation is ignored and arithmetic symbols retain their function.
4. With FORMULAS OFF, no symbol has special meaning and must be matched exactly. Punctuation is not ignored unless KEYWORD is on.
5. FORMULAS ON is often useful for other than formula replies, e.g., THOMAS EDISON would allow EDISON, THOMAS to match correctly (the comma is ignored and order is not observed).

*Periods, commas, question marks, and extra blanks are ignored with KEYWORD on.

6. All answers are restricted to a single line.

F. Other frames:

1. Multiple choice frames always require a single letter in reply. No other reply is accepted. That letter must match one of the tags in the answer set or else another answer is requested by the message: "CHOOSE ONE OF THE ABOVE LETTERS."

2. Answers tagged with numbers in a question frame imply FORMULAS ON, all else OFF, and no letter will be acceptable in the reply in order for a match to occur. Thus, 1 will match 1.0 or 3/3 or 3*1 but not X/X. If all answer tags in any question frame are numbers, then PLANIT will not accept a reply to that question if it contains letters. Another answer will be repeated with the message, "NUMERICAL ANSWER PLEASE."

(D) The Decision Frame

This is an extremely powerful frame and can be used for determining just about anything the student has done. In this frame the LD specifies branching (or any of the three F: C: B:) conditions on a basis of a pattern of errors over several frames. He can also query the contents of items set in CALC. Finally, he can combine his decisions to be a function of what the student has done over a set of frames plus the contents of items set in CALC.

Decision frame questions are composed of the following primitives:

1. The Connectives: IF, AND, OR.
2. The Relational Operators: LS - less than; LQ - less than or equal to; EQ - equal to; GQ - greater than or equal to; GR - greater than; and NQ - not equal to.
3. The Conditions: RIGHT, WRONG, SEEN, USED, MINUTES, +, -.
4. Frame numbers and labels.
5. Lettered or Numbered answers.
6. CALC expressions.

The beginning of any decision question must start off with the "IF" connective. The question can run over onto more than one line, but all lines must start off with a connective.

The questions can have any one of three forms:

A. IF Frame number (or label), Lettered (or numbered) answer -+.
E.g., IF 5,AB 7-12,+ 3,1P 6,+-
This is interpreted as follows: If the student went through Frame 5, Answer A or B and then went through Frames 7 to 12 (inclusive) correctly and then Frame 3, Answer 1 or P and then saw Frame 6. The first form is the "Pattern" form for it describes an exact pattern. In order for this question to be satisfied, the student must go through the frames mentioned in the exact order specified with no deviations to frames in between nor to the answers given.

B. IF Relational number RIGHT (WRONG, SEEN, USED, MINUTES) Frame numbers (labels).
E.g., IF GQ 5 WRONG 1,7,LAB-25
If the student got at least five wrong out of Frames 1, 7, and all frames between the frame labeled "LAB" and Frame 25.

E.g., IF RIGHT 3-5. If he got Frames 3 to 5 right.
E.g., IF LG 5 WRONG. If he got five frames wrong.
In this form any part is optional and can be left out except RIGHT, WRONG, or SEEN.

E.g., IF LQ 20 MINUTES 1-5, 7 if the student went through Frames 1-5 and 7 in at most 20 minutes. This includes all repetitions of frames.

E.g., IF USED FACT 1, 2 and USED MEAN 3.
If he used the primitive CALC function "FACT" in either of Frames 1 or 2 and used the LD defined function "MEAN" in Frame 3.

C. IF CALC Expression Relational CALC Expression
E.g., IF CNT GQ 5. If the contents of CNT is greater than or equal to five.

E.g., IF FACT(CNT)-SQRT(STUDIQ(5,CNT))*TAN(50)**2 LS COMB(5.CNT).
If the factorial of the contents of CNT minus the square root of the element of the matrix STUDIQ whose subscripts are 5 and the contents of CNT, times the tangent squared of 50 is less than the combination of 5 things taken CNT at a time.

All of the three forms, A, B, and C can be connected by any of the connectives AND, or OR.

E.g., IF 3,AB LAB,+ OR 2 SEEN LOGIC AND IQ LQ 120
OR GQ 3 RIGHT HIST-MTH.

If the student went through Frame 3, Answer A (or B) and then was correct on LAB OR he has seen the frame labeled LOGIC twice and the contents of the item, IQ, is less than or equal to 120 OR he got at least three right out of all the frame between HIST and MTH (inclusive).

The terms of these expressions are separated by OR's. Although no parentheses are permitted, they will be used here to clarify the order in which the example just given would be evaluated.

```
IF (3,AB LAB,+), OR (2 SEEN LOGIC AND IQ LQ 120)
OR (GQ 3 RIGHT HIST-MTH).
```
The terms are always evaluated left to right.

```
P/Q/M/D/C
*D

FRAME 9.ØØ LABEL=*DET

2.  CRITERIA.
* IF 3,-     F: LET'S TRY SOME EXAMPLES   B:EXAM
* IF 5-10,SK  EXAM,+ OR LQ 2 SEEN 20,23-25   B:72
* IF 2 RIGHT HIST-MTH AND STUDIQ LQ 130 OR 7-10,ABCD56
* AND ALL SEEN 5-35   C:CNT=10   C:LINK(2)=25   B:105
* IF NONE SEEN 5-50 OR 5,A 6,+
* F:FINE NOW YOU'VE GOT IT  C:  TEMX=FACT(40)  B:67
```

Explanation of Frame 9:

The first line reads: If he got Frame 3 wrong, printout the feedback message "LETS TRY SOME EXAMPLES" then branch to frame whose label is EXAM.

The second line reads: If he went through Frames 5 to 10, answers S or K, and then went through the frame whose label is EXAM correctly OR he has seen at most two out of Frames 20 and 23 to 25, then branch to Frame 72.

The third line will be easier to follow if we label its terms.

Let: 2 RIGHT HIST-MTH → A; STUDIQ LQ 130 → B; 7-10,ABCD56 → C;
ALL SEEN 5-35 → D
Then line three is read as: IF (A AND B) OR (C AND D).
Then set CNT equal to 10, LINK(2) equal 25 and branch to Frame 105.

Finally, those last two lines introduce the use of ALL and NONE. In the first one we have: If the student has seen ALL the Frames 5-35. In the second we have: If he hasn't seen any of the Frames 5-50.

As shown in Group 4 of the Q-frame, the action commands can be grouped with a particular answer. Similarly in the Decision frame, action commands F, C and B (R not used) can be grouped following an IF statement. Remember, to continue a feedback message onto more than one line, the lines must start with "F:". Commands not prefaced occurring at the top of the frame are done first.

```
E.g.,
FRAME X LABEL=X
2. CRITERIA.
*F:THIS IS DONE FIRST, THE REST OF THE GROUP
*F:IS THEN PROCESSED.
*IF 1,A...etc.
```

Search Boundaries:

All examples shown thus far imply restrictions to the scope of the search. If this is the first time this frame is used (i.e., this frame number appears nowhere else in the student's record), the search begins from the first frame as before. If, however, this decision frame was used before, then the search starts off from the frame record associated with the last time this decision frame was used.

The LD can override the limitation by specifying the frame number from which the search will begin. He can use the primitive "FROM" followed by the frame number or label. The primitive "FROM" is used just after the "IF". e.g., IF FROM 20 GQ 5 RIGHT...etc., or IF FROM EXAM GQ 5 RIGHT...or IF FROM 5 6,A 7,BCD...the search starts off from the last occurrence of that frame record whose number is the same as that which follows the primitive "FROM" in the decision statement.

If the frame number is not found, a search is made of the entire record to see if the pattern is satisfied anywhere.

Remember, the pattern must exist exactly as specified in the decision question (with no deviations). The search starts from the beginning of the lesson and ends when the question is satisfied or until the last record is examined, whichever comes first.

Finally, for RIGHT and WRONG, D (decision) and P (problem) type frames are ignored.

Consider the following examples:

	FRAME	ANSWER	RIGHT/WRONG	FRAME TYPE
1.	1	A	+	Q
2.	2	K	-	Q
3.	5	A	-	M
4.	7	C	+	Q
5.	11	C	+	Q
6.	4	P	-	M
7.	5	B	+	M
8.	7	C	+	Q
9.	7	C	+	Q
10.	9	C	+	Q
11.	10	C	+	Q
12.	20	O	-	D
13.	15	S	+	M
14.	35	O	-	P
15.	12	O	+	D
16.	20	O	-	D

Figure 2

A) IF FROM 1 5, AB 7-10, C
B) IF 4 SEEN 5, 12-35
C) IF 3 RIGHT 5, 12-35

Referring to Figure 2, Question A would be satisfied. Note that the pattern is disturbed in Lines 3-5, but the search continues and a complete match is found in Lines 7-11*. By removing the "FROM", Question A would not be satisfied, since the specified pattern occurred prior to the last use of Frame 20.

Question B would not be satisfied. Although the student did see four frames, he did not see all four in between the limits as determined by the use of Frame 20 previously.

Question C would not be satisfied for similar reasons given to B; i.e., he was correct on them all, but not within the two appearances of Frame 20. Notice that by adding "FROM 2" to Questions B and C, both would then be satisfied, i.e.,

B) IF FROM 2 4 SEEN 5, 12-35
C) IF FROM 2 3 RIGHT, 12-35

*All frames in between 7 and 10 need not occur for a match. The only requirement is that the frame numbers are sequential (i.e., the next number is greater than or equal to the preceding one) and that the two frames mentioned (7 and 10) appear. This is with respect to pattern-type questions and does not apply to forms using RIGHT, WRONG, ETC.

The CALC mode

The CALC mode can be entered via "#CALC" or "←" for the LD or student respectively. The following is a list and examples of legal CALC statements:

1. Compute an arithmetic statement

```
*2 + 2
4.ø
```

2. Use function names

```
*SQRT (19 + 6) + FACT (3) + COMB (4,3)
15.ø
```

3. Assign names

```
*X = 10 + SIN (45)
X = 10.707
*Y = X - 5
Y = 5.707
```

4. Change names

```
*CHANGE FACT TO FACTORIAL
DONE
*FACTORIAL (4)
24.ø
```

5. Cause the computer to assign names

```
*ASSIGN NAMES
WILL DO
*FACTORIAL (4)
L1 = 24.ø
```

6. Drop names

```
*DROP X

GONE
```

7. Ask for values not given

```
*Y + Z1 + Z2

ENTER VALUES (COMMAS BETWEEN) FOR:  Z1, Z2

#3, 4

L2 = 12.707
```

8. Change the precision

```
*2 PLACES

IN

*L2

L3 = 12.71
```

9. Print all options which can be concurrently used

```
*PRINT OPTIONS
 RESTORE  DEGREES  RADIANS  CHANGE  SET  TO  PRINT  DROP  PLACES
 NAMES  ASSIGN  FOR  MATRIX  ARRAY  READY  OPTIONS  VERBOSE  CONCISE
 SUM  PROD  RESET  FUNCTION  RANDOM  FINISHED  ALLOW  KEYWORD
 PHONETIC  GOTO  FORMULAS  WITHIN  PROHIBIT  TIME  FRAME  WAIT
 RESPONSE  F   C   R   B   P   D   I   CALC  CONTINUE  EXECUTE  GET
 SAVE  DISPLAY  RESTART  NOLINK  NONE  ADD  ALL  BUT  ONLY  USED
 NOT  RIGHT  WRONG  SEEN  MINUTES
```

10. Print all names which can currently be used

```
*PRINT NAMES
 N  VALUES  STEPS  HELP  RANK  SQRT  COMB  FACT  LOG  LN  SIN  COS
 TAN  COT  SEC  CSC  ABSOLUTE  TRUNCATE  ZTOP  PTOZ  PTOT  PTOX
 DATA
```

11. Define a matrix (maximum four dimensions)

```
*MATRIX (ABC, 10, 10, 10)

IN

*ABC (1,1,1) = L2

ABC = 12.71

Z = ABC (1,1,1) + Y

Z = 18.42
```

12. Set values or matrices into a protected area

```
*SET Z

DONE

*SET ABC

DONE
```

13. Drop all temporary names

```
*DROP NAMES

GONE
```

Now Y, L1, L2, L3 have all been erased. Only Z and matrix ABC remain together with the primitive names.

14. Store and use functions. (Any number of arguments are legal but the function must be single-valued.)

```
*FUNCTION XYZ (X) = 2X + 4

IN

*Y = XYZ (3) + Z + .58

Y = 29.Ø
```

15. Do summation, products and transformations

(The word DATA will be predefined by the program and the current values from the most recent problem will be preset.)

The following would produce a vector of standard scores.

```
*MEAN = SUM   DATA (I, 1) / 10   FOR(I = 1, 10)

MEAN = 3Ø.5Ø  (if the values were right)

*SUM   DATA (I, 1)  **2 / 10  FOR (I = 1, 10)

L1 = 133Ø.25  (M**N means M^n)

*STDDEV = (L1 - MEAN **2) ** (1/2)

STDDEV = 2Ø.Ø

IN

*Z SCORE (I) = (DATA (I, 1) - MEAN) / STDDEV FOR (I = 1, 10)
```

16. Set values permanently for the duration of the lesson (or until they are specifically dropped).

```
*SET PI = 3.14159

PI = 3.14

*6 PLACES

IN

*PI

L2 = 3.14159Ø
```

17. Stop the automatic name assignment

```
*DONT ASSIGN NAMES

NO MORE NAMES

*PI

3.14159Ø
```

18. Change the meaning of the trig. function arguments

```
*SIN (PI / 4) ** 2 RADIANS + COS (45) ** 2 DEGREES

1.Ø
```

There are certain other commands that can be used here which affect the operation of the program, e.g., BE VERBOSE and BE CONCISE will determine the level of explanation given during the lesson building mode,

To leave this computational mode and resume interaction (lesson building or answering questions as a student) the user either types ↑ and follows it with his next message or types READY Ⓒⓡ and the computer will request the next input. The use of either of the two exit methods always returns the person to the activity which he previously interrupted.

If an LD enters the computational mode another way, namely by typing #CALC Ⓒⓡ *, he will have all of the previously described capability <u>plus</u> that which follows:

19. He may change the name of any of his command list.

```
*#CALC

OK* CHANGE CALC TO COMPUTE

DONE

*# COMPUTE

OK*
```

20. When he permanently stores names of variables, matrices or functions, they are put in an area which is not accessible to the student. The LD can use these names in providing anticipated numerical answers to questions.

```
*FUNCTION MEANDIFF = SUM (DATA (I,1) - DATA (I,2)) / N(1) FOR
(I=1,N(1))

IN

*MEANDIFF

-2.Ø
```

The student mode cannot access that name

*When the # character is typed, the computer compares user identification to determine access to edit commands. If a student attempts to use this, access is denied.

```
*ALLOW MEANDIFF
```

Now the LD is making MEANDIFF available to the student as well.

```
*←MEANDIFF

-2.ϕ
```

In this way the LD can build new options for student use if he so desires.

The LD may leave this mode as before or by typing another edit command, e.g., #CONTINUE.

Many possibilities are probably evident to the reader such as mixing functions, matrices and numbers in a single expression. Your own ingenuity can fill in from here.

NOTE 1

TECHNIQUE USED FOR EVALUATING ALGEBRAIC EXPRESSIONS

With FORMULAS ON, PLANIT will attempt to match equivalent algebraic expressions. The technique is as follows:

1. PLANIT first searches the student's answer to see if all symbols (with the exception of numbers) in the LD's answer exist in the student's answer. If the student's answer has missing symbols, the answer is wrong. If the symbols are all present, PLANIT will go on to Step 2.
2. To all symbols in the LD's answer, it will assign successive prime numbers starting with 3, going left to right.
3. PLANIT will next perform the indicated arithmetic.
4. It will compare numerical answer with student's numerical answer (obtained by the same routine).
5. If the answers are the same numerically, student's answer is correct; otherwise he is wrong.

The following comparison will illustrate this process:

LD's ANSWER	POSSIBLE STUDENT ANSWERS	
A+(B+C)*D+5	a. A+(B*D+C*D)+5	right
	b. A+D*(C+B)+5	right
	c. 1+4+D*(B+C)+A	right
	d. D*(B+C)+5+3	wrong

If we go through the steps listed above, the LD's answer would work out as follows: First assign prime numbers to A+(B+C)*D+5. 3→A, 5→B, 7→C, 11→D. Then perform the arithmetic.

$$3+(5+7)*11+5 = 3+(55+77)+5 = 3+132+5 = 140$$

Student Answers a, b, and c are fine because they have all the symbols and, when prime numbers are substituted for the same symbols as in the LD's answer, the numerical result is the same. Answer d is wrong because it has one symbol missing (even though the numerical result would come out the same).

A+(B+C)*D+A+2 would also be correct since in this case A gets the value of three assigned to it. Note that this is not the right answer but gets through as such because of the technique employed. Care must be used to avoid this problem, although the chances are small that this could happen by accident. To insure the uniqueness of the answer obtained by this routine, two or more passes could be made through the LD's answer set assigning each time new numbers to evaluate. The student's answer would then be evaluated each time (once for each set of numbers). However, it was felt that the extra coding and processing time involved was not worth it.

NOTE 2

PHONETIC ENCODING*

The technique used for encoding a word phonetically consists of four steps or passes, shown below. (The word upon which Steps 2, 3, and 4 operate is the output from the previous step.)

A. Letter equivalent. The first pass maps all letters into their letter equivalents (i.e., every letter in Row 1 is transformed into the letter immediately below it in Row 2). All other characters go through unchanged, e.g., ?→?, 5→5, #→#, etc.

1. ABCDEFGHIJKLMNOPQRSTUVWXYZ (original letter)

2. ABCDABCHACCLMMABCRCDABHCAC (letter equivalent)

B. The H replacement. With the exception of the first letter in any word, the second pass transforms each H in the word to the letter which precedes it.

C. Elimination of successive identical consonants. The third pass eliminates all but the first element of an uninterrupted sequence of a single consonant, e.g.:

SITTING becomes SITING
SUCCESSIVE becomes SUCESIVE
IRRESISTIBLE becomes IRESISTIBLE

D. Elimination of all A's. All vowels have been mapped into A (Step 1), the fourth pass eliminates these, and the final word has no vowels.

Examples:

ORIGINAL WORD:	IRRESISTIBLE	PHONETIC	FONETIC	BILLBOARD
Step A	ARRACACDABLA	BHAMADAC	BAMADAC	BALLBAARD
Step B	ARRACARDABLA	BBAMADAC	BAMADAC	BALLBAARD
Step C	ARACACDABLA	BAMADAC	BAMADAC	BALBAARD
Step D (final)	RCDDBL	BMDC	BMDC	BLBRD

The user must by now be aware of the danger in using the phonetic routine: Undesirable words can sneak through as phonetical equivalents. For example: THINK and THOMAS are phonetically equivalent; i.e., THINK → DMC and THOMAS → DMC. Therefore, if the LD is looking for THOMAS EDISON as the answer and the student types in I THINK IT WAS EDISON (KEYWORD ON in this case), he would get full credit. Nevertheless, the routine has shown itself to be extremely useful in many areas of practical concern and, when used judiciously, can be most helpful.

*Patterned in concept from article: Hewes, W. L., and Stow, K. H., Information Retrieval by Proper Name, Data Processing Magazine, June 1965, 18-22.

The maximum number of letters that the routine can handle in a given word is 16. Additional characters remain unchanged.

NOTE 3

Summary and Basic Layout

1. Legal commands (#)
The first of any group of legal commands must be prefaced with a # character. Succeeding legal commands may be written without the #.

Legal commands perform four functions: switch operating modes, manipulate a lesson, editing the text of a lesson, and displaying student records.

A. Switching mode of operation

#CO) #CONTINUE)	places PIANIT in the lesson-building mode at the next available sequential frame number. (#CO is used to start building a lesson also.)
#EX) #EXECUTE)	places PIANIT in the execution mode so that the existing lesson will operate as the student will see it.
#RESTART	clears out data tables and starts over. PIANIT responds with P/Q/M/D/C.
#CALC	places PIANIT in the computational mode and makes available an unrestricted use of CALC. See the description of CALC for added detail. Note: #CALC and ← both cause one to enter the computation mode but ← is the more restricted student form.

B. Manipulating an existing lesson

#GET MATH) #SAVE MATH)	moves a lesson from disc into PIANIT or from PIANIT to disc. The name of a new lesson is assigned by the lesson designer (e.g., MATH). The lesson name may have two to five characters, must start with a letter,and may terminate with numbers if desired. Note: Duplicate copies of a lesson can be saved simply by assigning a unique name to each.
#GET MATH 1234) #SAVE MATH 1234)	moves a lesson from magnetic tape into PIANIT* or from PIANIT to magnetic tape; otherwise, the same as above. The reel number is specified in place of 1234. Use ØØØØ if a blank tape is desired. Contact the computer operator for the reel number of the tape thus assigned.
#GET MATH SMITH	Transfers the lesson MATH as used by student SMITH into PIANIT so that the records may be displayed. The LD must give the correct identification for both himself and the student.

#SAVE saves a previously named lesson back onto disc.

*Note: If the lesson is on disc, PIANIT will ignore the tape number and retrieve the lesson from disc. Use the system delete command and delete the lesson first or use another name.

The user will be asked to identify himself. The same rules hold for the identification of a lesson name: two to five or less characters, all letters, or letters followed by numbers.

C. Editing the text of a lesson
Lines may be inserted into a lesson, deleted from a lesson, or just printed on the teletype by specifying the line or lines and following that with an I, D, or P, respectively (Insert, Delete, Print). The critical part is specifying the line or lines. In the following examples a P will denote print but remember that an I or D could be used just as well for appropriate actions to occur.

a. Specifying a single line: #1, 2, 4, P (i.e., print line 4 in Group 2 from Frame 1).

b. Specifying several lines: #1, 2, 1-5, P (i.e., print all lines between 1 and 5 in Group 2 from Frame 1).

c. Specifying entire groups: #1, 2, P (i.e., print Group 2 from Frame 1).

d. Specifying several groups: #1, 2-4, P (i.e., print all groups between 2 and 4 from Frame 1).

e. Specifying an entire frame: #1, P (i.e., print Frame 1).

f. Specifying several frames: #1-4, P (i.e., print all frames between 1 and 4).

g. The general form of which a through f above were specific examples is:

```
#FRM1 - FRM2, GRP1 - GRP2, LINE1 - LINE2, COMMAND
```

where those frames, groups, and lines included in the specification are affected. Note the effect (above) of omitting certain optional parts of this general form.

Note: The absence of a command letter will be interpreted as a print (P).

Frame labels may be substituted for frame numbers where desired.

For inserting frames, specific frames, groups or lines must be designated. One cannot specify the insertion of three lines in Group 4 all in one Edit command, for example; but the insertion of an entire frame or entire group can be specified. Avoid using the "-" in insertion-type editing.

D. Displaying student records

```
#DISP   ) prints the records accumulated on the student who was identified
#DISPLAY) in the GET MATH SMITH command above.

#DISP FRM1 - FRM2    ) same as above but for a subset of the records that
#DISPLAY FRM 1 - FRM2) occur between the specified frames.
```

2. The frame types P/Q/M/D/C
The following paragraphs will explain what information to supply in filling in groups for the five types of frames, but remember, you are not required to supply an entry for every group. If you wish to skip any group, type a space and return. If you have finished building a frame before the groups are exhausted, type #CO or a dollar sign($).

P- The Problem Frame. Nine groups (frame header always Group 1).

Group 2. Used for data base names (currently for experimental applications).

Group 3. Option control ALL/ALL BUT (TAN,1-3), etc. The calculation options (functions) that are to be available to the student are specified here.

Group 4. Currently not in use.

Group 5. Steps to the solution. Each comment separated by a semicolon constitutes a separate step. These will be printed by student request, one comment for each request.

Group 6. Sample structure (O)ne, (T)wo independent groups, (M)atched groups, also student preference Y/N. e.g., Y - allows the student to control sample size.

Group 7. Population type to be used in generating the sample (information requested depends on the reply to Group 6). e.g., NN specifies a bivariate Normal population.

Group 8. Population parameters (information requested depends on reply to Groups 6 and 7) and sample size if N in Group 6.

Group 9. Sample header. The first line is for the data header remarks. The second line is for column location (place any character where the right edge of the column should line up). Remember that the first column will always be a tally column, numbering the rows of the sample. To delete this column, type a NO in place of its location. For example:

```
* BOYS -- GIRLS
* No X     X
```

(Q) The Question Frame

Two types - Constructed Response, POS/NEG

Constructed Response

Four groups

Group 2. Any questions or text can be put here.

Group 3. Enter all anticipated answers in one of two forms preceded by tags:

L ANS1 ... where L represents an alphabetic character tag for the answer, "ANS1", or

N ANS2 ... where N represents an integer tag for the answer, "ANS2".

