Development and Application of Computer Assisted Instruction for Mini-Computers

Thomas A. Burne

Western Michigan University

8-1977

Follow this and additional works at: https://scholarworks.wmich.edu/masters_theses

Part of the Computer Sciences Commons, and the Psychology Commons

Recommended Citation
https://scholarworks.wmich.edu/masters_theses/2252

This Masters Thesis-Open Access is brought to you for free and open access by the Graduate College at ScholarWorks at WMU. It has been accepted