Any number (up to 60) lines can be specified for Group 3. The answer tags must be A-Z or Ø-9. Identical tags may be assigned if the answers so designated are not to be distinguished from each other. A plus (+) sign immediately adjacent to any answer tag indicates that a student response which matches that answer will be counted correct. Several plus signs may thus be used.

Answer tags designate three functions:

- Tags A-Z imply that the answer will be matched exactly (or phonetically or minimally if PHONETIC and KEYWORD functions are used. See below).

- Tags 1-9 imply that the designated answer is a function of the arithmetic evaluation of the line. Simply stated, the line is put through CALC first and matching is attempted on the result. Any legal CALC statement may thus be implemented. KEYWORD ON, KEYWORD OFF, PHONETIC ON, PHONETIC OFF are four such legal CALC statements which appropriately set up methods of answer matching. Any CALC statement may be followed by the function WITHIN X where X represents a decimal number specifying an allowable error in the student's answer. For example:

  ```
  1+  MEAN(1)  WITHIN .001
  ```

  The answer to be matched is the numerical result of the MEAN function and must be exact within the allowable specified error to be counted correct.

  Most CALC statements will not cause messages to appear on the teletype as they normally do in CALC. However, exceptions to this are the CALC options PRINT and HELP, which will cause the corresponding messages to appear.

- Tag Ø should be interpreted in the same way as Tags 1-9 except for two important differences: (1) the associated

CALC statement will be done before the student responds to that frame rather than after; and (2) the result of the CALC expression will not be matched to the student's reply.

All answer tags (except for Tag Ø) are executed in order from top to bottom, regardless of any other sequence ordering, until a match, if any, is found. No further answers will be executed in that group after a match is found. Keep this in mind when using a dummy answer to turn off PHONETIC or KEYWORD.

Group 4. Action set. Four basic commands F:, B:, C:, R:. Each of the four commands, F:, C:, R:, and B: can be mixed as desired on a given line of actions. Each can also have entries following the colon (e.g., F: YOU AREN'T EVEN CLOSE. B: 5).

In describing how these commands operate with and without entries following the colons, remember that the next command word is not considered to be in the expression that follows a previous colon. In the above example, B is not included in the message, YOU AREN'T EVEN CLOSE. The colon following the B guarantees that B will not be considered in the previous message.

F: For specifying feedback. If no message follows F: PLANIT chooses one.

R: Provides for feedback messages same as F: but then waits for another answer. If no message follows R: PLANIT prints "WRONG TRY AGAIN."

B: Branch to frame number, frame label, lesson name, program name, or a CALC name of a frame number. NOLINK may be desired following the program name.

B: with nothing following it, is legal only when used as a "return" to previous lesson.

C: With nothing following it, causes PLANIT to print "THE CORRECT ANSWER IS: (the correct answer)." The only thing that can follow C: is a CALC statement. This allows the LD to perform a CALC statement as a function of the student's answer.

If these commands are preceded by the tag associated with the answer, they are performed only if the student's reply matched an answer bearing that tag. Commands not preceded by a tag are done only if no match was found.

Action commands for a tag must be completely stated on one line, e.g., F: message cannot run over onto another line. Any combination of actions can be used on any line and more than once, executed in the following order: F, C, R, B.

Tags occurring more than once means the action associated with the first occurrence is done first time through frame. Second occurrence is done second time through frame, etc.

POS/NEG form of a Question frame. Has five groups.

Group 2. Same as constructed response.

Group 3. General form: AAAA/BBBB where A's and B's indicate only two possible single-word answers. Unless specifically POS/NEG then any positive or negative answer is acceptable.

Group 4. LD must specify conditions under which AAAA is true. If conditions not met, BBBB is true.

The conditions for specifying truth is done by IF statements using CALC forms, e.g.:

IF 1+1 EQ 4. or CONT LQ FACT(33) AND...etc.

Group 5. Action frame same as Group 4 constructed response except that "+" and "-" are arbitrarily assigned answer tags (correct and incorrect). E.g.,

+F B:AHEAD (to be done if student is correct)

-R: RECALCULATE (to be done if student is wrong)

(M) The Multiple Choice Frame
The Multiple-choice frame is built in exactly the same way in nearly all respects as the Question frame. There are only two important distinctions: (1) there are no unanticipated responses. PLANIT will require the student to "CHOOSE ONE OF THE ABOVE LETTERS." (2) Only the answers bearing lettered answer tags will be printed as choices. Numbered answers will be executed so that work in CALC may be done but these will not be matched with student replies. PHONETIC and KEYWORD functions do not apply for multiple-choice frames. The + sign is used in the same way as in the Question frame to denote the correct answer(s) but will not be printed with the alternative answer-choice in which it appears. Periods may follow answer tags for sake of appearance if desired. E.g.,

```
 A. NORTH AMERICA
+B. SOUTH AMERICA
 C. AFRICA
 D. EUROPE
```

When the student sees this list the + sign will not appear.

(D) The Decision Frame
Used for decisions based on what the student has done across frames.

General form: IF, FRAME #, NUMBERED OR LETTERED ANSWER, FEEDBACK, BRANCH.

These are three general forms for decision-type statements: (1) those that are based on clusters of responses; (2) those that are based on specific responses; and (3) those that are based on the current state of CALC. A decision statement always begins with IF and usually ends with the commands F:, C:, and/or B:. C: if used, must be used with a CALC statement. If a statement has no commands, branch to the next frame is implied. Every line must start with an IF, AND or OR. No parentheses are allowed to group AND and OR conditions. In the following illustrations possible arrangements of the decision statements are shown. Vertical columns itemize options to occupy that position in the statement.

```
                 Optional entries***
             ___________________________
             GR
IF           GQ                RIGHT                            AND
AND     (    EQ     NUMBER     WRONG     FRAME(S)*     )        OR
OR           LQ                SEEN                             COMMAND
             LS                MINUTES

IF                  RIGHT                                       AND
AND     (    ALL    WRONG                FRAME(S)*     )        OR
OR           NONE   SEEN                                        COMMAND

IF                                                               AND
AND     (    FROM 2***, FRAME(S)*, TAGS** etc....)               OR
OR                                                               COMMAND
```

*Frames can be designated either by number or label and can be grouped by use of "-" (e.g., 4-7, A).

**Answer tags can be grouped after the comma by typing each tag so designated (e.g., 4-7, ABC 5-9, + , etc.).

***The primitive "FROM" in the decision question is optional. If not used, the search through the records is restricted to those frames the student went through after the last occurrence of this decision frame. If used, the search is restricted to start from the last time that frame was entered. If not found, the search starts from the beginning of all frames.

		GR		
		GQ		
IF		EQ		AND
AND	EXPRESSION	LQ	EXPRESSION	OR
OR		LS		COMMAND
		NQ		

IF				AND
AND	FUNCTION NAME	USED	FRAMES	OR
OR				COMMAND

(C) The Copy Frame
General form C N where N is frame number or label. C N means copy frame N.

APPENDIX 4

DESCRB: A LESSON IN DESCRIPTIVE STATISTICS

TABLE OF CONTENTS

I. INTRODUCTION

Computer assisted instruction (CAI) is particularly useful in statistics courses. The author of such instruction may harness the computer's computational power to generate a variety of sample data from populations with specified characteristics. The student may use this same power to reduce or eliminate computational drudgery in exercises involving this data. Such use of the computer, integrated with tutorial dialogue, is a distinctive feature of the CAI lesson described below.

DESCRB[1] provides a review or refresher training for the following topics in descriptive statistics: empirical frequency distributions, measures of central tendency (mean, median and mode) and the variance and standard deviation as indices of variability. The lesson was prepared and is executable with the PLANIT computer-assisted instructional system,[2] operating under SDC's time-shared computer system. Teletypewriter terminals located at SDC, or at remote locations, are used for two-way communication between users (students or lesson authors) and the system. DESCRB presents the student with individualized tutorial instruction, pseudorandom sample data for computational exercises, and computational assistance (through PLANIT's CALC mode) for these exercises. Data obtained from trial use of DESCRB by seven college students indicate that the lesson can be completed in one to two hours of on-line instruction.

II. PREREQUISITES

The student should complete the CAI lesson INTRO[3] before beginning DESCRB. INTRO builds up the student's skill in using the teletypewriter to communicate with CAI lesson material and introduces him to the mathematical conventions he must employ in the computational aspects of his subsequent instruction.

It is also advisable, though not essential, for the student to have recently completed a reading assignment in descriptive statistics before he begins DESCRB. The aforementioned students who first tried this lesson read Chapters 3-5 in

[1] Bobby R. Brown, student associate of System Development Corporation's Education and Training Staff during the summer of 1966, is the author of this lesson. Mr. Brown is currently a Ph.D. candidate at Pennsylvania State University.

[2] PLANIT is thoroughly described in: S. L. Feingold and C. H. Frye. User's guide to PLANIT. SDC document TM-3055/000/01, 1967. A brief description is available in the SDC brochure on PLANIT, BRT-500/003/00.

[3] INTRO was written by John Hawkins, student associate of System Development Corporation's Education and Training Staff, during the summer of 1966. A student can complete INTRO in 50 to 80 minutes.

J. P. Guilford's text[4] before beginning. These chapters include all the topics covered in DESCRB (as well as some material not included in the lesson). There are two brief references to this text which the student will encounter during his use of the lesson. At the outset he is asked, in the form of a printed message from the teletypewriter, whether he has completed the assigned reading in Guilford. (He is permitted to continue the lesson in any case.) Towards the close of DESCRB the student is referred to page 83 of the text for a derivation of the raw score form of the variance formula. The lesson can be easily edited to replace these references with others, for students using a different textbook.

Note: The student should also be furnished copies of the figures shown on pages 122 and 123 of this appendix. The lesson poses questions about these graphical illustrations which the student is required to answer via the teletypewriter.

III. OBJECTIVES

The lesson objectives are two fold: First, in descriptive statistics the objective is to provide a review of the most important elementary aspects of descriptive statistics for univariate data. The student who completes the lesson should have become familiar with the data classification and tallying procedures antecedent to the construction of histograms and frequency polygons. He must also exhibit competence in reading and interpreting such representations of sample data frequencies before progressing to instruction on measures of central tendency and variability. For the latter, the student's work is checked and guided (in computational exercises) until he exhibits an ability to compute correct values for the various measures, applying their definitions and/or computational formulae to sample data.

A second objective is to continue the student's introduction to the use of PLANIT's CALC mode for on-line computation which was begun in the lesson INTRO. The computational exercises in descriptive statistics thus have a dual purpose.

IV. LESSON DESIGN

The general pattern for each block of instruction is to present the subject information in a tutorial mode, followed by question-answering diagnostic sequences and, finally, computational exercises which use and amplify upon the subject matter. When the student's response indicates mastery of the subject matter, the lesson moves him forward to the next block of instruction. We will now describe the content and sequencing of lesson subject matter, followed by a discussion and examples of instructional presentation techniques.

[4] J. P. Guilford, Fundamental Statistics in Psychology and Education (4th edition). McGraw Hill Book Co., New York, 1965.

A. ORGANIZATION

The subject matter of lesson DESCRB is logically organized under three major topics of descriptive statistics:

- Frequency Distributions
- Measures of Central Tendency
- Indices of Variability

The following outline shows the order in which subtopics and teaching points within the major topics are introduced by the lesson. However, all students will not see the subject matter in the same detail. This will become apparent later.

1. Content Outline

Constructing and Using Frequency Distributions:

- Usefulness of ordering and grouping data.
- Concepts of "class intervals" and "frequency distribution."
- Derivation of frequencies; interval limits and tally marks.
- Two guidelines for setting up intervals.
- Three steps in setting up intervals
  - Determining the range of data.
  - Choice of interval size
  - Determining start point of the lowest interval.
- Concept of "exact limits" of intervals.
- Implications from frequency distributions
  - The "crude mode" as a most typical score
  - Shape of a distribution of test scores as indicative of test difficulty level.
- Graphical representation of frequency distributions
  - Using a histogram
  - Using a frequency polygon.

Measures of Central Tendency

- Definitions of "mode," "median," and "arithmetic mean."
- Determining the median value in a sample.
- Determining the modal value in a sample.
- Introduction of CALC functions and expressions for student use.[5]

  S(X) = sum of column X

  SS(X) = sum of squares of column X

  N(X) = number of scores in column X

  S(X)/N(X) = mean of column X

  (SS(X)-((S(X)**2/N(X)))/N(X) = variance of column X

- Computing the mean of a sample--student uses S(X)/N(X) or equivalent.
- Computing the overall mean of two combined samples--student uses (S(1)+S(2))/(N(1)+N(2)) or equivalent.
- Computing the "sum of row products" of two samples--Introduce student use of SRP function.
- Computing "sum of squared differences" of two samples--Introduce use of SSD function.
- Property of the mean; deviations from sample mean always sum to zero.
- Comparing summed deviations from the mean with summed deviations from an arbitrary point in a sample.

Indices of Variability

- "Central tendency" versus "variability"; need for a concept of variability.
- Comparing usefulness of "range" with "variance" and "standard deviation."

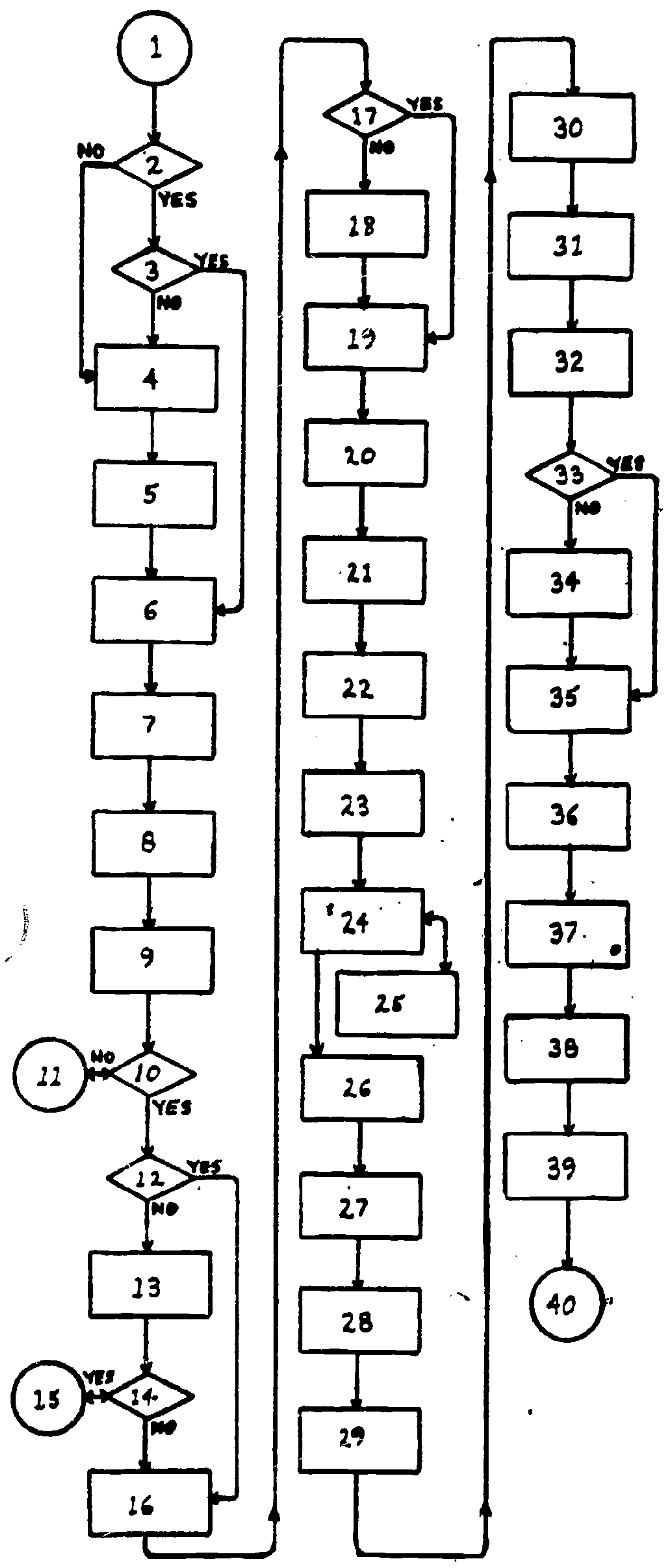
[5]The reader is referred to pages 126 and 127 of the section entitled Computational Exercises for a discussion of the students' use of these functions.

- Computing the variance of a sample from precomputed deviations--student uses SS(X)/N(X) or equivalent.
- Computing the standard deviation of a sample from precomputed deviations--student uses SQRT(SS(X)/N(X)) or equivalent.
- The effect on "variance" when deviations are measured from points other than the mean.
- Minimum value for sum of squared deviations occurs when deviations are measured from the mean; the "least squares" property of the mean.
- Computing the variance directly from raw scores--student uses (SS(X)-((S(X)**2)/N(X)))/N(X), or equivalent in discrete steps.
- Computing the standard deviation from raw score variance--student uses SQRT(variance value) or equivalent.
- Comparing variance and standard deviation values computed from raw scores with the same indices computed from deviation scores.
- Comparing usefulness of raw score and deviation-from-the-mean formulas.
- Introduction of two new CALC functions for student use
  - M(X) for mean of sample data in column X
  - SSDM(X) for sum of squared deviations from mean of sample data in column X
- Computing and comparing the mean of two samples, using the M(X) function.
- Computing and comparing the standard deviations of two samples, using SQRT(SSDM(X)/N(X)) or equivalent.

2. Lesson Structure

Figure 1 is a flow chart of lesson DESCRB. It shows the sequential development of subject matter, major decision points in the lesson, and sequences of remedial or review instruction. The descriptive phrases in Figure 1 are appended with frame numbers (in parentheses) to aid in locating information in the lesson, or in a lesson printout.[6]

[6] A complete lesson printout is beyond the scope of this document. Should the reader require a copy, contact Dr. Harry F. Silberman, System Development Corporation, 2500 Colorado Avenue, Santa Monica, California 90406.



1. Enter lesson (0-5)
2. Student confident of entry knowledge? (1-3)
3. Can student pass entry test? (4-9)
4. Class interval and frequency distribution (10-17)
5. Guidelines for setting-up intervals (18-20)
6. Setting up a frequency distribution (21-24)
7. Questions on frequency distribution (25,26)
8. Exact interval limits (27)
9. Crude mode and shape of distribution (28-30)
10. Does student have handout sheets? (31)
11. Student gets handouts from instructor
12. Can student use frequency table? (32)
13. Remedial instruction (33,34)
14. Student still having trouble? (33)
15. Student gets help from instructor
16. Using a histogram (35)
17. Can student use frequency polygon? (36)
18. Remedial instruction (37)
19. Define mode, median and mean (38,39)
20. Determining medians (39.5-39.8)
21. Determining the mode (39.9)
22. Bi-modal and uni-modal data (39.95)
23. Introduce CALC functions (40,41)
24. Computing means (41.3-43 and 47.5-51)
25. Remedial on use of CALC (44-47)
26. Computing exercises (51.3-51.4)
27. Sum of deviations from mean (52-59)
28. Indices of variability (59-64)
29. Computing variance (65,66)
30. Computing standard deviation (67)
31. "Least squares" property (68-72)
32. Computing raw-score variance (73)
33. Variance correct? (73)
34. Remedial exercise (74-77)
35. Computing raw-score standard deviation (78)
36. Compare raw-score and deviation solutions (79,80)
37. M(X) and SSDM(X) functions
38. Computing means (84-86)
39. Computing standard deviations (86-88)
40. End of lesson (89)

Legend:				
	○	Enter/exit lesson	▭	Interactive instruction and exercises
	◇	Lesson decision	( )	Lesson frame numbers

Figure 1. Lesson DESCRB Flow Chart.

Now, with reference to Figure 1:

- Upon entering lesson DESCRB, the student is asked if he has read material from the assigned text,[7] if he knows what a frequency distribution is, and if he can construct a frequency distribution. If he answers NO to the latter two questions, the lesson gives him instruction on how to set up frequency distributions (items 4 and 5). Otherwise, he is tested on his knowledge of class intervals and frequency distributions. If he passes the test (item 3), the lesson presents exercises in setting up and reading data from frequency distributions (beginning at item 6). Otherwise, the lesson presents the preliminary instruction noted earlier.

- Instruction on the graphical representation of frequency distributions requires that the student refer to prepared handout sheets. The student is asked if he has them (item 10). If not, the lesson advises him to obtain them from the instructor. Because subsequent questions relate specifically to this material, the lesson will wait until the student types YES before proceeding. The handouts, which contain fictitious data, are shown in Figures 2 and 3. The frequency distribution of language test scores is put on a separate page from the other two figures, because the lesson advises students not to look at the frequency distribution when using the histogram and frequency polygon.

- If the student's responses indicate that he is not able to use the frequency distribution handout, he is asked to locate a specific interval on the figure. If he can't find that interval, the lesson advises him to seek help from his instructor. It may be that he doesn't have the appropriate handout, or that he is looking in the wrong column. The lesson will wait at this point (item 14) until the student types YES.

- When the student can successfully read data from a frequency polygon (items 17 or 18), the lesson begins instruction on measures of central tendency (items 19 through 27). The student must next demonstrate his ability to use the CALC functions made available to him, by computing means of single samples and a combined mean of two samples. Then the tutorial exercises are extended to include computing a sum of row products, a sum of row differences, and a combined sum of row products plus a sum of row differences plus the sum of a column of data. In all of these exercises, if the student fails to obtain correct numerical results after several attempts, the lesson will finally print recommended expressions for the student to use in his computations. After that,

[7]*Op. cit.*, Guilford (Chapter 3, Frequency Distributions).

Table A. Frequency Distribution of Scores on a Language Test for 100 Freshman Girls.

Interval	Frequency
90-99	1
80-89	3
70-79	6
60-69	22
50-59	24
40-49	20
30-39	16
20-29	8
10-19	
0-9	1

Figure 2. DESCRB Handout--Page 1.

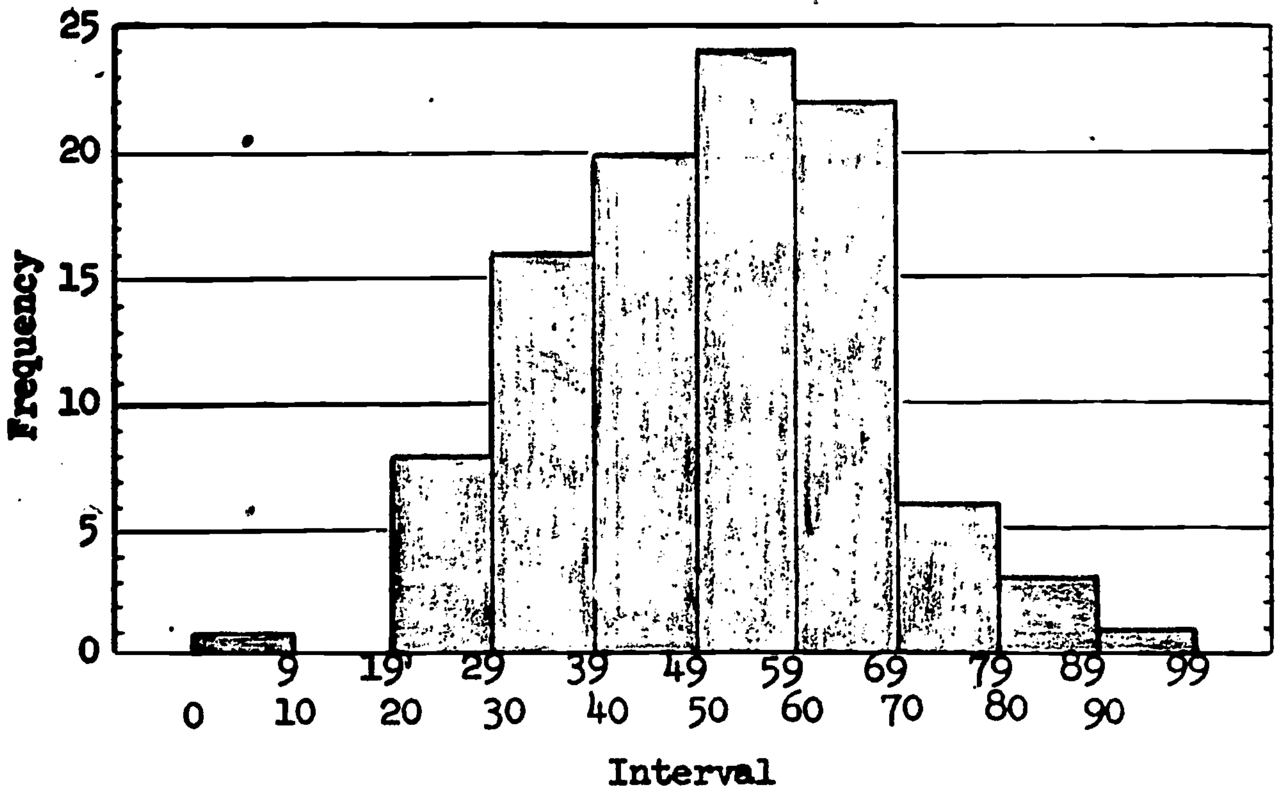


Figure A. Histogram for data of Table A.

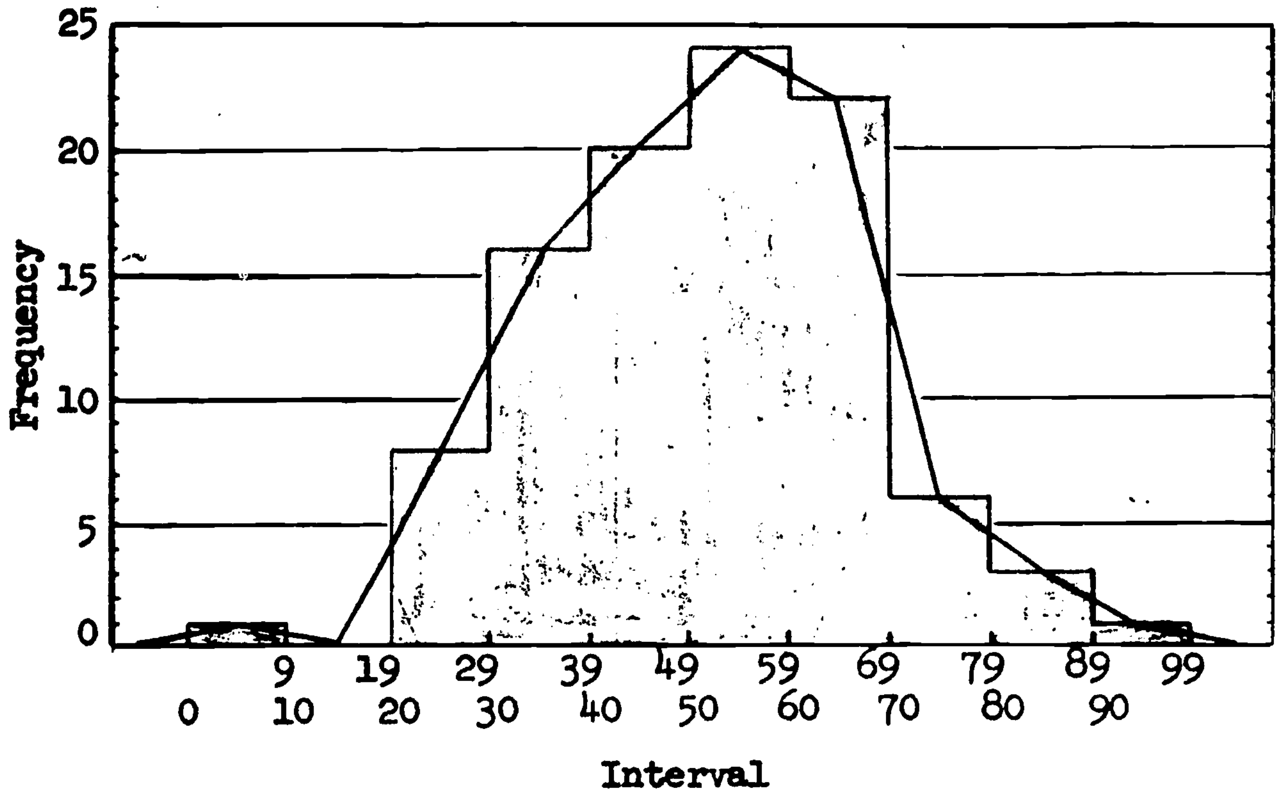


Figure B. Frequency polygon over histogram.

Figure 3. DESCRB Handout--Page 2.

the student must only observe the designated typing format to obtain the desired answer. If the student's answer indicates that he is having trouble on the first computational exercise, a special tutorial exercise (item 25) shows the student how to use a function for computing a sample mean with PLANIT's CALC mode.

- Finally, the lesson presents exercises which are designed to portray an important property of the sample mean (item 27)--the sum of deviations of scores from their mean is zero. The student first computes the sample mean for a presented set of scores, using CALC for this purpose; the "lesson" then presents the set of deviations of these scores from their mean and the student sums the set. The process is repeated when the student furnishes an arbitrary number between 5 and 15 which is added to his computed sample mean; new deviations are computed for the student using the original scores, but their deviations are now measured from a point in the sample other than the mean. The student is asked to compare the various summed deviations.

- To introduce the need for indices of variability (item 28), DESCRB asks the student to compute the means of two samples and observe that, while the means are quite similar, the samples are clearly different. Again, as with measures of central tendency, the student must demonstrate in tutorial exercises that he can compose mathematical expressions, for input to the CALC mode, to obtain numerical values for sample variances and standard deviations that are based on deviations from a sample mean. An exercise (item 31), similar to that for item 27, is used to demonstrate the "least squares" property of the mean. The student obtains and compares two variance values, one based on deviations from a sample mean and another based on deviations computed from an arbitrary point in the sample. Subsequent exercises (item 32) require the use of a raw score formula to compute variance. If several trials indicate that the student is having trouble with the composition of a CALC expression for the raw score variance formula, a step-by-step tutorial exercise (item 34) is presented. The student then compares his "raw score" numerical results with the variance and standard deviation obtained, for identical sample data, using deviations from the mean. Finally, the student is given two very simple CALC expressions (item 37), which he subsequently uses to compute and compare means and standard deviations.

B. FORM OF PRESENTATION

This section focuses on characteristics of the lesson's tutorial style, and covers computational exercises.

1. Tutorial Interaction

Lesson DESCRB uses a tutorial style of presentation in teaching new subject matter to students, in remedial and review sequences, and in all computational exercises. Student responses are continually evaluated and immediate feedback to the student is provided. Feedback for both correct and incorrect responses is generally quite brief; for example, NO, TRY AGAIN or VERY GOOD, YOU RING THE BELL.[8] On the other hand, incorrect student responses or response patterns can cause the lesson to activate remedial sequences where the interactive dialogue becomes more substantial.

Student responses anticipated and prompted by DESCRB are of several types, all quite short. Most of the questions that require a non-numerical response expect either a simple yes/no answer or the completion of a sentence by supplying the missing words. All of the computational exercises ultimately require the student to submit his answer in numerical form, either integer or decimal. (Students may obtain their numerical answers by inspection of data, by hand calculation, or by use of PLANIT's CALC mode.) Because each student's solution to computational exercises dealing with measures of central tendency and variability will be matched against a value in CALC, the lesson author has attempted to develop, in the student, the habit of responding numerically rather than with words. For the initial instruction on frequency distributions, each student's response that is formed as the verbal equivalent of a correct numerical answer is also accepted as correct, but the student is advised to submit subsequent answers in numerical form wherever possible. Thereafter, those frames which expect a numerical answer do not accept an equivalent verbal entry.

Example A, page 133, shows a tutorial sequence from lesson DESCRB. The student is to compute the mean of a column of scores, but forgets how to enter the CALC mode to input an expression for numerical evaluation. The lesson diagnoses the problem and provides remedial help to the student.

2. Computational Exercises

After completing instruction on frequency distributions, the remainder of DESCRB, dealing with measures of central tendency and variability, uses tutorial exercises for demonstration and computational practice. The student is presented one or more columns of sample data, assumed to represent scores like those on an achievement or ability test. The samples are presented to reinforce and supplement points made by the instructional text. Sometimes the sample data alone are sufficient, but most often the point is further demonstrated by requiring the student to perform calculations on the data, and/or by comparing different computational forms yielding the same numerical result.

[8] In this case, to emphasize the feedback statement, the lesson author actually caused the teletype bell to ring repeatedly as a part of the feedback.

The sample data for each exercise is either fixed in the lesson, such that it is always the same for all students, or is the result of a pseudorandom number generation process under control of the lesson. In the latter case, which includes all exercises on the mean, variance and standard deviation, the data presented will be different for each student. The data will also be regenerated whenever the lesson causes a given student to re-enter an exercise already performed. It can be seen, therefore, that the correctness of a student's numerical answer for the random data exercises cannot be evaluated by matching against a value pre-computed by the lesson author. Instead, answer matching is accomplished by author-defined functions, used in anticipated answer sets in lieu of numerical values. During lesson execution, PIANIT's CALC mode makes use of these functions to calculate anticipated answers, based on the actual data that each student is using in an exercise. More detail on the generation of sample data and use of PIANIT's CALC mode for answer matching is presented in Part V, "Technical Discussion."

To students, the exercise data in DESCRB takes one of the following forms, depending on the nature of the exercise:

- A single column of scores.
- Two columns of scores.
- One column of scores and another of deviations from the mean of the first column.
- One column of scores and another of deviations from a value within 5 to 15 score points of the mean of the first column.

For each one- or two-column sample presented, the student is asked to make certain calculations pertaining to measures of central tendency and variability (noted earlier, pages 121 and 124). The lesson informs all students of the functions available to them in PIANIT's CALC mode. They are as follows:

- S(X) ... sum of column X
- SS(X) ... sum of squares of column X
- N(X) ... number of scores in column X
- SSD ... sum of squared differences between two columns
- SRP ... sum of row products from two columns
- SQRT( ) ... square root of any value or arithmetic expression enclosed in parentheses
- M(X) ... mean of column X
- SSDM(X) ... sum of squared deviations from the mean of column X

It is assumed that the student has been oriented to the use of the first two functions and the standard arithmetic operators in lesson INTRO. Therefore, the S(X) and SS(X) functions are merely reviewed by DESCRB. The remaining functions are then introduced by DESCRB as needed for subsequent exercises: N(X) prior to computing means, SRP and SSD just before introducing the concept of deviations from the mean, and SQRT as part of exercises on computing the standard deviation. The M(X) and SSDM(X) functions are introduced only after the student has already practiced composing expressions equivalent to these functions.

In a few of the exercises, it is sufficient for students to use one of the functions alone to arrive at a numerical solution; e.g., SRP or M(X). However, most of the exercises require that students either:

- Combine the functions into a single expression which will yield a numerical result in one operation; e.g., S(1)/N(1) for the mean of column 1, SSDM(2)/N(2) for the variance of column 2

or

- Perform the calculation in separate steps; e.g., even though the student could compute a variance from raw scores in column 1 using the expression (SS(1)-((S(1)**2/N(1)))/N(1), he might prefer to find the variance in three steps·
    - S(1)**2/N(1)
    - (SS(1))-(value from first step)
    - (value from second step)/N(1)

The tutorial approach used in the exercise frames is to encourage the student to obtain numerical results from single expressions, suggesting the step-by-step methods only if the student cannot successfully combine the functions. Example B, page 139, shows a representative computational exercise from DESCRB. In this case, the problem for the student was to first compute a mean value for each of two columns of scores, where column 1 is a sample from a normal distribution and column 2 is a sample from a rectangular distribution. After noting the similarity between means, the "lesson" generated a new column of scores and a column representing deviations from the mean of those scores. The student was then instructed to compute the variance and standard deviation of the scores in column 1, using the deviations from column 2.

V. TECHNICAL DISCUSSION

We will now elaborate on techniques used by the lesson author to manipulate sample data for exercises, define computational functions for use by students and in lesson frames,[9] and determine whether the students' numerical answers are correct or incorrect. The reader who is not already acquainted with the capabilities of Problem Frames and the CALC mode of PLANIT[10] may wish to skip this section.

Population parameters governing the generation of sample data for computational exercises were specified in PLANIT Problem Frames. In these frames the author also specified sample size limits and printing formats for the data accompanying each exercise or series of exercises. As a result, the sample data in DESCRB is generated according to the specified population parameters, where the distribution of the population is either normal or rectangular. There are from eight to fourteen data entries pseudorandomly generated by the computer from the specifications for each sample. Format specifications in each Problem Frame set up the header titles and spacing between columns of data for teletype printout.

PLANIT automatically places the pseudorandom data into two identical matrices called VALUES (for use by the lesson) and DATA (available for manipulation). The number of rows in these matrices is determined by the sample size and the number of columns by the distribution (univariate or bivariate) specified by the lesson author in each Problem Frame of DESCRB. For example, if the frame calls for a one-column (univariate) sample of size N, VALUES and DATA each become N x 1 arrays, i.e., matrices with N rows and one column, or N x 2 matrices for two-column (bivariate) samples.

Each time that a Problem Frame specifies sample data for the matrices VALUES and DATA, the author has used CALC statements in lesson frames[11] to store an image of that data in a fixed dimension matrix (20 x 2) called FLIP, defined at the beginning of the lesson. To do this, the lesson author has defined the contents of FLIP as follows:

```
FLIP (I,1) = DATA(I) FOR(I = 1, N(1))
```

for one-column samples, and

[9] The subset of functions available to students was noted earlier under Computational Exercises, page 126.

[10] *Op. cit.*, Feingold and Frye (pages 34-40, The Problem Frame, and pages 54-79, The CALC Mode).

[11] *Ibid.*, Feingold and Frye (page 51, Using CALC Statements in the Frame).

```
FLIP (I,X) = DATA(I,X) FOR(I = 1, N(X) X = 1,2)
```

for two-column samples.

The number of observations in each column of data are put, by PLANIT, into a two-element array, called N such that N(1) becomes an acceptable way of designating the number of observations in column one and N(2), in column two. Normally, some of the higher numbered entries of FLIP remain unused since none of the sample sizes reach twenty.

The reader may be interested to know why a 20 x 2 matrix called FLIP was used, since FLIP is not one of the basic PLANIT components. FLIP was used to facilitate the writing of a uniform list of statistical functions that were made available to the student. They are uniform in the sense that they are identical in composition and use for each data manipulation problem of the lesson. Since some of the problems use two groups of data and others use only one, the PLANIT matrices, VALUES and DATA, will vary in their column dimensions from two to one. However, functions that operate on the data must reference the matrices according to their defined format; they must include two subscripts after a two-dimensional matrix, but only one after a one-dimensional matrix. If the functions are to be completely uniform and not vary in their composition for univariate and bivariate problems, then the data must also be kept in a matrix whose diemsnions do not vary. FLIP is such a matrix. By making all functions refer to FLIP, it is necessary for the lesson to immediately transfer any newly generated data into it. The lesson does this by using one of the statements noted above. Since all of the functions are written in such a way that they operate on the entries of FLIP, the composition of the functions do not vary throughout the lesson. The functions are defined as follows:

```
. S(X)=SUM FLIP (I,X) FOR (I=1, N(X))

. SS(X)=SUM FLIP (I,X)*FLIP (I,X) FOR (I=1, N(X))

. M(X)=S(X)/N(X)

. SD=S(1)-S(2)

. SSD=SUM (FLIP(I,1)-FLIP(I,2))**2 FOR (I=1, N(1))

. SSDM(X)=SUM SUM(FLIP(I,X)-J)**2 FOR (I=1,N(X) J=M(X`)

. V(X)=SSDM(X)/N(X)

. STD(X)=SQRT (V(X))

. SRP=SUM FLIP (I,1)*FLIP(I,2) FOR (I=1, N(1))
```

Thus by using FLIP, a matrix with static dimensions, to hold the sample data, the functions as defined above remain valid throughout the various kinds of problems, whereas the dimensions of VALUES and DATA may change with each execution of a Problem Frame.

All of the above functions, with the exception of SD, V(X) and STD(X) are made available to students at various points in the lesson. To obtain numerical solutions using PLANIT's CALC mode, the student must often compose and type in expressions consisting of one or more of these functions, arithmetic operators and other symbols; e.g., +, -, /, *, ( ), ←, ↑.[12]

The lesson author incorporated all of these functions into numerical answer lists of the lesson as a way of denoting correct and anticipated wrong answers to the various exercise questions. The use of functions, rather than absolute numerical answers, was necessary because each student receives different sample data for exercises. For example, in the answer list accompanying an exercise, the author specified such expressions as

S(1)/N(1) WITHIN .5

for the mean of column 1 data, or

SRP WITHIN .5

for the sum of row products obtained from two columns, or

S(2) WITHIN .5

for the sum of data appearing in column 2.

Assume for a moment that the answer list for a given lesson exercise contained both S(1) and S(2) as anticipated answers, and the student was to "FIND THE SUM OF COLUMN 1". Then if the student's numerical answer matched the computer's calculation of S(2), based on the current contents of the FLIP matrix, the lesson could come back with "NO, I SAID THE SUM OF COLUMN 1, NOT COLUMN 2."

In summary, this section has briefly covered how frames in lesson DESCRB are used to:

[12]The ← and ↑ symbols, respectively, are used by the student to enter and exit PLANIT's CALC mode. For example, the student might enter ←S(1)/N(1) with reference to some specific data. After receiving the numerical solution for the mean of column 1 scores, say 49.5∅, he would type ↑49.5 to enter his answer. This would return him from CALC to a lesson frame, where his answer could then be evaluated as correct or incorrect.

- control and manipulate sample data for exercises;
- define computational functions for use by students and by PLANIT;
- determine whether student numerical answers are correct or incorrect.

VI. EXAMPLE A

Tutorial Mode

Illustrates the tutorial mode of presentation in a remedial sequence on how to enter PLANIT's CALC mode and compute the mean of a column of scores.

Lesson DESCRB, Frames 43.ØØ through 47.ØØ.

Example A--Lesson Printout

```
FRAME 43.00  (Q)

G2. TEXT.
WHAT IS THE MEAN OF THE COLUMN OF SCORES ABOVE?

G3. ANSWERS.
1+ S(1)/N(1) WITHIN .5
A S(1)

G4. ACTIONS.
1 F: VERY GOOD.  YOU JUST BYPASSED A RATHER BORING REVIEW.B:AHEAD
-F: LET'S REVIEW A BIT.
A F: LET'S REVIEW A BIT.
- F: LET'S REVIEW A BIT.


FRAME 44.00  (M)

G2. TEXT.
     TYPE IN THE LETTER ASSOCIATED WITH THE FIRST STEP YOU WOULD
TAKE IN ORDER TO HAVE THE COMPUTER FIND THE MEAN OF THIS
SET OF DATA FOR YOU.

G3. ANSWERS.
A+ ENTER 'CALC'
B+ TYPE ←
C  SUM THE SCORES
D  TYPE 'S(1)/N(1)'

G4. ACTIONS.
AB F: B:B4
C B:B5
D B:B6


FRAME 45.00  (M) LABEL= B5

G2. TEXT.
     PERHAPS YOU DID NOT READ THE QUESTION CAREFULLY.  IF YOU WERE
GOING TO COMPUTE THE MEAN BY HAND THE FIRST STEP MIGHT BE TO SUM
THE SCORES.  HOWEVER THE QUESTION ASKED FOR THE FIRST STEP YOU
WOULD TAKE TO HAVE THE COMPUTER FIND THE MEAN FOR YOU.  WHAT ACTION
DO YOU TAKE TO PREPARE THE COMPUTER TO DO CALCULATIONS?

G3. ANSWERS.
A+ ENTER 'CALC'
B+ TYPE ←
C  SUM THE SCORES
D  TYPE 'S(1)/N(1)'

G4. ACTIONS.
AB F: THAT'S RIGHT. B:B4
CD F: NO. B:B4
```

Example A--Student Printout

```
WHAT IS THE MEAN OF THE COLUMN OF SCORES ABOVE?

*I'M NOT SURE WHAT TO DO FIRST
 LET'S REVIEW A BIT.

     TYPE IN THE LETTER ASSOCIATED WITH THE FIRST STEP YOU WOULD
TAKE IN ORDER TO HAVE THE COMPUTER FIND THE MEAN OF THIS
SET OF DATA FOR YOU.

A  ENTER 'CALC'
B  TYPE ←
C  SUM THE SCORES
D  TYPE 'S(1)/N(1)'

*C

     PERHAPS YOU DID NOT READ THE QUESTION CAREFULLY.  IF YOU WERE
GOING TO COMPUTE THE MEAN BY HAND THE FIRST STEP MIGHT BE TO SUM
THE SCORES.  HOWEVER THE QUESTION ASKED FOR THE FIRST STEP YOU
WOULD TAKE TO HAVE THE COMPUTER FIND THE MEAN FOR YOU.  WHAT ACTION
DO YOU TAKE TO PREPARE THE COMPUTER TO DO CALCULATIONS?

A  ENTER 'CALC'
B  TYPE ←
C  SUM THE SCORES
D  TYPE 'S(1)/N(1)'

*D
 NO.
```

Comments

The student was presented with a column of 12 socres in a prior frame and now is asked to compute a mean value for the scores. The correct student response is specified in Group 3 of Frame 43.ØØ as the expression S(1)/N(1) WITHIN .5. PLANIT will calculate the mean, using this expression, and the student's numerical answer will be compared. As can be seen from Group 4 of Frame 43.ØØ, any answer other than the correct numerical value for the mean, causes the student to enter a remedial sequence (Frames 44.ØØ through 47.ØØ).

Our student didn't attempt to calculate the answer, but indicated an uncertainty on how to proceed. Because of his next answer at Frame 44.ØØ, that frame directed him to Frame 45.ØØ (labeled B5). Note that Frame 44.ØØ might have sent him to Frames 46.ØØ or 47.ØØ (B4 or B6, next page) depending on his answer.

Example A--Lesson (cont'd)

```
FRAME 46.00  (M) LABEL= B6

G2. TEXT.
     YOU HAVE ANTICIPATED THE NEXT STEP, BUT YOU HAVE OVERLOOKED
A COMMAND WHICH MUST BE SENT TO THE COMPUTER BEFORE TYPING 'S(1)/N(1)'.
TRY AGAIN.

G3. ANSWERS.
A+ ENTER 'CALC'
B+ TYPE ←
C  SUM THE SCORES
D  TYPE 'S(1)/N(1)'

G4. ACTIONS.
AB F:THAT'S RIGHT. B:B4
CD F:NO. B:B4


FRAME 47.00  (Q) LABEL= B4

G2. TEXT.
     YOU MUST ENTER CALC BEFORE USING ANY OF THE BUILT IN FUNCTIONS.
YOU CAN ENTER CALC BY TYPING '←'.  RECALL THAT S(1) IS THE BUILT IN
FUNCTION FOR THE SUM OF COLUMN 1, AND N(1) IS EQUAL TO THE NUMBER
OF SCORES IN COLUMN 1.  A VERY EASY WAY TO COMPUTE THE MEAN IS TO
ENTER CALC AND TYPE 'S(1)/N(1)'.
\
FIND THE MEAN OF THE SCORES IN EXAMPLE 1.

G3. ANSWERS.
1+ S(1)/N(1) WITHIN .5

G4. ACTIONS.
1 F:
-R: NO. TRY AGAIN.
```

Example A--Student (cont'd)

```
     YOU MUST ENTER CALC BEFORE USING ANY OF THE BUILT IN FUNCTIONS.
YOU CAN ENTER CALC BY TYPING '←'.  RECALL THAT S(1) IS THE BUILT IN
FUNCTION FOR THE SUM OF COLUMN 1, AND N(1) IS EQUAL TO THE NUMBER
OF SCORES IN COLUMN 1.  A VERY EASY WAY TO COMPUTE THE MEAN IS TO
ENTER CALC AND TYPE 'S(1)/N(1)'.

FIND THE MEAN OF THE SCORES IN EXAMPLE 1.

*←S(1)/N(1)
52.1667

*↑52.1667
TRUE.
```

Comments

Frame 45.ØØ (prior page) branched our student to Frame 47.ØØ (labeled B4). The student demonstrates that he now understands how to enter the CALC mode of PLANIT and compute the mean.

End of Example A

VII. EXAMPLE B

Computational Exercises

Illustrates sample data generation, student's use of CALC functions in computation, and lesson author's use of CALC functions for evaluation of student's answers.

Lesson DESCRB, Frames 59.ØØ through 67.ØØ.

Example B--Lesson Printout.

```
FRAME 59.00  (Q)

G2. TEXT.
     SO WHEN DEVIATIONS ARE TAKEN FROM THE MEAN THEY SUM TO 0.0.
DEVIATIONS FROM ANY OTHER POINT WILL NOT SUM TO 0.
\
     THE MEAN TELLS US SOMETHING ABOUT A SET OF SCORES, BUT JUST
KNOWING THE MEAN LEAVES MANY QUESTIONS UNANSWERED.  IN THE
FOLLOWING SCORES YOU WILL OBSERVE THAT THE MEANS OF THE TWO COLUMNS
ARE QUITE SIMILAR, BUT THE TWO SETS OF SCORES ARE OBVIOUSLY
DIFFERENT.


FRAME 60.00  (P)

G3. OPTIONS.
ALL

G6. STRUCT./STUD. PREF.
T N

G7. DIST.
NR

G8. M1 R1 S1 N1 MS R2 S2 N2.
80 10 N 14 0 100 N 14

G9. HEADER.
ITEM NO.    COL.1     COL.2
    V         V          V


FRAME 61.00  (Q)

G4. ACTIONS.
C: FLIP(I,X)=DATA(I,X) FOR(I=1,N(X)  X=1,2)


FRAME 62.00  (Q)

G2. TEXT.
WHAT IS THE MEAN OF COLUMN 1?

G3. ANSWERS.
1+ S(1)/N(1) WITHIN .5

G4. ACTIONS.
1 F:GOOD. WHAT IS THE MEAN OF COLUMN 2?
-R: NO.  THE MEAN OF COLUMN 1 PLEASE.  TRY AGAIN.
```

Example B--Student Printout

```
     SO WHEN DEVIATIONS ARE TAKEN FROM THE MEAN THEY SUM TO 0.0.
DEVIATIONS FROM ANY OTHER POINT WILL NOT SUM TO 0.

     THE MEAN TELLS US SOMETHING ABOUT A SET OF SCORES, BUT JUST
KNOWING THE MEAN LEAVES MANY QUESTIONS UNANSWERED.  IN THE
FOLLOWING SCORES YOU WILL OBSERVE THAT THE MEANS OF THE TWO COLUMNS
ARE QUITE SIMILAR, BUT THE TWO SETS OF SCORES ARE OBVIOUSLY
DIFFERENT.

ITEM NO.   COL.1      COL.2
   1-       78         52
   2-       81         77
   3-       78         42
   4-       80         82
   5-       79        116
   6-       80        117
   7-       81         58
   8-       76        124
   9-       78         77
  10-       80         31
  11-       79         79
  12-       81        119
  13-       81         70
  14-       80         86

WHAT IS THE MEAN OF COLUMN 1?

*-S(1)/N(1)
79.4286

*179.4286
GOOD. WHAT IS THE MEAN OF COLUMN 2?
```

Comments

The student has just finished exercises on measures of central tendency and has compared the sum of deviations from a mean with the sum of deviations from an arbitrary point. He is now being introduced to the concept of variability.

Frame 60.00, a Problem Frame, calls for PLANIT to produce a two-column sample with 14 entries in each column. The samples are to be generated from the normal distribution with mean 80 and standard deviation of 10/3 for the first column, and for the second column from the rectangular distribution with identical mean and range 100. The specifications are contained in G.6, G.7 and G.8 shown on the left. The spacing between columns of data is specified with G.9. The statement "ALL" in G.3 denotes that all functions defined by the lesson author are to be made available to the student. Frame 61.00 then causes PLANIT to fill the FLIP matrix with the sample values in DATA, so that functions which make use of FLIP can now be used by the student and by PLANIT (see Part V, Technical Discussion, for further details).

The student is now asked to compute the mean of column 1. He enters CALC with the appropriate function, then enters his derived answer for evaluation. His answer is correct and he is asked to compute the mean of column 2.

Example B--Lesson (cont'd)

```
FRAME 63.00  (G)

G3. ANSWERS.
1+ S(2)/N(2) WITHIN .5

G4. ACTIONS.
1 F:
-R: NO.  TRY AGAIN.


FRAME 64.00  (G)

G2. TEXT.
     WE OFTEN WANT TO KNOW HOW MUCH VARIABILITY IS DEMONSTRATED
BY A SET OF SCORES.  WE COULD LOOK AT THE RANGE, BUT BECAUSE THE
RANGE IS DEPENDENT ON ONLY TWO SCORES IT IS NOT A VERY GOOD
INDICATOR OF VARIABILITY.  WE NEED A MEASURE OF VARIABILITY WHICH
IS BASED ON ALL THE SCORES RATHER THAN ON THE TWO EXTREME SCORES.
SEVERAL SUCH MEASURES EXIST.  WE WILL LOOK AT THE MOST IMPORTANT
OF THESE, THE 'VARIANCE' AND THE 'STANDARD DEVIATION'.
\
TYPE GO WHEN YOU ARE READY TO GO ON.

G3. ANSWERS.
A+ GO

G4. ACTIONS.
A F: OK, HERE WE GO.
-F: OK, HERE WE GO.


FRAME 65.00  (G)

G2. TEXT.
     LET'S GENERATE ANOTHER COLUMN OF SCORES AND A COLUMN OF
DEVIATIONS FROM THE MEAN.\
SCORES ,DEVIATIONS\

G4. ACTIONS.
C: RESET (VALUES,8,8)
C: VALUES(I,1) = TRUNCATE(RANDOM*100) FOR (I=1,8)
C: FLIP(I,1)= VALUES(I,1) FOR(I=1,8)
C:MEAN = S(1)/N(1)
C: VALUES(I,2)= VALUES(I,1)-MEAN FOR(I=1,8)
C:4 PLACES

FRAME 65.30  (F)

G9. HEADER.
ITEM   COL. 1   COL. 2
  X      X        X.XXXX

FRAME 65.50  (D)

G2. CRITERIA.
C:FLIP(I,2)=VALUES(I,2) FOR (I=1,8)
```

Example B--Student (cont'd)

```
*←(S(2)/N(2)
CLOSING PARENTHESIS MISSING.

*←S(2)/N(2)
80.50

*↑80.50
FINE.

     WE OFTEN WANT TO KNOW HOW MUCH VARIABILITY IS DEMONSTRATED
BY A SET OF SCORES.  WE COULD LOOK AT THE RANGE, BUT BECAUSE THE
RANGE IS DEPENDENT ON ONLY TWO SCORES IT IS NOT A VERY GOOD
INDICATOR OF VARIABILITY.  WE NEED A MEASURE OF VARIABILITY WHICH
IS BASED ON ALL THE SCORES RATHER THAN ON THE TWO EXTREME SCORES.
SEVERAL SUCH MEASURES EXIST.  WE WILL LOOK AT THE MOST IMPORTANT
OF THESE, THE 'VARIANCE' AND THE 'STANDARD DEVIATION'.

TYPE GO WHEN YOU ARE READY TO GO ON.

*GO
 OK, HERE WE GO.

     LET'S GENERATE ANOTHER COLUMN OF SCORES AND A COLUMN OF
DEVIATIONS FROM THE MEAN.

        SCORES   DEVIATIONS

ITEM    COL. 1   COL. 2
  1-     13      -27.4999
  2-     75       34.50
  3-      1      -39.4999
  4-     24      -16.4999
  5-     69       28.50
  6-     54       13.50
  7-     64       23.50
  8-     24      -16.4999
```

Comments

Again the student's computation of the mean value is evaluated by the lesson designer's specification in Group 3 of Frame 63.ØØ. Frame 64.ØØ introduces the concepts of "range," "variance," and "standard deviation."

Frames 65.ØØ and 65.3Ø then cause a new set of scores and the deviations from the mean of those scores to be generated and presented to the student. Frame 65.ØØ fills column 1 of the FLIP matrix with the new scores and Frame 65.5Ø fills column 2 of FLIP with the calculated deviations.

Example B--Lesson (cont'd)

```
FRAME 66.00  (0)

G2. TEXT.
     THE DEFINITION OF THE VARIANCE IS: 'THE SUM OF THE SQUA
DEVIATIONS FROM THE MEAN, DIVIDED BY N'.  SO IN ORDER TO COMPUTE
THE VARIANCE OF THE ABOVE SCORES WE SQUARE EACH VALUE IN COLUMN 2,
SUM THE SQUARES, AND DIVIDE BY THE NUMBER OF VALUES IN COLUMN 2.\
COMPUTE THE VARIANCE OF THE SCORES IN COLUMN 1.  (REMEMBER THE
FUNCTIONS YOU LEARNED IN YOUR FIRST LESSON.)

G3. ANSWERS.
1+ SS(2)/N(2) WITHIN .5
2 SS(1)/N(1) WITHIN .5

G4. ACTIONS.
1 F: VERY, VERY GOOD.
2 R:YOU SQUARED THE WRONG COLUMN, TRY AGAIN.
-R:NO.  'SS(2)' WILL BE HELPFUL IN THIS PROBLEM.  TRY AGAIN.
-R:NO.  TRY AGAIN.
-R:NO. TO GET THE CORRECT ANSWER TYPE '←SS(2)/N(2)'.


FRAME 67.00  (0)

G2. TEXT.
WHAT IS THE STANDARD DEVIATION OF THE SCORES?

G3. ANSWERS.
1+ SQRT(SS(2)/N(2)) WITHIN .5

G4. ACTIONS.
1 F: VERY GOOD. YOU RING THE BELL.
1 F: OK, YOU GOT IT THAT TIME.
-R: NO.  DID YOU USE THE FUNCTION 'SQRT'?  TRY AGAIN.
-R:NO.  STANDARD DEVIATION = SQUARE ROOT OF VARIANCE.  TRY AGAIN.
-R: TRY '←SQRT(SS(2)/N(2))'. I THINK YOU WILL GET IT THIS TIME.
```

Example B--Student (cont'd)

```
     THE DEFINITION OF THE VARIANCE IS: 'THE SUM OF THE SQUARED
DEVIATIONS FROM THE MEAN, DIVIDED BY N'.  SO IN ORDER TO COMPUTE
THE VARIANCE OF THE ABOVE SCORES WE SQUARE EACH VALUE IN COLUMN 2,
SUM,THE SQUARES, AND DIVIDE BY THE NUMBER OF VALUES IN COLUMN 2.

COMPUTE THE VARIANCE OF THE SCORES IN COLUMN 1.  (REMEMBER THE
FUNCTIONS YOU LEARNED IN YOUR FIRST LESSON.)

*←SS(1)/N(1)
2340.0

*↑2340
YOU SQUARED THE WRONG COLUMN, TRY AGAIN.

*←SS(2)/N(2)
699.7475

*↑699.7475
 VERY, VERY GOOD.


WHAT IS THE STANDARD DEVIATION OF THE SCORES?

*←SD(700)
226799.720

*↑226799.72
 NO.  DID YOU USE THE FUNCTION 'SQRT'?  TRY AGAIN.

*←SQRT(700)
26.4575

*↑26.4575
 OK, YOU GOT IT THAT TIME.
```

Comments

The student is then given a definition of variance and is expected to compute it using a "sum of squared deviations" solution. The anticipated answers are specified in Group 3 of Frame 66.00 as expressions of the constituent function SS(X), which makes use of the data in FLIP.

Our student inadvertently based his variance solution on the "score" column on his first try, which was sensed by the lesson due to the SS(1)/N(1) expression in Group 3 of Frame 66.00. His variance solution was correctly accomplished, using the column 2 "deviations" on his next try. In computing the standard deviation, the appropriate SQRT function was made explicit to the student for his second attempt.

End of Example B

APPENDIX 5

PROGM: A LESSON IN ON-LINE COMPUTER PROGRAMMING WITH THE TINT LANGUAGE

TABLE OF CONTENTS

I. INTRODUCTION

PROGM[1] is an instructional lesson which has been designed to teach students of statistics how to use TINT as a problem-solving aid.

TINT is a computer programming language[2,3] developed at System Development Corporation. It is particularly adaptable to writing a program "on-line" (i.e., while the user sits at a computer typewriter terminal) and then using the program immediately to obtain the necessary answers. TINT interprets a program of statements given to it to derive the results. The TINT interpreter refers to the total TINT system whereas a TINT program refers to a collection of statements upon which the TINT interpreter operates.

Often, computer programs are written over a long period of time and are reused many times, as would be the case for working out a company payroll. One normally expects to devote several days or weeks to producing such a program. However, one may only want to use the computer to do some arithmetic. He may want a programming language that will let him type his statements as fast and conveniently as possible and then "execute" (i.e., place the computer under the control of the program statements) without delay. TINT is ideal for this application. Therefore, the TINT interpreter is very useful for statisticians and students of statistics to use for performing the many tedious arithmetic operations that they frequently encounter.

PROGM was prepared through PLANIT, a computer-assisted instructional system that provides for the preparation of such lessons on the computer and also supervises the administration of the sequence to the student.[4,5] The purpose of the PROGM lesson is to teach students of statistics how to write TINT programs that would do some of the more common arithmetic tasks. PROGM gradually introduces conventions of the TINT language to the student so that, by the end of the lesson, he will have ceased to interact with the lesson but will, instead, be writing TINT programs.

---

1 The authors of PROGM were: C. H. Frye, F. D. Bennik, and S. L. Feingold, all members of the Education and Training Staff at System Development Corporation.

2 TINT, SDC brochure BRT-500/005/00.

3 Kennedy, Phyllis R. The TINT user's guide. SDC document TM-1933/000/03. 30 July 1965.

4 Feingold, S. L. and Frye, C. H. User's guide to PLANIT. SDC document TM-3055/000/01. October 17, 1966. 214 pp.

5 The computer programs referred to in this document have been implemented on the SDC AN/FSQ-32 time-shared computer, located in Santa Monica, California.

PROGM incorporates several features that allow the student to exercise partial control over the sequencing of the lesson presentation. He can review, request explanations and examples, or practice constructing and executing programs in TINT. These are made available in response to certain key control words that the student types. When he finishes with any of these options, he resumes the lesson again at a point where he can pick up the context of the lesson portion that was interrupted. In addition, when he requests the use of the TINT interpreter, TINT communicates performance data back to the lesson so that diagnoses can be made. Instructional time ranges from about five to fifteen hours.

II. OBJECTIVES

Assignments in statistics courses often require lengthy calculations for a single answer. Often the assigned problems constitute "special cases," to reduce the calculation requirements, and exercises for some statistical procedures are omitted altogether because of lack of time. If students could have access to a system such as the TINT interpreter, and learn how to use it, realistic assignments could be made for each kind of procedure, though the required expenditure of the student's time would not necessarily be increased.

The Pearson Product Moment Correlation Coefficient was chosen for a criterion exercise. To compute this correlation coefficient, one must compute many of the statistics needed in other statistical measures as well, including sums, sums of squares, sums of pair products, means, and variances. If the student could write a TINT program to do this calculation, he should be capable of using the TINT interpreter in computations for a wide variety of statistical procedures.

Satisfactory performance on the above exercise, the terminal goal of this lesson, would also insure that the student has learned how to use several important programming techniques, including: (1) how to declare and use item names; (2) how to make his program communicate with a typewriter terminal; (3) how to program logical branching; (4) how to program "loops"; (5) how to use array notation; and (6) how to write statistical formulas in program statement form. Even this relatively small part of the TINT language capability is adequate to program very sophisticated kinds of statistical calculations.

It is not the purpose of this lesson to teach the use or definitions of any statistical measures. It is assumed the student already knows the required ones (mean, variance, standard deviation and product moment correlation for sample data). In those few instances where statistical instruction does appear, it is there only to facilitate recall. There is no systematic presentation of statistical instruction.

III. LESSON DESIGN

The lesson can be described in terms of the organization and development of the subject matter, as well as the instructional strategies and modes of presentation employed. These are discussed separately.

A. ORGANIZATION

1. Topics

The PROGM lesson was subdivided into fifteen topics, each of which contributed information and practice that was considered to be essential to the realization of the terminal goal. Topics 1 through 3 are preparatory topics, Topics 4 through 13 are general programming topics, and Topics 14 and 15 pertain specifically to use of the TINT interpreter. Figure 1 gives topic titles and provides a highly simplified view of the subject-matter flow and relationships among topics.

Discussion of Figure 1:

- The lesson questions the student and assesses his prior experience with the teletype terminal and understanding of the time-sharing system commands and the PLANIT student mode. The student may need to see all (or parts) of Topics 1-3, he may need only a brief summary, or he may enter the TINT programming lesson (Topic 4) directly.

- After he enters Topic 4, subsequent topics continue to build upon terms, concepts and relations taught in prior topics. In general, the progress is basically linear through Topic 11.

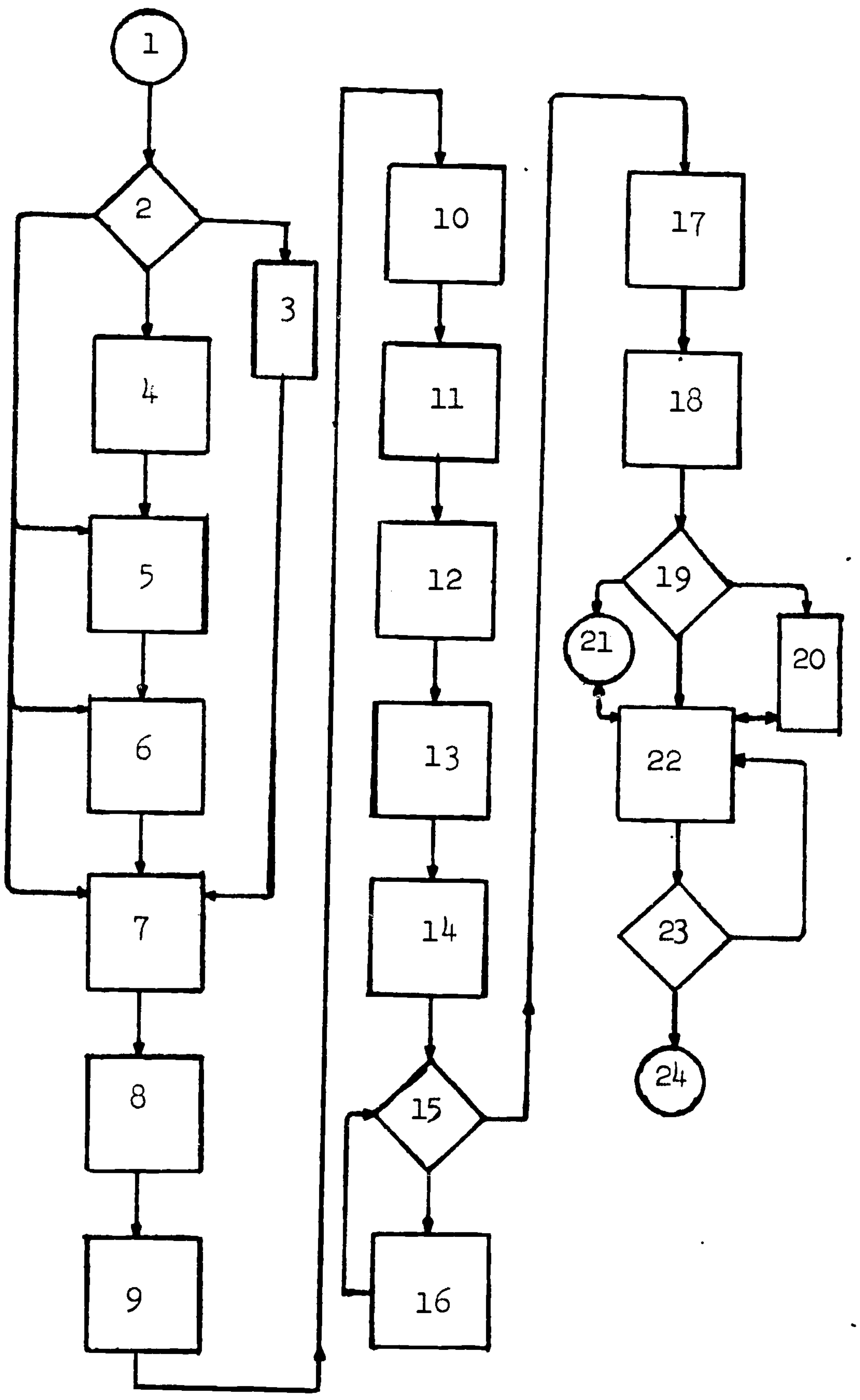
- The lesson determines each student's ability with basic arithmetic notation and operations. Then students either receive a brief summary of those points necessitated by the teletype-computer communication mode and proceed to Topic 13, or enter Topic 12, which presents rules of notation and exercises in the use of this notation for evaluating arithmetic expressions.

- In Topic 14, students learn a subset of the TINT interpreter language, which becomes their means of communication and control when they actually use the TINT program in Topic 15. They are also made aware of TINT command explanations and examples, which they can access on a command-specific basis both before and after entering Topic 15.

- Topic 15 presents a carefully graded series of tutorial programming exercises which gradually lead each student into the criterion exercise. The student makes use of the TINT program and may interact at any time with the Topic 14 TINT command explanations and examples.

2. Synopsis of Topics

This section describes the order of presentation and provides a summary of the subject matter for each of the 15 topics in PROGM.



1. Start lesson
2. What preparatory topics are needed?
3. Brief summary; Topics 1-3
4. Topic 1--The Teletype
5. Topic 2--TSS Exec. Commands
6. Topic 3--PLANIT Commands
7. Topic 4--Item Concept
8. Topic 5--Item Construction
9. Topic 6--Item Declarations
10. Topic 7--Use of Symbol, "="
11. Topic 8--IF-Statements
12. Topic 9--GOTO-Statements
13. Topic 10--Loops
14. Topic 11--AND/OR Connectives
15. Criterion test on arithmetic operations
16. Topic 12--Arithmetic Operators
17. Topic 13--Arrays
18. Topic 14--TINT Commands
19. Does student want to try EXPLAIN, EXAMPLE or TINT functions now?
20. TINT Command examples and explanations
21. Exit to the TINT interpreter
22. Topic 15--TINT Programming
23. Test achievement of terminal objective
24. End of lesson

Figure 1. Lesson Flow Chart.

Topic 1--The Teletype

- Use of the 'RETURN' key for entering messages.
- Use of the 'RUBOUT' and quotation mark keys for cancelling and correcting messages.
- Use of the 'ORIG' and 'CLR' buttons to turn the teletype on and off.

The student must know how to turn the teletypewriter terminal on and off, the distinction between upper case and lower case characters, how to use the RETURN key to enter messages, how to use the RUBOUT key for backspacing, and how to use the space key for canceling an entire line. He learns by practicing.

Topic 2--Time-Sharing System (TSS) Executive Commands

- Use of 'LOGIN', 'LOAD', and 'GO' commands.
- 'Program Mode' versus 'System Mode'.
- Use of quotation mark and exclamation mark to switch modes.
- TSS replies preceded by dollar sign.
- Use of 'STATUS' command.
- Use of 'DIAL' command.
- Use of 'QUIT' command.

As in Topic 1, certain conventions of the computer time-sharing system must be familiar to the student before he is ready to learn about programming. He must know how to use the LOGIN, LOAD and GO commands for getting onto the computer and starting his chosen program in operation. He receives, at his option, instruction about the STATUS and DIAL 9 commands for getting help when he needs it. The QUIT command is presented as a means for terminating contact with the TSS. He is also instructed on the use of the ! symbol to route messages to the TSS and the use of the " symbol to redirect messages back to the loaded program. Most of these conventions will be practiced as a demonstration of the achievement of the objectives. In the case of the LOGIN, LOAD, GO and QUIT commands, these are only explained since any practice would abort the operation of the lesson in a way that would only confuse the student.

Topic 3--PLANIT Commands

- Meaning of the asterisk as a PLANIT output (asterisk tells the student that PLANIT is "ready" for his reply).
- Usefulness of question mark to get help.
- PLANIT student modes (instruction or calculation).
- Use of left arrow ←, to switch to CALC and up arrow ↑, to return to instruction.
- Use of ←QUIT function.[6]
- Use of ←REVIEW function.[6]

Again, the student must learn certain conventions to obtain the maximum benefits from the PLANIT program as it presents the PROGM lesson. These conventions are: (1) the use of the ? symbol to elaborate on what the student is supposed to do (at any given point in the lesson); (2) the use of the ←REVIEW function, to allow the student to selectively browse over completed parts of the lesson; and (3) the use of the ←QUIT function to mark his place in the lesson between instructional sessions at the teletype. (The ← symbol precedes the function name in actual use to distinguish that reply from an attempted answer to a question. Note that QUIT, preceded by ← is different from the !QUIT command, explained in Topic 2 which is used to sever connection with the TSS.) The student practices these conventions until he can successfully use each one.

Since some students might have received all or part of the orientation contained in the first three topics from some prior experience, these are optional. The necessary conventions are itemized and the student is asked if he is already familiar with them. If he is, he can move on to Topic 4.

Topic 4--Item Name Concept

- Introduce concepts of program statements, data, storage locations, and names for locations.
- Usefulness of statements to manipulate data.
- Importance of statement ordering.

[6]See the discussion of certain special functions that are made available to the student in this lesson, beginning on page 167.

The goal of Topic 4 is to teach the student to associate item names with their corresponding computer storage locations where data are kept. The instruction consists of definition and analogy. Interaction with the student is restricted solely to concepts. Attainment of this objective is assessed in conjunction with Topic 7.

Topic 5--Constructing Item Names

- Five rules for composing item names.
- Single-letter versus multicharacter names.
- Assigning numerical values to names.
- Substituting item names for values.
- Using item names to compose algebraic expressions.

There are a few mechanical rules that one must follow in composing new item names. The goal of this topic is to get the student to memorize these rules and apply them both in constructing sample item names and in discriminating between samples that follow the rules and those that violate the rules.

Topic 6--Item Declarations

- Real, integer and Hollerith items.
- Difference between real and integer items.
- Distinction between input form and stored form of real and integer items.

In addition to learning to construct item names, the student must learn the conventions for appending a modifying word to the name (e.g., REAL, INTEGER, HOLLERITH) to form an acceptable item declaration. The modifier describes the item characteristics that a TINT program requires for proper interpretation of item content. The three modifiers of interest in this lesson are: INTEGER, REAL, and HOLLERITH. The first two have their usual connotation. HOLLERITH describes an item name that will represent a numerically coded collection of letters or symbols. The student is given practice exercises that are designed to bring out the characteristics of different item declarations. As a test of his comprehension, he is asked to deduce how some exemplary item names have been declared from the way the numbers that they represent are being interpreted. He is also required to compose some item declaration statements and these are checked.

Topic 7--Using the Replacement Symbol (=)

- Assigning values to item names (e.g., NUM = 3.∅).
- Successive modification of the value of an item name by incrementing or decrementing it (e.g., NUM = NUM + 1).

The primary objective in this topic is to teach the student the use of "=" in statements which prescribe the content to be stored in a designated computer memory location. The item name (previously declared) designates the location and must appear to the left of the equal sign. The item content is specified by the expression appearing to the right of the equal sign. These expressions on the right may be simple, consisting of a single integer, or they may be complex "formulae" composed of decimal numbers, arithmetic operation symbols and item names whose contents have been previously specified. The student is shown a use of the symbol (e.g., NUM = NUM + 1), for incrementing the contents of a given location. This use of the equal sign results in statements which would be false in mathematical contexts. The departure from mathematical usage is explained.

Students are asked to construct acceptable statements where the = is used, identify incorrectly composed statements, and predict program execution outcomes based on sample statements. Remedial help is provided and criterion questions determine whether the student should go on. Testing for this objective is also implicitly testing attainment of Topic 4 objectives.

Topic 8--Decision (IF) Statements

- Meaning of relational symbols; e.g., LS for less than, GR for greater than, etc.
- Differentiating true and false IF-statements.
- Implications of true and false IF-statements for program execution.
- Example of use of an IF-statement--"Absolute Value" function.
- Digression to discuss TINT statements:
  - Usefulness of READ and PRINT statements
  - Symbol for terminating statements
  - Multiple statements per line of input
- Optional digression to try out TINT.

The task of Topic 8 is to teach the student how to use the computer's decision-making capability. He must learn a fairly simple format for prescribing conditions which may appear in IF-statements and must learn how to trace the program execution sequence which results when the condition is true and when it is false. Also, the student must learn how to locate decision points and prescribe conditions in IF-statements so that his program will operate in the desired manner.

To test this objective, the student is given samples of IF-statements and is asked to state the outcome of their use. The number of exercises he sees depends on his performance. Then he uses samples of small programs in which IF-statements are crucial to their operation. Finally, he is asked to write a short program sequence incorporating an IF-statement. Remedial information is included in each of the exercises for those who answer incorrectly. Poor performance branches the student to review material from previous topics. In addition, he has the option of requesting a review at any time. The review capability is explained later.

Topic 9--GOTO Statements and Labels

- Usefulness of GOTO-statements.
- Pairing of GOTO-statements with an IF-statement to achieve a desired execution sequence.
- Need for and form of statement labels.
- Rules for composing statement labels; compare with item names.
- Single declaration for multiple items.
- Analysis of sample program logic and operations.

The student is taught to use the GOTO-statements to control the execution sequence of his program. The GOTO-statement is not a difficult topic but its usefulness is especially apparent when it is used in conjunction with an IF-statement. It is therefore an objective to teach the student how to use the IF- and GOTO-statements in pairs to achieve program branching as a function of the truth or falsity of IF-statement conditions.

A sample TINT program is used to illustrate the role of IF- and GOTO-statements in the process of summing a column of numbers of arbitrary length. The student is asked to identify several specific characteristics that the program exhibits (e.g., the entry point associated with the GOTO-statement and the effect on the statements, both before and after the entry point). These characteristics are governed by the IF- and GOTO-statements, and he demonstrates his understanding of the use of these statements in his answers. The student also "tries out" the program, operating it as though TINT was interpreting it.

Topic 10--Loops

- Iteration as a function of GOTO statements and statement labels.
- Analysis of a program loop structure.
- Statements within and outside the loop.
- Termination of loop as a function of the IF-statement condition.
- Importance of including a variable-valued item in the IF-statement.
- Danger of non-terminating loops.

A program loop is a special case of the IF- and GOTO statements, when they are used as a pair to cause a segment of the program to iterate a given number of times. The objective is to teach the student how to set up these iterative loops where a set of operations is to be repeated a number of times. Using the same illustration as in Topic 9, the characteristics of the loop are brought out through a tutorial sequence. No new information is presented for this topic since it is only a special application of one already covered. The student is asked to predict how some important conditions affect the implementation of a loop. For example, he is asked to identify initial and terminal conditions of the loop and predict the effect of designating the conditions for the loop in such a way that the loop variable never arrives at the terminal condition. A wrong answer triggers remedial help from feedback associated with Topic 10 frames, or a review of Topic 9.

Topic 11--AND/OR Connectives in Compound IF-Statements

- Review of simple IF-statement.
- IF-statements for comparison of numerical values of expressions.
- Usefulness of multiple comparisons.
- Logical operators 'AND' and 'OR'.
- Use of AND/OR connectives to form compound IF-statements.
- Implications of 'AND' versus 'OR' on truth or falsity of IF-statement.

Topic 11 extends the concepts of the IF-statement to include compound conditions connected by logical AND and OR operators. The objectives and the criteria are very similar to those for Topic 8, though related specifically to IF-statements that contain these connectives. Again, the students are asked to state the outcomes of sample IF-statements and, in addition, are given exercises in identifying various components of the compound IF-statement samples. The students are tutored until they demonstrate error-free performance on a sequence of exercises before they are allowed to go on.

Topic 12--Arithmetic Operators

- Form and meaning of seven operators; e.g., + for addition, - for subtraction (or changing the sign of an item), * for multiplication, ** for exponentiation, / for division, SQRT for square root, ABS for absolute value.
- Standard arithmetic operations; order of execution.
- Use of parentheses in algebraic expressions.
- Optional practice in the use of parentheses.

The two objectives of Topic 12 are (1) to identify the keyboard symbols that comprise the mathematical notation for programming and (2) to test the student's ability to use the basic rules of algebra that is assumed to be among his entry skills. These rules pertain to the order in which arithmetic operations are performed, both with and without explicitly placed parentheses. Though these are supposed to be entry behaviors, remedial exercises are also provided for those who cannot pass the criterion test, since experience has shown that these rules have often been forgotten.

Topic 13--Arrays

- Item name defining a block of storage locations.
- Array size restricted by the array declaration.
- Numerical subscript to designate elements in an array.
- Optional digression to review Topics 4, 5, 6.
- Subscript bracketing related to teletype keyboard.
- Comparison of TINT subscript conventions with mathematics subscript conventions.
- Substituting letter subscripts for numerical subscripts.
- Permissible value assignments for letter subscripts.
- Form of array declaration.

- Programs that accomplish the same function with and without arrays:
  - Comparative analysis of program operations
  - Comparative analysis of advantages in use
  - Primary usefulness of arrays
- Application of array to a computational program.
- Programming exercise incorporating an array, to review and synthesize all topics.
- Programming optimization; coding economy, efficiency of execution.

In this topic, the student must learn how to declare a number array, how to subscript the array, how to keep the subscripts within legal boundaries, and how to use the subscripted array in a TINT program. Information from each of the previous topics can easily be identified in Topic 13. The writing of program loops plays an especially important role in setting up the subscripting for the arrays. The sample TINT program (first seen in Topic 9) is modified to include the use of arrays.

The student will answer a number of questions about subscripting. Each question has remedial material associated with it for those who answer incorrectly.

Then the student compares illustrations of two TINT programs, one that uses arrays and one that does not. He is asked to identify characteristics that depend on the use of the arrays. He also practices with each type of program.

The evaluation of student performance for this topic consists of two parts. First, using the sample program that illustrates arrays, certain alterations are made to the program and the student is asked to predict what effect it will have on the operation of the program. Second, after he passes the first test, he is presented with a new illustrative program in which some key statements have been omitted. He is told how the program is expected to function, and must then fill in the missing statements, using what he has learned about arrays and subscripting.

An alternate track is provided through Topic 13 for those who report acquaintance from previous instruction with the array subscripting conventions often encountered in matrix algebra.

Topic 14--TINT Interpreter Commands

- Long form and short form of eleven TINT commands.
- Distinguishing TINT commands and TINT statements in terms of form and use.
- Review of program that uses arrays from Topic 13 and discussion of multiple statements-per-line.
- Contrast use of 'START' and 'CONTINUE' commands.
- Use of 'PRINT' command and 'PRINT ALL' form.
- Differences between 'PRINT' commands and 'PRINT' statements.
- Command utility, input forms, and TINT responses presented in the context of a program coding exercise, wherein a need is introduced for the use of each of the following eleven commands: START, SAVE, LOAD, PRINT variations, INSERT variations, CONTINUE, HALT, DELETE variations, RENUMBER, EXECUTE, and TRACE.
- Explanation of 'TINT', 'EXPLAIN' and 'EXAMPLE' student functions; optional student tryout of functions before entering Topic 15. (For an explanation of these functions, see the section entitled "Special Functions" beginning on page 167.)

The objectives for Topic 14 will be satisfied if: (1) the student demonstrates a clear, intuitive notion of the utility of eleven TINT commands for program coding, editing, execution, saving, and recalling, as well as for TINT program control; (2) he can input commands with correct spelling, punctuation, and grammatical construction as required by the TINT program input rules; (3) he knows the exact TINT response variations to expect, as a function of how he uses the eleven TINT commands; and (4) he can differentiate between TINT commands and TINT statements.

Corresponding to each of the four objectives, the student demonstrates his achievement in the following ways: First, problem situations are presented that create a hypothetical need for the use of each of the eleven commands. The student must choose the appropriate TINT command form to satisfy the need and thereby resolve the problem. In some cases, only one command is appropriate, while in others, more than one command alternative is possible. The students' command choices are tutorially evaluated until the correct choice is apparent.

Second, after choosing a command appropriate to a problem situation, the lesson requires the student to input the command from the teletype, just as if he were in direct communication with TINT. The input is tutorially evaluated until he finds the correct input.

Third, some problems require that the student obtain a particular response from TINT. The student continues manipulating the command language until the desired outcome is achieved. For each student command input, the lesson simulates the TINT program output.

Finally, following tutorial instruction, criterion questions test student mastery. Remedial instruction is provided, as required.

Topic 15--Writing TINT Programs

- Writing a program that will raise numbers to different powers and sum the results.
- Writing a program that will multiply pairs of numbers from two lists and sum the products.
- Criterion test programming exercise: computation of means, variances and Pearson Product Moment Correlation Coefficient.
- For each of the above exercises:
  - Optional analysis of programming errors detected by TINT
  - Check whether the student's numerical answers are correct or not after program execution
  - Help in correcting program logic as implied by answers or as requested by student
  - Performance summary for student

This topic consists of working out assigned exercises using TINT programs that the student writes and executes, operating the TINT interpreter and interacting with the PROGM lesson only to have his results and errors (if any) analyzed. There will also be help available by request as he does the exercises.

PLANIT has the capability of accessing other computer programs so that it can handle the mechanics of making the TINT interpreter conveniently available to the student without disturbing his work in the lesson. The lesson author simply names the desired program as an object of a branch in the lesson and conveys information to and from the program through a resident matrix called LINK. This capability is explained in more detail in the section entitled, "Technical Discussion."

As the student completes an exercise, he is given the next one. If he comes up with a wrong answer, the lesson attempts to analyze the error that probably caused it and he repeats the exercise. As the final exercise, the student writes a TINT program that will calculate a Pearson Product Moment Correlation coefficient for two columns of data, where the number of observations is one of the parameters of the program.

It is possible that the student may continue to make programming errors, despite the feedback given to him based on an analysis of his errors. If such is the case, remedial instruction is available to cover general problem areas, as well as specific steps toward solution. Entrance into the Topic 15 remedial sequences may be by student choice, or by a lesson decision based on student performance. The student can also return to Topic 14, to retrieve explanations and examples of the use of TINT interpreter commands, by using the ←EXPLAIN and ←EXAMPLE functions, which are discussed later in the section entitled, "Technical Discussion." If the student exhausts all available remedial and review material, but is still unable to perform the exercises correctly, the lesson will advise that he seek outside help.

What has been presented thus far reflects the sequential development of the available lesson subject matter. It is important to note that not all students will necessarily receive all available subject matter--many decisions of this type are made by the lesson--and that order and pacing of the presentation is often under student control.

B. PRESENTATION

The PROGM lesson authors developed the subject matter using three basic modes of presentation:

- Tutorial
- Expository
- Exercise

Both tutorial and exercise sequences are characterized by a requirement for continual student-lesson interaction. The expository mode utilizes interaction only as a means for controlling the lesson presentation. In the tutorial sequences, authors evidenced the use of both deductive and inductive techniques in developing subject matter.

1. Tutorial

In PROGM, the tutorial approach is used extensively to present new subject matter, as well as in special remedial and review sequences. The essence of this mode of presentation is constant interaction between the lesson and each student. Characteristics of the interaction are a function of individual ability with the subject matter, within constraints of the lesson's sensitivity to student responses from a teletype. The lesson continually assesses individual responses and response patterns to determine if subject matter review is indicated, or if the student could benefit from information available but not yet seen. In other cases, the student is questioned regarding his desire or ability to continue, and he is given choices by which he can exert considerable control over his routing through the lesson topics.

The majority of tutorial sequences use a deductive technique in developing the subject matter; that is, from information presented by the lesson, the student is expected to form certain conclusions. The student's acqusition and understanding of information are evidenced in problem-solving situations presented by the lesson. The lesson uses student responses as indicative of what is needed to help the student reach the desired conclusions. To a lesser extent, some instructional sequences provide a more inductive approach to the student's acquisition of information. For example, the student is first asked to solve a problem. He responds, and, as necessary, new information is introduced to aid the problem-solving process. After the student correctly completes the exercise, the lesson organizes and explains the information the student should have acquired from the exercise.

The tutorial sequences prompt student-lesson dialogue in a number of ways. There are multiple-choice questions, which require choice of a single answer to a direct question. Another multiple-choice format requires the student to choose which statements from a list best meet the conditions specified by a question. Constructed response questions may require a discrete word, number, algebraic expression or phrase as a response to a direct question. Other constructed response formats ask for various types of completion. For example, in one case students are asked to complete a sample program by listing those program statements and algebraic expressions that are missing. Student responses anticipated and evaluated by the lesson vary from discrete verbal and numerical entries, to more stylized or specialized forms of communication; e.g., algebraic expressions, TINT command phrases, special structure and symbology of TINT programming code.

Examples A-1 and A-2 (Section V) illustrate two tutorial sequences from the PROGM lesson.

2. Expository

A relatively small segment of the lesson uses a presentation mode that is basically expository. This is true in a portion of Topic 14, where the student interacts with the lesson only to obtain explanations or examples of TINT commands, on a command-specific basis. Once obtained, all explanations are expository, and all examples simulate the TINT interpreter. Tutorial dialogue in this case seems unnecessary. Earlier in Topic 14, the student is taught the use of TINT commands in an interactive tutorial mode. It is assumed that the TINT command examples and explanations will be accessed if the student wants to refresh his memory or obtain greater detail on the use of any TINT command while writing TINT programs in Topic 15.

Example B (Section V) shows a representative expository sequence from Topic 14.

3. Exercise

All lesson material introduced in a tutorial mode requires that the student undertake some sort of practice that makes use of the subject matter. This practice takes four general forms:

- Tutorial drill, to clarify and reinforce a term, concept, or relationship discussed specifically by the subject matter;
- Problems that cause the student to make analyses and to synthesize the subject matter elements already presented;
- Exercises in TINT communication;
- Exercises in TINT programming.

In the third form of practice, students may be making inputs either to the lesson, or directly to the TINT interpreter. For the former, the lesson expects a legal TINT input and simulates TINT responses. For all practical purposes, the student feels that he is interacting with the TINT interpreter, yet has considerable tutorial feedback available as a function of his exercise input. Conversely, the exercises in TINT programming do not provide tutorial assistance to the student until each exercise has been completed.

Examples C-1, C-2 and C-3 (Section V) are examples of the last three forms of exercise noted above. They are drawn from lesson topics 13, 14, and 15.

C. SPECIAL FUNCTIONS

1. The REVIEW Function

The REVIEW function is accessible to the student at all times, beginning with Topic 4. To operate the REVIEW function, the student would type ←REVIEW. This can be done at any time that the computer is expecting a reply (regardless of which reply might be expected). The ← symbol prefaces the word REVIEW to distinguish that reply as a student request rather than a usual reply.

Having typed ←REVIEW, the student is then asked for the number of the topic he wishes to review and the lesson resumes at the beginning of that specified topic. He is allowed to choose any topic number up to and including the largest number that designates a topic he has completed. Even while he is in the course of a review, if he wishes to discontinue that review, he can type ←REVIEW, in which case he will be asked if he wishes to review a different topic. If so, he will again be asked for the topic number and resume at that topic. If not, he will re-enter the lesson sequence near the place where he had originally asked for the review. The exact re-entry point is determined by the REENTRY function, described later.

If he continues in the REVIEW mode through an entire topic, he is asked, at the end of that topic, if he wishes to review another topic and, as described above, he will either resume at the topic of his choosing or will be placed back into the lesson by the REENTRY function.

Figure 2 shows the flow chart for the REVIEW function.

2. The QUIT Function

The QUIT function is available to the student at all times while he is working on the lesson. Just as with the REVIEW function, he can employ this function at any time that the computer is expecting him to reply by typing ←QUIT. The QUIT function marks the student's place, saves a record of his responses and path through the lesson,[7] and terminates the lesson. When the student resumes the lesson, he follows the same procedures that he used when he originally started. However, this time PIANIT will find a record of previous work with his identification and will resume the lesson at a point near where he last quit, a point determined by the REENTRY function. The REENTRY function is described later.

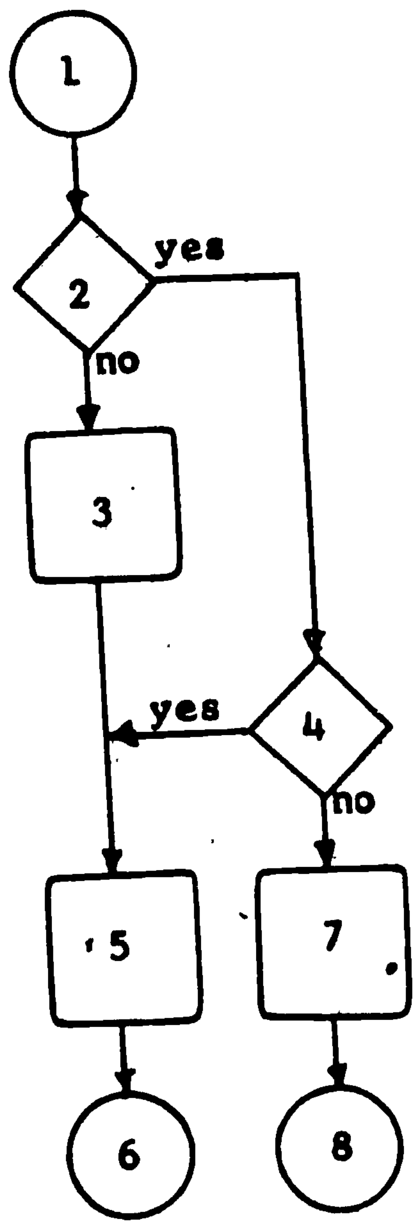
Figure 3 shows the flow chart for the QUIT function.

3. The EXPLAIN Function

Following instruction on the TINT command language (TOPIC 14), the student is "told" via PROGM that three new options are now available to him: EXPLAIN, EXAMPLE and TINT. He is also informed via the lesson that he can try any one (or all) of these options before entering Topic 15, or wait until a need arises after entering Topic 15. To activate these functions, the student merely types the function name, preceded by a left arrow (←) and followed by a carriage return. For example: ←EXPLAIN.

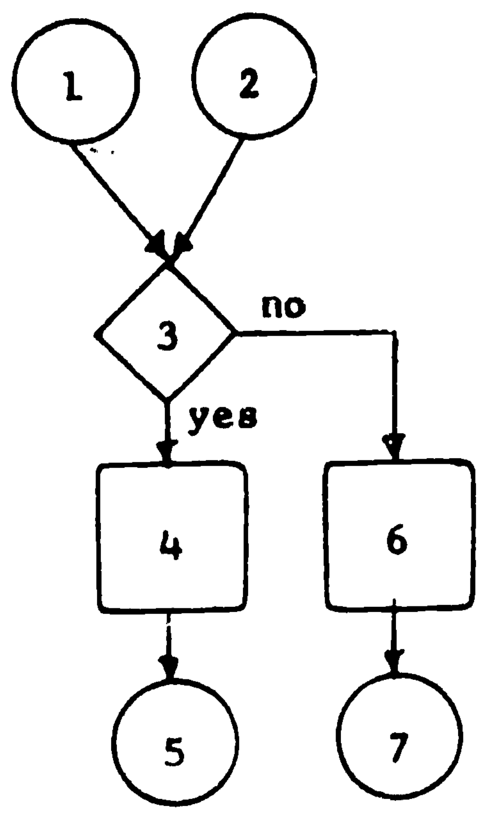
The purpose of the EXPLAIN function is to allow a student to obtain readily an expository explanation of the uses, forms, input rules and TINT responses associated with any of eleven TINT interpreter commands. After typing ←EXPLAIN, the student accesses the explanations on a command-specific basis. He is asked which command he wants explained and may enter either a number from a list of commands, the legal TINT long form, or the legal TINT abbreviation of the command. After receiving the explanation, he may either see an example of the use of that command, obtain an explanation or example of a different TINT command, or return directly to the lesson mainstream. Figure 4 is a flow chart for the EXPLAIN function.

[7]In addition to responses and path this record also contains some of the work the student may have done using PIANIT's CALC mode, e.g., a statement defining a mathematical function.



1. Entry point from any frame (by typing ←REVIEW).
2. Is the lesson in the REVIEW status?
3. Save the previous frame number, change the status to REVIEW and present the list of topics.
4. Does the student want to review another topic?
5. Have the student select the topic to be reviewed.
6. Branch to the requested topic.
7. Change the status to non-REVIEW and use the REENTRY function to locate the re-entry frame number.
8. Return to the lesson.

Figure 2. Flow Chart for the REVIEW Function.



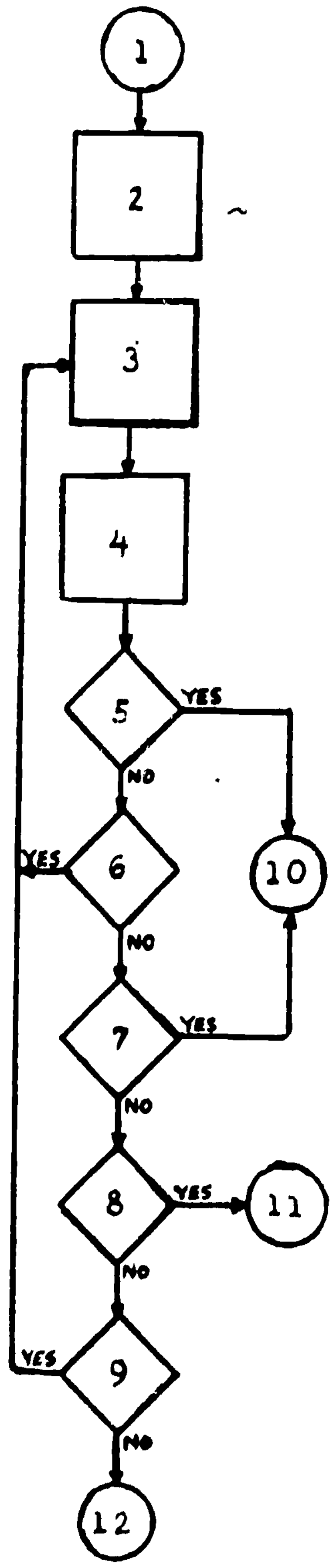
1. Entry point from any frame (by typing ←QUIT).
2. Entry point for next session.
3. Is the lesson now in a QUIT status? (No if entry is from 1.; yes otherwise.)
4. Change status to not-QUIT and use the REENTRY function to locate the re-entry frame.
5. Return to the lesson.
6. Save the number of the frame encountered by the student immediately preceding 1. and change the status to QUIT.
7. Use the PLANIT command, FINISHED.[8]

<u>Note</u>: All seven actions are contained in one PLANIT frame. The FINISHED command causes the lesson to terminate and resume at this frame on the next session, explaining why 2 is the new entry point. It also causes all necessary student records to be saved so that the student can pick up where he had left off upon returning.

Since the FINISHED command causes the student to repeat the frame from which the FINISHED action was taken, the effect of the manipulations in boxes 4 and 6 is to ensure that the answer in box 3 will always be "no" when branched to from the lesson, and "yes" when the frame is re-entered at the start of the new session.

Figure 3. Flow Chart for the QUIT Function.

[8] *Op. cit.*, Feingold and Frye, p. 70.



1. Entry point from any frame (by typing ←EXPLAIN).
2. Command list presented.
3. Student selects a command.
4. Appropriate explanation is found and presented.
5. Does student want example of that command?
6. Does student want explanation of another command?
7. Does student want example of another command?
8. Did student type ←TINT?
9. Did student type ←EXPLAIN?
10. Activate EXAMPLE function.
11. Activate TINT function.
12. Go to Topic 15.

Figure 4. Flow Chart for the EXPLAIN Function.

4. The EXAMPLE Function

This function allows a student to easily obtain one or more examples of a TINT command as used in a sample programming task. The examples simulate the TINT command inputs and the resultant TINT outputs, both bearing explanatory annotations for the student. After typing ←EXAMPLE, the student accesses the examples on a command-specific basis by typing either a number from a list of commands, the legal TINT long form, or the legal TINT abbreviation of the command. After seeing the example requested, the student may decide to see an example of another command or return to the programming lesson (Topic 15). Or, he might decide to get a command explanation by typing ←EXPLAIN. Figure 5 is a flow chart for the EXAMPLE function.

5. The TINT Function

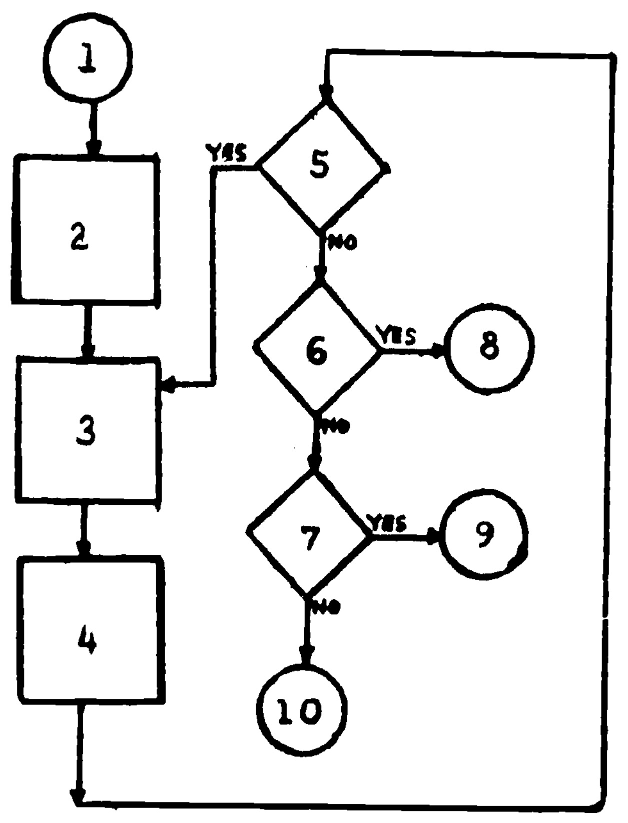
The TINT function becomes available to the student as soon as he completes Topic 14. As with the previously explained functions, he can activate the TINT function by typing, ←TINT at any time (after Topic 14) that the computer is expecting him to make a reply. In using the TINT function, the student is requesting access to the TINT interpreter in which he can write and execute his TINT programs. When he is through using TINT, he types READY, and he ceases to operate TINT. Before he resumes the lesson, if he has made one or more programming errors while working in TINT, he will be asked if he wants his errors analyzed. An affirmative reply will cause his errors to be listed and he can then select those that he wants to have explained. When he has received as much error information as he requests, he resumes the lesson near the point where he left off, the exact place being determined by the REENTRY function (see below).

Figure 6 shows the flow chart for the TINT function.

6. The REENTRY Function

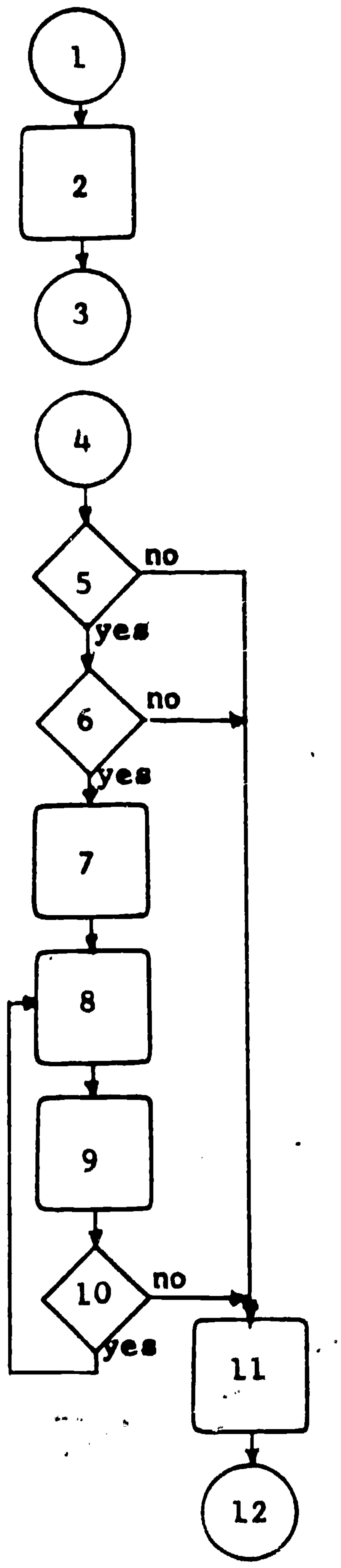
The REENTRY function is not presented to the student anywhere in the lesson. He will probably not be aware of its existence.

There are a number of functions that the student may use at his discretion, each of which will interrupt the normal sequence of his lesson (e.g., the functions REVIEW, QUIT, EXAMPLE, EXPLAIN, and TINT). The interruption may last less than a minute, several minutes, or several days. It would not be unusual for the student to quit for the day while he is in the process of a review. When he finally finishes his review, several days may have elapsed. To make the student resume at the exact point in the lesson where he had left off is inadequate. The questions being asked are probably closely tied to some context--an illustration or previous dialogue. The REENTRY function provides a means for the lesson author to designate re-entry points that constitute "punctuation marks" in the lesson. When a student is ready to resume at some given point in the lesson, the REENTRY function causes him to resume at the closest designated re-entry point preceding the point in the lesson where he had left off, repeating enough of the lesson to pick up the context again.



1. Entry point from any frame (by typing ←EXAMPLE).
2. Command list presented.
3. Student selects a command.
4. Appropriate example is found and presented.
5. Does student want example of another command?
6. Did student type ←EXPLAIN?
7. Did student type ←TINT?
8. Activate EXPLAIN function.
9. Activate TINT function.
10. Go to Topic 15.

Figure 5. Flow Chart for the EXAMPLE Function.



1. Entry point from any frame (by typing ←TINT).
2. Save the previous frame number and set up the conditions for TINT.
3. Call in TINT.
4. Return from TINT (by typing, ?READY).
5. Any errors?
6. Does the student want them analyzed?
7. List the errors.
8. Have the student select the error he wishes to have explained.
9. Explain the indicated error.
10. Explain any more errors?
11. Use the REENTRY function to locate the re-entry frame.
12. Return to the lesson.

Figure 6. Flow Chart for the TINT Function.

7. The RECOVER Function

The RECOVER function is only available to the user who can supply the authorized lesson-author identification. The purpose for this function is to allow such a user of PROGM to override the lesson sequence for any given student and fix the student's records so that, on his next session, the student will resume at a designated topic. The need to use this function could arise either from the loss of the student's records, which are automatically kept, or from a judgment that the sequence prescribed by the lesson is not appropriate.

The RECOVER function is also very useful during the checkout of a new lesson when the student who is making trial runs encounters a snag in the lesson and cannot proceed. Using the RECOVER function, the monitor can move him to another place in the sequence so he can continue.

When an authorized person wants to alter a student's record, he uses the PLANIT command, "GET",[9] supplying the lesson name, student identification, and author identification. After he receives the message, "LESSON SUCCESSFULLY LOADED" he types ←RECOVER. He is then asked for the topic number at which the student should resume. Having given that, the computer replies "RECOVERED. SAVE HIS RECORD NOW." This is done by using the PLANIT command, "SAVE",[10] in the way that "GET" had been used. The student resumes exactly as if he had quit at the beginning of the designated topic number.

IV. TECHNICAL DISCUSSION

The technical discussion of the PROGM lesson will be most meaningful to those who are familiar with the PLANIT system.[11] The reader who desires more information is referred to the PLANIT user's manual.

A. LESSON STRUCTURE

The PROGM lesson is broken down into six lesson segments to conform to the maximum space restrictions imposed by PLANIT. The segments contain from 50 to 150 frames each, depending on the average number of characters in the frames of that segment. A PLANIT frame is usually equivalent to several typical programmed instructional frames, of the kind found in booklet form, since a single PLANIT frame may also contain remedial information for anticipated wrong answers. Also, one PLANIT frame might require only a few seconds or over an hour to answer, depending on the nature of the question posed there. For these reasons,

[9] *Ibid.*, p. 45.

[10] *Ibid.*, p. 46.

[11] *Op. cit.*, Feingold and Frye.

a count of the frames in the lesson is not an adequate indication of its length either temporally or spatially. The estimated time that a student requires to complete his instruction is a useful measure of lesson length. More empirical data is needed to establish this, but trials of large segments with undergraduates at San Fernando Valley State College indicate that the PROGM lesson represents from five to fifteen hours of instruction, depending on the student.

The six segments are united not only by their subject matter, but also by ten LINK items that convey information from one segment to the next. In the PROGM lesson, the LINK items are used to convey lesson progress, requests for review, to designate entering frame numbers, remedial work prescriptions, and error reports (mainly from the TINT interpreter). Through these LINK items, the six segments are integrated into one lesson.

Information is conveyed through the LINK items by number codes. TINT communicates its error information through such a code--each error condition that has occurred has a designated number associated with it that is transmitted through the LINK items. The called segment then interprets the numbers according to the conditions that they represent. The same is true for other information that is transmitted through the LINK items.

For the reader who has access to the PROGM lesson through SDC's Q-32 time-sharing system, or has a printout of the lesson segments,[12] Figure 7 provides an index of those frame numbers, within a segment, associated with each lesson topic.

B. FUNCTION STRUCTURE

The six functions, REVIEW, QUIT, EXAMPLE, EXPLAIN, TINT, and RECOVER, are CALC functions in PLANIT, each of which makes use of the GOTO primitive. Essentially, each function (RECOVER excepted), when it is used, causes the present place to be marked (by "saving" the current frame number) and the GOTO causes control to be passed to a designated frame where the particular request is processed. The request often involves the exchange of segments, though this is handled automatically so that, at most, the student may be aware of a brief pause in the lesson presentation.

The REENTRY function is really a combination of a CALC function and a REENTRY matrix, consisting of a vector of re-entry frame numbers. When the student is finished with one of the above functions (REVIEW, EXAMPLE, etc.) another function

[12]Presenting the entire lesson printout would yield a frame-ordered listing of all frames, groups and lines of information comprising the lesson. A complete lesson printout for the TINT programming lesson is beyond the intent and scope of this document. Should the reader feel that such a listing is necessary, he should contact Harry F. Silberman, System Development Corporation, Santa Monica, California 90406.

	LESSON SEGMENTS					
Lesson Topics	A	B	C	D	E	F
1 The Teletype	3-6.9					
2 TSS Exec. Commands	7-14					
3 PLANIT Commands	15-20					
4 Item Name Concept		28.1-28.9				
5 Constructing Item Names		29-37				
6 Item Declarations		37.5-43				
7 Use of Symbol "="		44-63				
8 IF-Statements		64-81.5				
9 GOTO Statements		81.9-93.8				
10 Loops		93.9-104				
11 AND/OR Connectives		105-117				
12 Arithmetic Operators	25-56		118-126			
13 Arrays			127-213			
14 TINT Commands				1.8-29	57-90	
15 Writing TINT Programs						190-346

Note:

The cells in Figure 7 contain the "begin" and "end" frame numbers for lesson topics 1-15, subdivided among the six lesson segments A through F. This shows in which segments and blocks of frames the subject matter is developed for each topic; that is, the subject matter outlined in the section entitled "Synopsis of Topics." For example, if the reader has a printout of lesson Segment A, he could find the subject-matter frames dealing with the "Time-Sharing System" (Topic 2) by looking at Frames 7.00 through 14.00. The frame numbering in any given segment is independent of frame numbering in other segments. The entire range of frame numbers within each lesson segment is not reflected by Figure 7; there are many frames that do not deal with the topical subject matter, but are solely for purposes of setup and control of lesson processes. Also, the reader cannot determine the number of frames from this data; e.g., there are 63 frames in the block of frames numbered 1.80 through 29.00 of Segment D.

Figure 7. Lesson Structure

scans the REENTRY matrix, looking for two entries that bracket the frame number that was originally saved. When these entries are found, control is passed to the frame bearing the smaller of the two numbers from the REENTRY matrix. If the original frame number is the same as one of the entries in the REENTRY matrix, control is transferred to that frame.

The QUIT function is a variation of the PLANIT command, FINISHED. The use of QUIT causes the current frame number to be saved and control is transferred to a frame where the student is "FINISHED." When he returns, he is sent back to the proper "re-entry" point. There are two primary advantages to this indirect use of the FINISHED command: (1) it allows the student to be re-entered at a designated point where the lesson context can be re-created; and (2) it provides a common frame through which all students will pass at least once in every session--the frame from which they were previously FINISHED. This technique allows the lesson author to insert modifications into the lesson at a point where he can be sure that all students will encounter them.

V. EXAMPLES

Sample frame sequences have been drawn from lesson PROGM as representative of the presentation modes discussed in Part III-B of this paper.

For each example, the left-hand page shows a printout of the lesson content as input by the lesson authors, and the right-hand page shows the teletype printout of a hypothetical student-lesson interaction based on the lesson content. Explanatory and technical comments are also provided on the right-hand pages, beneath the printout.

EXAMPLE A

TUTORIAL MODE

Example A.1--Constructed Response Question from Topic 8.

Example A.2--Constructed Response Question from Topic 14.

EXAMPLE A.1--Lesson Printout

```
FRAME 75.00  (0) LABEL= UNCOND

G2. TEXT.
   HERE ARE TWO STATEMENTS OUT OF A PROGRAM...
\IF NJM LS 0\NUM=-NUM\ETC.\
ASSUMING THAT SOME PRIOR STATEMENTS GAVE A VALUE TO 'NUM' AND
STATEMENTS FOLLOWING THOSE ABOVE USE THE VALUE THAT HAS BEEN ASSIGNED
TO 'NJM', IN WHAT WAY DO THESE TWO STATEMENTS CHANGE THE VALUE OF 'NUM'?

G3. ANSWERS.
0 KEYWORD ON
A+ ABSOLUTE
A+ POSITIVE
A+ PLUS
B NO
B I KNOW
B WILL NOT
A+ +
D CHANGED
D CHANGE

G4. ACTIONS.
C:6 PLACES
A F:RIGHT.  IF 'NUM' IS POSITIVE OR ZERO, NOTHING WILL HAPPEN BUT IF
F:'NUM' IS NEGATIVE, IT WILL BE CONVERTED TO '-NUM' WHICH IS POSITIVE.
F:THIS IS COMMONLY KNOWN AS THE 'ABSOLUTE VALUE' FUNCTION.
B F:POSITIVE VALUES OF 'NUM' WILL NOT BE CHANGED BUT WHAT WILL HAPPEN
R:TO NEGATIVE VALUES?
-R:WILL POSITIVE VALUES OF 'NUM' BE CHANGED BY THE STATEMENTS?
C-F:IF 'NUM' IS POSITIVE OR ZERO, THE 'IF' STATEMENT WILL BE FALSE AND
F:THE 'NUM=-NUM' STATEMENT WILL BE SKIPPED.  WHAT WILL HAPPEN IF 'NUM'
R:IS NEGATIVE?
-F:IF 'NUM' IS NEGATIVE, IT WILL BE CONVERTED TO '-NUM' WHICH IS
F:POSITIVE.  THIS IS COMMONLY KNOWN AS THE 'ABSOLUTE VALUE' FUNCTION.
D R:WHAT WILL THEY BE CHANGED TO?
```

EXAMPLE A.1--Student Printout

```
   HERE ARE TWO STATEMENTS OUT OF A PROGRAM...

IF NUM LS 0
NUM=-NUM
ETC.

ASSUMING THAT SOME PRIOR STATEMENTS GAVE A VALUE TO 'NUM' AND
STATEMENTS FOLLOWING THOSE ABOVE USE THE VALUE THAT HAS BEEN ASSIGNED
TO 'NUM', IN WHAT WAY DO THESE TWO STATEMENTS CHANGE THE VALUE OF 'NUM'?

*NOTHING
WILL POSITIVE VALUES OF 'NUM' BE CHANGED BY THE STATEMENTS?

*NO
POSITIVE VALUES OF 'NUM' WILL NOT BE CHANGED BUT WHAT WILL HAPPEN
TO NEGATIVE VALUES?

*CHANGED TO POSITIVE
RIGHT. IF 'NUM' IS POSITIVE OR ZERO, NOTHING WILL HAPPEN BUT IF
'NUM' IS NEGATIVE, IT WILL BE CONVERTED TO '-NUM' WHICH IS POSITIVE.
THIS IS COMMONLY KNOWN AS THE 'ABSOLUTE VALUE' FUNCTION.
```

Comments

This constructed-response question frame illustrates a meaningful and highly interactive tutorial dialogue. This is largely due to the lesson author's skill in constructing the anticipated answer set with use of KEYWORD in Group 3, combined with meaningful feedback statements in Group 4.

The student's first response, NOTHING, was unanticipated in Group 3, causing the -R: feedback to be executed from Group 4. The remaining student responses all contained key words which matched Group 3 answer tags. The key word POSITIVE dominated the key word CHANGED in determining the feedback for the last student response.

End of EXAMPLE A.1

EXAMPLE A.2--Lesson Printout

```
FRAME 12.50  (Q)

G2. TEXT.
   SEE ANYTHING WRONG WITH THE PROGRAM SO FAR?

G3. ANSWERS.
POS/NEG

G5. ACTIONS.
+F:O.K. ...WHAT'S WRONG WITH THE CURRENT INPUT? B:12.70
-B:12.60


FRAME 12.60  (Q)

G2. TEXT.
   LET'S COMPARE A PORTION OF THE DESIRED PROGRAM WITH WHAT
WE'VE INPUT SO FAR:\\
                                           CURRENT INPUT\
      DESIRED PROGRAM                    *1.00 INTEGER ENTRIES;
*1.00 INTEGER ENTRIES, I;                 *2.00 REAL NJM 20, TOTAL;
*2.00 REAL NUM 20, TOTAL;                 *3.00 TOTAL=0;
*3.00 TOTAL=0;                            *4.00 READ ENTRIES;
*4.00 READ ENTRIES;                       *5.00 I=0;
*5.00 I=0;                                *6.00
*6.00 XXXX. READ NUM[I];
\\   COMPARE THESE FOR A MINUTE... WHEN YOU FIND WHAT'S WRONG, TYPE
... GO ...TO CONTINUE ON.

G3. ANSWERS.
0 WAIT 60
1 KEYWORD ON
A YES
B GO
B NO

G4. ACTIONS.
A R:O.K. ...ANOTHER MINUTE.
'-R:\\NEED MORE TIME? (YES/NO)
B F:O.K. ...WHAT'S WRONG WITH THE CODE?
'-F:\\TIME'S UP NOW.  WHAT'S WRONG WITH YOUR CURRENT INPUT?
```

EXAMPLE A.2--Student Printout

```
   SEE ANYTHING WRONG WITH THE PROGRAM SO FAR?

*NO

   LET'S COMPARE A PORTION OF THE DESIRED PROGRAM WITH WHAT
WE'VE INPUT SO FAR:

      DESIRED PROGRAM                        CURRENT INPUT

*1.00 INTEGER ENTRIES, I;              *1.00 INTEGER ENTRIES;
*2.00 REAL NUM 20, TOTAL;              *2.00 REAL NUM 20, TOTAL;
*3.00 TOTAL=0;                         *3.00 TOTAL=0;
*4.00 READ ENTRIES;                    *4.00 READ ENTRIES;
*5.00 I=0;                             *5.00 I=0;
*6.00 XXXX. READ NUM[I];               *6.00

   COMPARE THESE FOR A MINUTE... WHEN YOU FIND WHAT'S WRONG, TYPE
... GO ...TO CONTINUE ON.

*

NEED MORE TIME? (YES/NO)

*YES
O.K. ...ANOTHER MINUTE.

*GO
O.K. ...WHAT'S WRONG WITH THE CODE?
```

Comments

An earlier frame simulated some coding inputs to TINT, where the first line contained a missing item declaration. Frame 12.5∅ then asks a constructed-response question. The student didn't catch the error initially and answered NO to the question. Had the answer been YES, the student would have immediately branched to Frame 12.7∅ (next page). In this case, however, he receives a remedial frame where he is allowed to make a comparison to find the error. Note the use of ∅ WAIT 6∅ (seconds) in Group 3 of Frame 12.6∅ and the result in the student's printout after one elapsed period of 60 seconds. Had the student not typed GO during the second one-minute wait interval, he would have received TIME'S UP NOW .... and been branched automatically to the next frame.

EXAMPLE A.2--Lesson (cont'd)

```
FRAME 12.70  (Q)

G3. ANSWERS.
1 FORMULAS OFF
1 KEYWORD OFF
A+ITEM I
A+I
2 KEYWORD ON
A+ITEM MISSING
A+I MISSING
A+ITEM I
B SIX
B XXXX
B STATEMENT
C MISSING
D NOT DECLARED
D NO DECLARATION
2 FORMULAS ON
B 6 MISSING
B 6

G4. ACTIONS.
A F:RIGHT, WE FORGOT TO DECLARE ITEM 'I' IN THE CURRENT INPUT.
  F:NOW THAT YOU'VE FOUND THE ERROR, WE'LL CORRECT IT LATER WHEN
  F:WE EDIT THE PROGRAM.  LET'S CONTINUE NOW.....
B F:LINE 6.00, CONTAINING THE STATEMENT LABEL (XXXX.) AND THE 'READ'
  F:STATEMENT, HASN'T BEEN INPUT YET.  BUT WHAT'S MISSING FROM WHAT
  R:WE'VE ALREADY TYPED AND INPUT?  TRY AGAIN.
C R:WHAT'S MISSING FROM THE CURRENT INPUT?
D R:WHAT'S NOT DECLARED IN THE CURRENT INPUT?
-R:COMPARE LINE 1.00 OF BOTH SETS OF CODE AND TRY AGAIN.
-F:ON LINE 1.00 OF THE CURRENT INPUT, WE FORGOT TO DECLARE 'I' AS
 F:AN INTEGER ITEM.  WE'LL CORRECT IT LATER USING THE '?INSERT'
 F:COMMAND.  LET'S GO ON NOW.....
```

EXAMPLE A.2--Student (cont'd)

```
*WELL, LINE SIX WASN'T TYPED
LINE 6.ØØ, CONTAINING THE STATEMENT LABEL (XXXX.) AND THE 'READ'
STATEMENT, HASN'T BEEN INPUT YET.  BUT WHAT'S MISSING FROM WHAT
WE'VE ALREADY TYPED AND INPUT?  TRY AGAIN.

*ON LINE 1, I IS MISSING
RIGHT, WE FORGOT TO DECLARE ITEM 'I' IN THE CURRENT INPUT.
NOW THAT YOU'VE FOUND THE ERROR, WE'LL CORRECT IT LATER WHEN
WE EDIT THE PROGRAM.  LET'S CONTINUE NOW.....
```

Comments

Frame 13.7Ø awaits the student's answer as to the nature of the program error. As can be seen from the student's printout, the frame provides for considerable sensitivity to key features of the student's response. Again, this was accomplished by the use of KEYWORD and FORMULAS functions in Group 3, combined with relevant tutorial feedback in Group 4 of the frame.

An attempt is made to match the student's answer with KEYWORD and FORMULAS operating. For example, if the student had typed: LINE 6.ØØ ISN'T THERE, the answer would have been matched against the last anticipated answer in Group 3 (B 6) because with KEYWORD ON, 6 becomes sufficient for a match to occur and with FORMULAS ON, 6.ØØ and 6 are equated. On the other hand, had the student replied I SEEM TO BE MISSING THE POINT or DO YOU MEAN WHAT'S MISSING?, this would have matched answer tag C (MISSING) and the resultant feedback of WHAT'S MISSING FROM THE CURRENT INPUT? wouldn't be very appropriate. This indicates that in using KEYWORD with anticipated answer sets, there is some risk that the prescribed action may not be suited to all student responses in which the "keyword" is imbedded.

End of EXAMPLE A.2

EXAMPLE B

EXPOSITORY MODE

Providing an explanation and example of the TINT ?PRINT command, from TOPIC 14.

EXAMPLE B--Lesson Printout

```
FRAME 62.50  (M) LABEL= CUMBAK

G2. TEXT.
   YOU WERE ASKING ABOUT TINT COMMANDS.  WANT ANOTHER.....

G3. ANSWERS.
A+  EXPLANATION
B+  EXAMPLE
C   NO MORE FOR NOW

G4. ACTIONS.
A C:SET MARKER=1 F:ENTER YOUR COMMAND CHOICE. B:WHCMD
B C:SET MARKER=2 F:ENTER YOUR COMMAND CHOICE. B:WHCMD
C F:O.K., WE'LL GO BACK TO THE PROGRAMMING LESSON NOW... B:
```

EXAMPLE B--Student Printout

```
   YOU WERE ASKING ABOUT TINT COMMANDS.  WANT ANOTHER.....

A   EXPLANATION
B   EXAMPLE
C   NO MORE FOR NOW

*A
ENTER YOUR COMMAND CHOICE.
```

Comments

Assume that a student had been looking at TINT command explanations in Topic 14. He finally elected to enter Topic 15. Shortly after starting the programming exercises in Topic 15, he decides he needs an explanation of the ?PRINT command forms. He types ←EXPLAIN and re-enters Topic 14 at frame 62.5∅ labeled CUMBAK. (Refer to the discussion of ←EXPLAIN and ←REENTRY functions, part III, C for further details.)

The student chose "A" (Explanation) as his answer to the multiple-choice question. Had he chosen "C", he would have been sent back to Topic 15.

Note that this frame ended with ENTER YOUR COMMAND CHOICE. The student's entry is evaluated by the frame labeled WHCMD (next page).

EXAMPLE B--Lesson (cont'd)

```
FRAME 65.00  (Q) LABEL= WHCMD

G3. ANSWERS.
0 WAIT 20
1+6 WITHIN(5)
A YES
B NO
C LIST
2 KEYWORD ON
D START
D ST
E CONTINUE
E CO
F EXECUTE
F EX
G PRINT
G PR
H INSERT
H IN
I DELETE
I DE
J HALT
J H
K SAVE
K SA
L LOAD
L LO
M TRACE
M TR
N RENUMBER
N RE
O READY
G4. ACTIONS.
'-F:TYPE A NUMBER OR THE COMMAND NAME FROM THE LIST.  TYPE... LIST
  R:TO SEE THE LIST OF COMMANDS AGAIN.
A R:ENTER YOUR COMMAND CHOICE PLEASE.
1 C:SET HOLD=RESPONSE
D C:SET HOLD=1
E C:SET HOLD=2
F C:SET HOLD=3
G C:SET HOLD=4
H C:SET HOLD=5
I C:SET HOLD=6
J C:SET HOLD=7
K C:SET HOLD=8
L C:SET HOLD=9
M C:SET HOLD=10
N C:SET HOLD=11
O C:SET HOLD=12
B F:O.K., WE'LL GO BACK TO THE PROGRAMMING LESSON NOW...STANDBY. B:
C F:O.K., HERE IT IS.....  C:SET HOLD=0  B:LISTCOM
```

EXAMPLE B--Student (cont'd)

```
*TYPE A NUMBER OR THE COMMAND NAME FROM THE LIST.  TYPE... LIST
TO SEE THE LIST OF COMMANDS AGAIN.

*?PRINT
```

Comments

PLANIT waited for the student to specify his command choice. In this case, the waiting period was 2ø seconds because Group 3 of Frame 65.øø (line 1) has the instruction ø WAIT 2ø. Comparing the student printout with Group 4 of Frame 65.øø, we see that after 20 seconds had elapsed the feedback message preceded by a (') symbol was executed by PLANIT as a prompt to the student.

The student then typed ?PRINT, the correct word form of the TINT command for which an explanation was desired. If the student had typed LIST, he would have looped back to the immediately preceding frame labeled LISTCOM, which would have presented a list of the 11 TINT commands, each command preceded by a unique numerical tag of 1-11. Now note that in Group 3 of Frame 65.øø legal word forms and short forms of TINT commands would be matched, as would numerical entries of 1 through 11. The expression "6 WITHIN(5)" anticipates any student numerical entry from 1 through 11. The WITHIN(5) sets up the bounds for a range of anticipated numerical answers; 6+5=11 and 6-5=1. The advantage to the lesson author, in terms of economy of expression, should be obvious.

In Group 4, note that if the response is numerical, a CALC item HOLD is set equal to the value of RESPONSE (a PLANIT primitive which always contains the most recent numerical response from the teletype). If instead, the response is in word form or short form, HOLD will be set to a unique numerical value as a function of the student's response. Because our student responded ?PRINT, which bears a 'G' tag in Group 3, HOLD was set to 4 as prescribed by Group 4. With KEYWORD ON, the fact that the student preceded PRINT with a ? was irrelevant for answer-matching.

EXAMPLE B--Lesson (cont'd)

```
FRAME 66.00  (Q) LABEL= TODIRT

G4. ACTIONS.
B:FINDIT(HOLD,MARKER)


FRAME 70.00  (Q) LABEL= PRINT1

G2. TEXT.
   THE TINT COMMAND '?PRINT' (OR ?PR) IS USED TO LIST ONE OR MORE
LINES OF PROGRAM CODE ON THE TELETYPE.  IT MAY ALSO BE USED TO
LIST THE CONTENTS OF A SPECIFIC ITEM OR ARRAY, OR OF ALL ITEMS
AND ARRAYS.\
   TO PRINT YOUR ENTIRE PROGRAM, TYPE:\\?PRINT ALL\
   TO PRINT LINE 7.00 FROM YOUR PROGRAM, USE ANY OF THESE:\
?PRINT 7\?PRINT 7.0\?PRINT 7.00          ...OR...
\?PRINT LASTLINE    ...IF 7.00 IS THE LAST LINE IN YOUR PROGRAM.\
   IF YOU WANTED TO PRINT LINES 1.50 THROUGH 6.00, YOU COULD TYPE:\
?PRINT 1.5 6\?PRINT 1.50 6.0    ...ETC.  OR, IF LINE 6.00 IS THE LAST =
                                         LINE OF YOUR PROGRAM:\
?PRINT 1.5 ALL\?PRINT 1.50 ALL    ...WOULD ALSO LIST LINES 1.50 =
THROUGH 6.00.\
   TO PRINT THE CONTENTS OF ALL ITEMS AND ARRAYS IN YOUR PROGRAM, TYPE:
\?PRINT RESULTS\
   TO PRINT THE CONTENTS OF A SPECIFIC ITEM OR ARRAY, SUBSTITUTE THE
NAME OF THE ITEM OR ARRAY (E.G., TOTAL, I, NUM) IN PLACE OF THE WORD
'RESULTS'.  TINT WILL NOT PRINT ZERO AND BLANK VALUES.

G4. ACTIONS.
B:WUDYU


FRAME 90.00  (Q) LABEL= WUDYU

G2. TEXT.
   WOULD YOU LIKE TO SEE AN EXAMPLE OF THAT COMMAND USED WITH
A SAMPLE PROGRAM?

G3. ANSWERS.
POS/NEG

G5. ACTIONS.
+B:FINDIT(HOLD,2)
-B:NXT
```

EXAMPLE B--Student (cont'd)

```
   THE TINT COMMAND '?PRINT' (OR ?PR) IS USED TO LIST ONE OR MORE
LINES OF PROGRAM CODE ON THE TELETYPE.  IT MAY ALSO BE USED TO
LIST THE CONTENTS OF A SPECIFIC ITEM OR ARRAY, OR OF ALL ITEMS
AND ARRAYS.

   TO PRINT YOUR ENTIRE PROGRAM, TYPE:

?PRINT ALL

   TO PRINT LINE 7.00 FROM YOUR PROGRAM, USE ANY OF THESE:

?PRINT 7
?PRINT 7.0
?PRINT 7.00          ...OR...

?PRINT LASTLINE    ...IF 7.00 IS THE LAST LINE IN YOUR PROGRAM.

   IF YOU WANTED TO PRINT LINES 1.50 THROUGH 6.00, YOU COULD TYPE:

?PRINT 1.5 6
?PRINT 1.50 6.0    ...ETC.  OR, IF LINE 6.00 IS THE LAST
LINE OF YOUR PROGRAM:

?PRINT 1.5 ALL
?PRINT 1.50 ALL    ...WOULD ALSO LIST LINES 1.50 THROUGH 6.00.

   TO PRINT THE CONTENTS OF ALL ITEMS AND ARRAYS IN YOUR PROGRAM, TYPE:

?PRINT RESULTS

   TO PRINT THE CONTENTS OF A SPECIFIC ITEM OR ARRAY, SUBSTITUTE THE
NAME OF THE ITEM OR ARRAY (E.G., TOTAL, I, NUM) IN PLACE OF THE WORD
'RESULTS'.  TINT WILL NOT PRINT ZERO AND BLANK VALUES.

   WOULD YOU LIKE TO SEE AN EXAMPLE OF THAT COMMAND USED WITH
A SAMPLE PROGRAM?

*YES
```

Comments

Now Frame 66.ØØ instructs PLANIT to branch to the frame designated by matrix FINDIT, entered with arguments HOLD (rows) and MARKER (columns). Earlier in Topic 14, matrix FINDIT was defined with 11 rows and 2 columns. It was filled with frame numbers such that all explanations are denoted by one 11 x 1 array in the matrix, and all examples by another 11 x 1 array. MARKER was set to 1 when the student chose "Explanation" (Frame 62.5Ø) after re-entering Topic 14. HOLD was set to 4 by Frame 65.ØØ (prior page). Thus, the appropriate command explanation is accessed by PLANIT.

The student next receives an expository explanation on uses of the TINT ?PRINT command. After that, he is asked if he would like to see an example of that command. Because the student answers YES, Group 4 of Frame 9Ø.ØØ instructs PLANIT to enter matrix FINDIT again, with arguments of HOLD (previously set to 4 indicating ?PRINT) and 2 (indicating an "example").

EXAMPLE B--Lesson (cont'd)

```
FRAME 82.00  (Q) LABEL= PRINT2

G2. TEXT.
   LET'S USE THE '?PRINT' COMMAND ON THE FOLLOWING PROGRAM:\
?PRINT ALL              (YOU WANT TO LIST THE ENTIRE PROGRAM)\
*  1.00 INTEGER ENTRIES, I;\*  2.00 REAL NUM 20, TOTAL;
*  3.00 TOTAL=0;\*  4.00 READ ENTRIES;\*  5.00 I=0;
*  6.00 XXXX. READ NUM[I];\*  7.00 TOTAL=TOTAL+NUM[I];
*  8.00 I=I+1;\*  9.00 IF I LS ENTRIES;\* 10.00 GOTO XXXX;
* 11.00 PRINT TOTAL;\*PRINT COMPLETE\\*ENTER COMMAND
?PRINT 2.0              (TO PRINT LINE 2.00)
\*  2.00 REAL NUM 20, TOTAL;\*PRINT COMPLETE\\*ENTER COMMAND
?PRINT LASTLINE         (OR...?PRINT 11 WOULD ALSO PRINT LINE 11.00)
\* 11.00 PRINT TOTAL;\*PRINT COMPLETE\\*ENTER COMMAND
?PRINT 1 3              (TO PRINT LINES 1.00, 2.00 AND 3.00)
\*  1.00 INTEGER ENTRIES, I;\*  2.00 REAL NUM 20, TOTAL;
*  3.00 TOTAL=0;\*PRINT COMPLETE\\*ENTER COMMAND
?PRINT 9.0 ALL          (TO PRINT LINES 9.00, 10.00 AND 11.00)
\*  9.00 IF I LS ENTRIES;\* 10.00 GOTO XXXX;\* 11.00 PRINT TOTAL;
*PRINT COMPLETE\\*ENTER COMMAND

G4. ACTIONS.
B:ANOTHR
```

EXAMPLE B--Student (cont'd)

```
      LET'S USE THE '?PRINT' COMMAND ON THE FOLLOWING PROGRAM:

   ?PRINT ALL            (YOU WANT TO LIST THE ENTIRE PROGRAM)

   *  1.00 INTEGER ENTRIES, I;
   *  2.00 REAL NUM 20, TOTAL;
   *  3.00 TOTAL=0;
   *  4.00 READ ENTRIES;
   *  5.00 I=0;
   *  6.00 XXXX. READ NUM[I];
   *  7.00 TOTAL=TOTAL+NUM[I];
   *  8.00 I=I+1;
   *  9.00 IF I LS ENTRIES;
   * 10.00 GOTO XXXX;
   * 11.00 PRINT TOTAL;
   *PRINT COMPLETE

   *ENTER COMMAND
   ?PRINT 2.0            (TO PRINT LINE 2.00)

   *  2.00 REAL NUM 20, TOTAL;
   *PRINT COMPLETE

   *ENTER COMMAND
   ?PRINT LASTLINE       (OR...?PRINT 11 WOULD ALSO PRINT LINE 11.00)

   * 11.00 PRINT TOTAL;
   *PRINT COMPLETE

   *ENTER COMMAND
   ?PRINT 1 3            (TO PRINT LINES 1.00, 2.00 AND 3.00)

   *  1.00 INTEGER ENTRIES, I;
   *  2.00 REAL NUM 20, TOTAL;
   *  3.00 TOTAL=0;
   *PRINT COMPLETE

   *ENTER COMMAND
   ?PRINT 9.0 ALL        (TO PRINT LINES 9.00, 10.00 AND 11.00)

   *  9.00 IF I LS ENTRIES;
   * 10.00 GOTO XXXX;
   * 11.00 PRINT TOTAL;
   *PRINT COMPLETE

   *ENTER COMMAND
```

Comments

This time, matrix FINDIT brings forth Frame 82.ØØ, which provides the student with an expository simulation of both TINT inputs and outputs. The simulation serves as the requested "example" of the use of TINT ?PRINT command forms. The simulated command inputs are annotated to indicate what each command form accomplishes in the example.

End of EXAMPLE B

EXAMPLE C

EXERCISE MODE

EXAMPLE C.1--Analysis and synthesis questions from Topic 13.

EXAMPLE C.2--TINT communication exercises that simulate TINT, from Topic 14.

EXAMPLE C.3--Exercise in writing a program by direct interaction with TINT, from Topic 15.

EXAMPLE C.1--Lesson Printout

```
FRAME 197.ØØ  (Q)

G2. TEXT.
   NOW WE ARE GOING TO MODIFY PROGRAM ''B'' SO THAT IT WILL COMPUTE
THE MEAN, VARIANCE AND STANDARD DEVIATION FOR AS MANY AS FIFTY SCORES.
I'LL SHOW YOU THE PROGRAM BUT SOME KEY STATEMENTS WILL BE INCOMPLETE.
THERE WILL ALSO BE SOME LINE NUMBERS ADDED THIS TIME SO THAT I CAN
SPECIFY THE STATEMENT MORE EASILY.  HERE IS THE PROGRAM:


FRAME 198.ØØ  (Q)

G2. TEXT.
1)    REAL NUM 5Ø, SUM, SUMSQ, MEAN, VARIANCE, STDDEV;
2)    INTEGER ENTRIES, I;
3)    SUM=     ;
4)    SUMSQ=   ;
5)    I=       ;
6)    READ ENTRIES;
7)    LOOP1. READ NUM[I];
8)    SUM=SUM+NUM[I];
9)    I=I+1;
1Ø)   IF I LS ENTRIES;
11)                      ;
12)   MEAN=SUM/ENTRIES;
13)         =Ø;
14)   LOOP2. SUMSQ=SUMSQ+ (    ?    ) **2
15)   I=I+1;
16)   IF I LS ENTRIES;
17)                      ;
18)   VARIANCE=SUMSQ/ENTRIES - MEAN**2;
19)   STDDEV=SQRT(VARIANCE);
2Ø)   PRINT MEAN;
21)   PRINT VARIANCE;
22)   PRINT STDDEV;
```

EXAMPLE C.1--Student Printout

```
     NOW WE ARE GOING TO MODIFY PROGRAM ''B'' SO THAT IT WILL COMPUTE
THE MEAN, VARIANCE AND STANDARD DEVIATION FOR AS MANY AS FIFTY SCORES.
I'LL SHOW YOU THE PROGRAM BUT SOME KEY STATEMENTS WILL BE INCOMPLETE.
THERE WILL ALSO BE SOME LINE NUMBERS ADDED THIS TIME SO THAT I CAN
SPECIFY THE STATEMENT MORE EASILY.  HERE IS THE PROGRAM:

1)    REAL NUM 50, SUM, SUMSQ, MEAN, VARIANCE, STDDEV;
2)    INTEGER ENTRIES, I;
3)    SUM=    ;
4)    SUMSQ=   ;
5)    I=       ;
6)    READ ENTRIES;
7)    LOOP1. READ NUM[I];
8)    SUM=SUM+NUM[I];
9)    I=I+1;
10)   IF I LS ENTRIES;
11)                       ;
12)   MEAN=SUM/ENTRIES;
13)           =0;
14)   LOOP2. SUMSQ=SUMSQ+ (   ?   ) **2
15)   I=I+1;
16)   IF I LS ENTRIES;
17)                       ;
18)   VARIANCE=SUMSQ/ENTRIES - MEAN**2;
19)   STDDEV=SQRT(VARIANCE);
20)   PRINT MEAN;
21)   PRINT VARIANCE;
22)   PRINT STDDEV;
```

Comments

The problem for the student is to analyze the operation of this sample program and to synthesize many concepts and rules learned in prior topics in order to correctly fill in the missing entries. A series of constructed-response questions follows.

EXAMPLE C.1--Lesson (cont'd)

```
FRAME 199.00  (9)

G2. TEXT.
   NOW, LET'S FIX UP THE OMISSIONS.  STATEMENTS 3, 4 AND 5 ALL HAVE
THE RIGHTHAND EXPRESSION OMITTED.  IT SHOULD BE THE SAME FOR ALL THREE.
WHAT IS MISSING?

G3. ANSWERS.
1 + Ø
A  ZERO
2 KEYWORD ON
B  I KNOW
C  NUM[I]
C  NUM

G4. ACTIONS.
C:KEYWORD OFF
1 F:
A F:NO, THE ITEM NAME 'ZERO' IS UNDEFINED.  IF 'ZERO' HAD BEEN DECLARED
  R:AND HAD BEEN ASSIGNED A VALUE, THEN IT WOULD BE OK.  TRY AGAIN.
B R:FROM WHAT VALUE SHOULD THESE THREE ITEMS ACCUMULATE?
3 F:THE ITEMS IN EACH OF THE THREE STATEMENTS (3, 4 AND 5) ACCUMULATE.
  R:FROM WHAT VALUE SHOULD THEY START ACCUMULATING?
C F:NO, I COULD SEE YOUR MISTAKE FOR STATEMENT 3 BUT IT CERTAINLY
  F:WOULDN'T FIT THE NEXT TWO.  I'M ASKING FOR THE STARTING VALUE
  F:FOR ALL THREE ITEMS.  =R:TRY AGAIN.
- R:NO, I THINK YOU ARE GUESSING.  IF YOU DON'T KNOW, ADMIT IT.
B F:EACH SHOULD BE ASSIGNED THE VALUE, Ø.  THE 'SUM', ETC., ACCUMULATE
  F:FROM THERE.
```

EXAMPLE C.1--Student (cont'd)

```
   NOW, LET'S FIX UP THE OMISSIONS.  STATEMENTS 3, 4 AND 5 ALL HAVE
THE RIGHTHAND EXPRESSION OMITTED.  IT SHOULD BE THE SAME FOR ALL THREE.
WHAT IS MISSING?

*ZERO
NO, THE ITEM NAME 'ZERO' IS UNDEFINED.  IF 'ZERO' HAD BEEN DECLARED
AND HAD BEEN ASSIGNED A VALUE, THEN IT WOULD BE OK.  TRY AGAIN.

*Ø
FINE.
```

Comments

Here is an interesting case where Group 3 of the frame is used to detect two seemingly equivalent responses, ZERO and Ø. But this time the reason is not to count a verbal answer equivalent to a numerical one. Rather, it is to differentiate the verbal response from the numerical one, so that appropriate remedial feedback can be provided as required.

EXAMPLE C.1--Lesson (cont'd)

```
FRAME 200.00   (9)

G2. TEXT.
THOSE THREE STATEMENTS SHOULD READ:
\3)    SUM=0;\4)    SUMSQ=0;\5)    I=0;
\AT LINE 11, THE ENTIRE STATEMENT IS MISSING.  WHAT SHOULD THAT BE?

G3. ANSWERS.
0 KEYWORD ON
A+ GOTO LOOP1
A+ GOTO LOOP1;
B   GO TO
C   LOOP2
C   LOOP2;
D   LOOP 1
D   LOOP 1;
E   GOTO
F   I KNOW

G4. ACTIONS.
A F:VERY GOOD.
B F:DON'T SEPARATE THE 'GO' AND 'TO' FOR TINT.  WRITE, 'GOTO'.
  R:TRY AGAIN.
C F:WITH THAT, THE 'READ NUM[I];' STATEMENT WOULD EXECUTE ONLY ONCE.
  F:THE PROGRAM MUST REACH THE 'READ NUM[I];' STATEMENT ONCE FOR EACH
  R:ENTRY.  TRY AGAIN.
D F:YOU INCLUDED A SPACE WHERE IT DIDN'T BELONG.  ''LOOP1'' IS ALL ONE
  R:WORD.  TRY AGAIN.
E F:''GOTO'' IS RIGHT BUT I DIDN'T RECOGNIZE THE REMAINDER OF THE
  R:STATEMENT.  FOLLOW THE ''GOTO'' WITH THE CORRECT LABEL.
F F:THE IF-STATEMENT SHOULD BE YOUR CLUE.  ASK YOURSELF WHY THE IF-
  F:STATEMENT IS THERE AND WHAT THE PROGRAM SHOULD DO IF THAT STATEMENT
  R:IS TRUE.  TRY AGAIN.
- F:NO, NOTICE THAT THIS STATEMENT FOLLOWS THE IF-STATEMENT.
  R:TRY AGAIN.
-F F:THIS WILL BE A ''GOTO'' STATEMENT.  TRY AGAIN=R:.
-F F: C:
```

EXAMPLE C.1--Student (cont'd)

```
THOSE THREE STATEMENTS SHOULD READ:

3)    SUM=0;
4)    SUMSQ=0;
5)    I=0;

AT LINE 11, THE ENTIRE STATEMENT IS MISSING.  WHAT SHOULD THAT BE?

*I DON'T KNOW
THE IF-STATEMENT SHOULD BE YOUR CLUE.  ASK YOURSELF WHY THE IF-
STATEMENT IS THERE AND WHAT THE PROGRAM SHOULD DO IF THAT STATEMENT
IS TRUE.  TRY AGAIN.

*IS IT GOTO LOOP2 ?
WITH THAT, THE 'READ NUM[I];' STATEMENT WOULD EXECUTE ONLY ONCE.
THE PROGRAM MUST REACH THE  READ NUM[I];' STATEMENT ONCE FOR EACH
ENTRY.  TRY AGAIN.

*GO TO LOOP1;
DON'T SEPARATE THE 'GO' AND 'TO' FOR TINT.  WRITE, 'GOTO'.
TRY AGAIN.

*GOTO LOOP1;
VERY GOOD.
```

Comments

Practice continues in the form of a tutorial constructed-response exercise. Note the usefulness of the KEYWORD function as combined with the anticipated answer set in Group 3. The student's first response brings forth a hint from Group 4 of Frame 2ØØ. The LOOP2 portion of the student's next response was anticipated by the lesson author in Group 3 and, with KEYWORD ON, the IS IT and ? were ignored. The student's third response was matched on GO TO, but is still incorrect. The student finally enters the correct GOTO statement.

EXAMPLE C.1--Lesson (cont'd)

```
FRAME 201.00  (M)

G2. TEXT.
AT LINE 13, WHICH ITEM NAME MUST HAVE THE VALUE, 0, ASSIGNED TO IT
AGAIN?  (CHOOSE A LETTER).

G3. ANSWERS.
  A.  NUM
  B.  SUM
  C.  SUMSQ
  D.  MEAN
  E.  VARIANCE
  F.  STDDEV
  G.  ENTRIES
 +H.  I

G4. ACTIONS.
A R:NO, WE ARE PRESERVING THOSE VALUES.  TRY AGAIN.
B R:UNNECESSARY.  'SUM' IS NOT USED ANYMORE.  TRY AGAIN.
C R:UNNECESSARY.  IT MUST STILL BE ZERO FROM LINE 4.  TRY AGAIN.
D R:NO.  WE WANT TO SAVE THE VALUE IN 'MEAN'.  TRY AGAIN.
EF F:UNNECESSARY.  THAT ITEM GETS A VALUE ASSIGNED BELOW LINE 17.
   R:TRY AGAIN.
G F:NO.  WE NEED TO PRESERVE THE VALUE IN 'ENTRIES' FOR THE STATEMENTS
   R:BELOW.  TRY AGAIN.
H F:

FRAME 202.00  (Q)

G2. TEXT.
WHAT SHOULD APPEAR IN PLACE OF THE (    ?    ) IN LINE 14?

G3. ANSWERS.
A+ NUM[I]
B  NUM
1 KEYWORD ON
C  I KNOW
D A
D B
D C

G4. ACTIONS.
C:KEYWORD OFF
A F:
B R:YES, BUT IT MUST BE SUBSCRIPTED (I.E.  NUM[?] ).  TRY AGAIN.
C F:THIS IS TO BE THE SUM OF THE SQUARED DATA ENTRIES.  THE  **2
  R:REPRESENTS THE SQUARING.  WHAT ITEM NAME REPRESENTS THE DATA?
- F:NO, THAT STATEMENT SUMS THE SQUARE OF EACH DATA ENTRY.  THE **2 IS
  F:THE SYMBOL FOR SQUARING.  WHAT ITEM NAME HOLDS THE DATA TO BE
  R:SQUARED?
- R:NO, TRY ONCE MORE.
-C C:
D R:TYPE THE ANSWER, NOT THE LETTER CHOICE.
```

EXAMPLE C.1--Student (cont'd)

```
AT LINE 13, WHICH ITEM NAME MUST HAVE THE VALUE, 0, ASSIGNED TO IT
AGAIN?  (CHOOSE A LETTER).

   A.   NUM
   B.   SUM
   C.   SUMSQ
   D.   MEAN
   E.   VARIANCE
   F.   STDDEV
   G.   ENTRIES
   H.   I

*B
UNNECESSARY.  'SUM' IS NOT USED ANYMORE.  TRY AGAIN.

*I

CHOOSE ONE OF THE ABOVE LETTERS.
*H
FINE.


WHAT SHOULD APPEAR IN PLACE OF THE (   ?   ) IN LINE 14?

*NUM
YES, BUT IT MUST BE SUBSCRIPTED (I.E.  NUM[?] ).  TRY AGAIN.

*NUM[I]
TRUE.
```

Comments

Of interest in these frames is the author's attempt to anticipate student responses and provide useful feedback. For example, in Frame 2ø2.øø the author has even anticipated that the student may try to enter a multiple-choice tag from the prior frame, rather than entering a constructed response (Group 3--D A, D B, D C). Our sample student didn't try this, but did receive feedback appropriate to his answers.

EXAMPLE C.1--Lesson (cont'd)

```
FRAME 203.ØØ  (Ø)

G2. TEXT.
WHAT STATEMENT BELONGS IN LINE 17?

G3. ANSWERS.
Ø KEYWORD ON
A+ GOTO LOOP2
A+ GOTO LOOP2;
B  GO TO
C  LOOP1
C  LOOP1;
D  LOOP 2
D  LOOP 2;
E  GOTO
F  I KNOW

G4. ACTIONS.
A F:VERY GOOD.
B F:DON'T SEPARATE THE 'GO' AND 'TO' FOR TINT.  WRITE, 'GOTO'.
  R:TRY AGAIN.
C F:THAT WILL MAKE THE PROGRAM GO BACK INTO THE FIRST LOOP AGAIN WHICH,
  F:IF YOU WILL FOLLOW THE LOGIC, WILL SET UP A NON-TERMINATING LOOP.
  R:WE DON'T WANT THAT.  TRY AGAIN.
D F:YOU INCLUDED A SPACE WHERE IT DIDN'T BELONG.  ''LOOP2'' IS ALL ONE
  R:WORD.  TRY AGAIN.
E F:''GOTO'' IS RIGHT BUT I DIDN'T RECOGNIZE THE REMAINDER OF THE
  R:STATEMENT.  FOLLOW THE ''GOTO'' WITH THE CORRECT LABEL.
F F:THE IF-STATEMENT SHOULD BE YOUR CLUE.  ASK YOURSELF WHY THE IF-
  F:STATEMENT IS THERE AND WHAT THE PROGRAM SHOULD DO IF THAT STATEMENT
  R:IS TRUE.  TRY AGAIN.
- F:NO, NOTICE THAT THIS STATEMENT FOLLOWS THE IF-STATEMENT.
  R:TRY AGAIN.
-F F:THIS WILL BE A ''GOTO'' STATEMENT.  TRY AGAIN=R:.
-F F: C:


FRAME 204.ØØ  (Ø)

G2. TEXT.
   NOW I WILL PRINT THE COMPLETE PROGRAM FOR YOU:=


FRAME 205.ØØ  (Ø)

G2. TEXT.
1)    REAL NUM 5Ø, SUM, SUMSQ, MEAN, VARIANCE, STDDEV;
2)    INTEGER ENTRIES, I;
3)    SUM=Ø;
4)    SUMSQ=Ø;
5)    I=Ø;
6)    READ ENTRIES;
7)    LOOP1. READ NUM[I];
8)    SUM=SUM+NUM[I];
9)    I=I+1;
1Ø)   IF I LS ENTRIES;
11)   GOTO LOOP1;
12)   MEAN=SUM/ENTRIES;
13)   I=Ø;
14)   LOOP2. SUMSQ=SUMSQ+NUM[I]**2;
15)   I=I+1;
16)   IF I LS ENTRIES;
17)   GOTO LOOP2;
18)   VARIANCE=SUMSQ/ENTRIES - MEAN**2;
19)   STDDEV=SQRT(VARIANCE);
2Ø)   PRINT MEAN;
21)   PRINT VARIANCE;
22)   PRINT STDDEV;
```

EXAMPLE C.1--Student (cont'd)

```
WHAT STATEMENT BELONGS IN LINE 17?

*GOTO LOOP2
VERY GOOD.

   NOW I WILL PRINT THE COMPLETE PROGRAM FOR YOU:

1)    REAL NUM 50, SUM, SUMSQ, MEAN, VARIANCE, STDDEV;
2)    INTEGER ENTRIES, I;
3)    SUM=0;
4)    SUMSQ=0;
5)    I=0;
6)    READ ENTRIES;
7)    LOOP1. READ NUM[I];
8)    SUM=SUM+NUM[I];
9)    I=I+1;
10)   IF I LS ENTRIES;
11)   GOTO LOOP1;
12)   MEAN=SUM/ENTRIES;
13)   I=0;
14)   LOOP2. SUMSQ=SUMSQ+NUM[I]**2;
15)   I=I+1;
16)   IF I LS ENTRIES;
17)   GOTO LOOP2;
18)   VARIANCE=SUMSQ/ENTRIES - MEAN**2;
19)   STDDEV=SQRT(VARIANCE);
20)   PRINT MEAN;
21)   PRINT VARIANCE;
22)   PRINT STDDEV;
```

Comments

The student completes the exercise on analysis and synthesis of concepts and receives a summary of what he has accomplished.

End of EXAMPLE C.1

EXAMPLE C.2--Lesson Printout

```
         FRAME 16.25  (0)

         G2. TEXT.
             NOW YOU TRY TO CONDUCT THE SEARCH THROUGH THE 5 LINES OF YOUR
         PROGRAM.  USE EITHER THE '?PRINT X Y' OR THE '?PRINT X' FORMS OF
         THE '?PRINT' COMMAND.

      FRAME 16.30  (0)

      G2. TEXT.
      \*ENTER COMMAND\*? =

      G3. ANSWERS.
      A+PRINT 1 5
      B+PRINT 1 4
      C+PRINT 1 3
      D+PRINT 1 2
      E+PRINT 1
      F PRINT 2 5
      G PRINT 2 4
      H PRINT 2 3
      I PRINT 2
      J PRINT 3 5
      K PRINT 3 4
      L PRINT 3
      M PRINT 4 5
      N PRINT 4
      O PRINT 5
      P PRINT X Y
      Q PRINT X
      R PR
      0 KEYWORD ON

      G4. ACTIONS.
      A F:\*1.00 INTEGER ENTRIES;\*2.00 REAL NUM 20, TOTAL;\*3.00 TOTAL=0;
        F:*4.00 READ ENTRIES;\*5.00 I=0;\*PRINT COMPLETE
      B F:*1.00 INTEGER ENTRIES;\*2.00 REAL NUM 20, TOTAL;\*3.00 TOTAL=0;
        F:*4.00 READ ENTRIES;\*PRINT COMPLETE
      C F:*1.00 INTEGER ENTRIES;\*2.00 REAL NUM 20, TOTAL;\*3.00 TOTAL=0;
        F:*PRINT COMPLETE
      D F:*1.00 INTEGER ENTRIES;\*2.00 REAL NUM 20, TOTAL;\*PRINT COMPLETE
      E F:*1.00 INTEGER ENTRIES;\*PRINT COMPLETE
      F F:*2.00 REAL NUM 20, TOTAL;\*3.00 TOTAL=0;\*4.00 READ ENTRIES;
        F:*5.00 I=0;\*PRINT COMPLETE
      G F:*2.00 REAL NUM 20, TOTAL;\*3.00 TOTAL=0;\*4.00 READ ENTRIES;
        F:*PRINT COMPLETE
      H F:*2.00 REAL NUM 20, TOTAL;\*3.00 TOTAL=0;\*PRINT COMPLETE
      I F:*2.00 REAL NUM 20, TOTAL;\*PRINT COMPLETE
      J F:*3.00 TOTAL=0;\*4.00 READ ENTRIES;\*5.00 I=0;\*PRINT COMPLETE
      K F:*3.00 TOTAL=0;\*4.00 READ ENTRIES;\*PRINT COMPLETE
      L F:*3.00 TOTAL=0;\*PRINT COMPLETE
      M F:*4.00 READ ENTRIES;\*5.00 I=0;\*PRINT COMPLETE
      N F:*4.00 READ ENTRIES;\*PRINT COMPLETE
      O F:*5.00 I=0;\*PRINT COMPLETE
      PQ F:THAT'S A GENERAL FORM OF THE COMMAND.  BUT SUBSTITUTE LINE
        F:NUMBERS FROM 1 THROUGH 5 TO SELECTIVELY PRINT LINES FROM
         R:YOUR CURRENT PROGRAM.\*? =
      -F:I DON'T RECOGNIZE THAT AS A PRINT COMMAND OF THE FORM '?PRINT X'
      F:OR '?PRINT X Y'.  TYPE:\\   'PRINT(SPACE)X(SPACE)Y'\\OR\
       F:   'PRINT(SPACE)X'\\FOR EXAMPLE:\
       F:   ?PRINT 1 3 (PRINT LINES 1 THROUGH 3)\   ?PRINT 3 (PRINT =
       R:LINE 3 ONLY)\\NOW TRY AGAIN.\*? =
      R F:YES, '?PR' IS A LEGAL TINT COMMAND FORM, BUT PLEASE USE '?PRINT'
        R:FOR THIS EXERCISE.\*? =
```

EXAMPLE C.2--Student Printout

```
   NOW YOU TRY TO CONDUCT THE SEARCH THROUGH THE 5 LINES OF YOUR
PROGRAM.  USE EITHER THE '?PRINT X Y' OR THE '?PRINT X' FORMS OF
THE '?PRINT' COMMAND.


*ENTER COMMAND
*? PRINT X Y
THAT'S A GENERAL FORM OF THE COMMAND.  BUT SUBSTITUTE LINE
NUMBERS FROM 1 THROUGH 5 TO SELECTIVELY PRINT LINES FROM
YOUR CURRENT PROGRAM.
*? PRINT 3 5
*3.00 TOTAL=0;
*4.00 READ ENTRIES;
*5.00 I=0;
*PRINT COMPLETE
```

<u>**Comments**</u>

The student begins an exercise in which he is to list selectively any one or a combination of the five lines of his program, in order to find which line is in error (missing item declaration). He may use either of two general forms of the TINT ?PRINT command. In response to the student's input of PRINT 3 5, the lesson simulates a TINT printout of lines 3.ØØ through 5.ØØ. Note that the student did not have to precede his command input with a ?, even though it would be required by TINT. This is because Group 2 of Frame 16.3Ø caused the ? symbol to be printed. Later in Topic 14 exercises, the lesson quits printing the ? and the student must begin to use it regularly as part of his command inputs, as he would have to for TINT. On the other hand, if the student had entered ?PRINT 3 5 instead of PRINT 3 5, it would have been accepted as correct because KEYWORD is turned ON in Group 3.

EXAMPLE C.2--Lesson (cont'd)

```
FRAME 16.40  (0)

G2. CRITERIA.
IF 16.3,+ F:\DID YOU FIND THE LINE YOU WERE LOOKING FOR? R:16.50
ELSE F:\FIND THE LINE YOU WERE LOOKING FOR? R:16.45


FRAME 16.45  (9)

G3. ANSWERS.
POS/NEG

G5. ACTIONS.
+F:OH? ... R:IS LINE 1.00 IN THE PRINTOUT YOU RECEIVED?
-F:THAT'S RIGHT, SO WE'LL GO BACK NOW AND LET YOU SEARCH FOR -
 F:LINE 1.00 AGAIN, USING THE '?PRINT' COMMAND. R:16.30
+F:HEY, ARE YOU TRYING TO KID ME?? ...LET'S GO BACK NOW AND YOU
 F:TRY THE '?PRINT' COMMAND AGAIN TO GET A LINE 1.00 PRINTOUT.
 R:16.30
-F:THAT'S RIGHT, SO WE'LL GO BACK NOW AND LET YOU SEARCH FOR
 F:LINE 1.00 AGAIN, USING THE '?PRINT COMMAND. R:16.30


FRAME 16.50  (9)

G3. ANSWERS.
1 KEYWORD OFF
4 YES
3 NO
2 KEYWORD ON
C*1
C*1.0
C*1.00

G4. ACTIONS.
A R:[illegible]... WHAT WAS THE LINE NUMBER?
C F:RIGHT... LET'S FIX LINE 1.00 NOW, USING THE '?INSERT'
  F:COMMAND. R:17.00
-A F:SURE YOU DID.  LOOK AT THE '?PRINT' COMMAND YOU USED, AND AT
   F:THE PRINTOUT OF LINE 1.00 THAT YOU RECEIVED.  NOTE THAT LINE
   F:1.00 DEFINES 'ENTRIES' AS AN 'INTEGER' ITEM, BUT DOES NOT
   F:INCLUDE ITEM 'I' IN THE DECLARATION.  THE LINE SHOULD START:\
   F:      * 1.00 INTEGER ENTRIES, I:\\DO YOU UNDERSTAND? .
```

EXAMPLE C.2--Student (cont'd)

```
FIND THE LINE YOU WERE LOOKING FOR?

*YES
OH? ...
IS LINE 1.ØØ IN THE PRINTOUT YOU RECEIVED?

*NO
THAT'S RIGHT, SO WE'LL GO BACK NOW AND LET YOU SEARCH FOR
LINE 1.ØØ AGAIN, USING THE '?PRINT' COMMAND.



*ENTER COMMAND
*? PRINT 1 4
*1.ØØ INTEGER ENTRIES;
*2.ØØ REAL NUM 2Ø, TOTAL;
*3.ØØ TOTAL=Ø;
*4.ØØ READ ENTRIES;
*PRINT COMPLETE

DID YOU FIND THE LINE YOU WERE LOOKING FOR?

*YES
GOOD... WHAT WAS THE LINE NUMBER?

*1.Ø
RIGHT... LET'S FIX LINE 1.ØØ NOW, USING THE '?INSERT'
COMMAND.
```

<u>Comments</u>

A decision (D) frame 16.4Ø next determines whether the student should see Frame 16.45 or Frame 16.5Ø, depending on his answer to Frame 16.3Ø (prior page). In this case, the student did not request a printout that contained line 1.ØØ (the line in error), so he sees Frame 16.45 as prescribed by the decision frame. Frame 16.45 then clarifies what line the student is looking for and branches him back to Frame 16.3Ø (prior page) to try the ?PRINT command again. This time the student is successful with a command of ?PRINT 1 4. The lesson simulates the appropriate TINT printout and the student indicates that he is now aware of the line that is in error.

End of EXAMPLE C.2

EXAMPLE C.3--Lesson Printout

```
FRAME 310.00  (Q)

G2. TEXT.
   SUPPOSE YOU WANTED TO FIND THE SUM OF THE 'PAIR PRODUCTS' OF TWO
LISTS OF NUMBERS (I.E. YOU WANT TO MULTIPLY EACH ENTRY IN THE FIRST
LIST BY ITS CORRESPONDING ENTRY IN THE SECOND LIST AND SUM THE
PRODUCTS). THE TWO LISTS WILL HAVE THE SAME NUMBER OF ENTRIES.


FRAME 311.00  (Q)

G2. TEXT.
DO YOU THINK YOU COULD WRITE SUCH A PROGRAM NOW?

G3. ANSWERS.
POS/NEG

G5. ACTIONS.
+ F:OK, LET'S TRY. B:TWOGRPS


FRAME 323.00  (Q) LABEL= TWOGRPS

G2. TEXT.
WRITE A TINT PROGRAM THAT WILL FIND THE SUM OF THE PAIR PRODUCTS OF THE
FOLLOWING TWO LISTS OF NUMBERS. (SAVE THE PROGRAM YOU WRITE.  YOU
WILL BE ABLE TO USE IT IN THE NEXT EXERCISE.)
```

EXAMPLE C.3--Student Printout

```
   SUPPOSE YOU WANTED TO FIND THE SUM OF THE 'PAIR PRODUCTS' OF TWO
LISTS OF NUMBERS (I.E. YOU WANT TO MULTIPLY EACH ENTRY IN THE FIRST
LIST BY ITS CORRESPONDING ENTRY IN THE SECOND LIST AND SUM THE
PRODUCTS).  THE TWO LISTS WILL HAVE THE SAME NUMBER OF ENTRIES.


DO YOU THINK YOU COULD WRITE SUCH A PROGRAM NOW?

*YES
OK, LET'S TRY.


WRITE A TINT PROGRAM THAT WILL FIND THE SUM OF THE PAIR PRODUCTS OF THE
FOLLOWING TWO LISTS OF NUMBERS.  (SAVE THE PROGRAM YOU WRITE.  YOU
WILL BE ABLE TO USE IT IN THE NEXT EXERCISE.)
```

<u>Comments</u>

The student is now asked to perform an exercise in TINT programming. This time he will be interacting directly with the TINT interpreter. After he indicates that he is ready, instructions are given and he is presented two lists of numbers (next page). These numbers would be varied by the lesson for each try at the exercise. If the student had answered NO to the question, he would have entered a remedial sequence to prepare him for the programming exercise.

EXAMPLE C.3--Lesson (cont'd)

```
FRAME 324.00  (P)

G6. STRUCT./STUD. PREF.
M N

G7. DIST.
NN

G8. M1 R1 S1 MS R2 C N.
50 100 N 0 100 R 10


FRAME 325.00  (P)

G9. HEADER.
PAIR NO.          LIST 1.    LIST 2.
      X                X          X

FRAME 326.00  (D)

G2. CRITERIA.
C:SET LSTFRM=327 F:\\I'LL LOAD TINT FOR YOU NOW...=B:220


FRAME 220.00  (D)

G2. CRITERIA.
F:STANDBY. C:LINK(1)=ARRAY(99000,0,0,0,0,0,0,0,0,0) B:PINT
```

4

EXAMPLE C.3--Student (cont'd)

```
    PAIR NO.          LIST 1.     LIST 2.
         1-             32          62
         2-             69          49
         3-             64          35
         4-             55          47
         5-             39          44
         6-             62          38
         7-             46          46
         8-             53          39
         9-             21          72
        10-             21          65


     I'LL LOAD TINT FOR YOU NOW...STANDBY.


     *ENTER COMMAND
     ?START

     *   1.00 INTEGER ENTRIES, I;
     *   2.00 REAL LIST1 50, LIST2 50, SUMPP;
     *   3.00 READ ENTRIES;
     *   4.00 I=0;
     *   5.00 LOOP1. READ LIST1[I];
     *   6.00 I=I+1;
     *   7.00 IF I LS ENTRIES;
     *   8.00 GOTO LOOP1;
     *   9.00 I=0;
     *  10.00 LOOP2. READ LIST2[I];
     *  11.00 I=I+1;
     *  12.00 IF I LS ENTRIES;
     *  13.00 GOTO LOOP2;
     *  14.00 SUMPP=0;
     *  15.00 I=0;
     *  16.00 LOOP3. SUMPP=SUMPP+LIST1[I]*LIST2[I];
     *  17.00 I=I+1;
     *  18.00 IF I LS ENTRIES;
     *  19.00 GOTO LOOP3;
     *  20.00 FORMAT FRMT, C'SUM OF PAIR PRODUCTS = ', F14.4;
     *  21.00 PRINT FRMT, SUMPP;
     *  22.00 ?EXECUTE
```

Comments

Frame 324 sets up the drawing of pseudorandom numbers from a bivariate normal distribution and Frame 325 sets up the presentation format. The list of numbers for the exercise is presented and PIANIT obtains the TINT interpreter (PINT version) via control Frame 220.00. TINT then initiates the programming task by printing *ENTER COMMAND and the student inputs a perfect twenty-one line program, followed by an ?EXECUTE command on line 22.00. Had TINT discovered any errors in the student's inputs, they would have been flagged and the student would have corrected them prior to execution.

EXAMPLE C.3--Lesson (cont'd)

```
FRAME 327.00  (Q)

G2. TEXT.
WHAT ANSWER DID YOU GET?

G4. ACTIONS.
C:FUNCTION SS(X,Y)=SUM PROD VALUES(I,X) FOR(J=1,Y I=1,N(1))
C:FUNCTION PP=SUM VALUES(I,1)VALUES(I,2) FOR(I=1,N(1))
C:FUNCTION CORR=SUM
SUM(PP-S1*S2/N(1))/SQRT((SS(1,2)-S1*S1/N(1))(SS(2,2)-S2*S2/N(1)))
FOR(S1=SS(1,1) S2=SS(2,1))


FRAME 328.00  (Q)

G3. ANSWERS.
1+ PP
2  SS(1,1)SS(2,1)
3  SS(1,2)+SS(2,2)
4  SS(1,2)SS(2,2)

G4. ACTIONS.
1 F:THAT'S RIGHT.  VERY GOOD.  B:333
2 F:NO, YOU SUMMED EACH COLUMN SEPARATELY AND THEN MULTIPLIED THE SUMS.
3 F:NO, YOU SUMMED THE SQUARES OF EACH COLUMN SEPARATELY AND THEN
  F:ADDED THE TWO SUMS.
4 F:NO, YOU SUMMED THE SQUARES OF EACH COULMN SEPARATELY AND THEN
  F:MULTIPLIED THE SUMS.
- F:NO, I DON'T KNOW HOW YOU GOT THAT.  MAYBE IT WAS A TYPING ERROR.
  R:TRY ONCE MORE.
- F:NO, THAT ISN'T THE CORRECT ANSWER.
```

EXAMPLE C.3--Student (cont'd)

```
ENTRIES = ? 1Ø
  LIST1(Ø)=?32
  LIST1(1)=?69
  LIST1(2)=?64
  LIST1(3)=?55
  LIST1(4)=?39
  LIST1(5)=?62
  LIST1(6)=?46
  LIST1(7)=?58
  LIST1(8)=?21
  LIST1(9)=?21
  LIST2(Ø)=?62
  LIST2(1)=?49
  LIST2(2)=?35
  LIST2(3)=?47
  LIST2(4)=?44
  LIST2(5)=?38
  LIST2(6)=?46
  LIST2(7)=?39
  LIST2(8)=?72
  LIST2(9)=?65
SUM OF PAIR PRODUCTS =      21517.ØØØØ
*EXECUTION COMPLETE


*ENTER COMMAND
?READY



WHAT ANSWER DID YOU GET?

*215L7*
21517.ØØØØ
THAT'S RIGHT.  VERY GOOD.
```

<u>Comments</u>

In response to the ?EXECUTE command, TINT requests the entries for LIST1 and LIST2. The student types in the numbers from the lists and the program computes the SUM OF PAIR PRODUCTS. Having completed the exercise, the student returns to the lesson by typing ?READY. The lesson then asks for and evaluates the student's answer. This is accomplished by the CALC functions defined by the lesson author in Group 4 of Frame 327.ØØ and by the anticipated answers in Group 3 of Frame 328.ØØ, which are written in terms of constituent functions. It was necessary that anticipated answers for the programming exercises be specified as functions, because each student uses different data for each exercise.

Had the student made errors while writing the program, the lesson would have asked: DO YOU WANT YOUR ERRORS ANALYZED? If the student answered YES, the lesson would proceed to explain the nature and remedy for the errors found by TINT. However, in this case the student's performance was perfect for the exercise.

End of EXAMPLE C.3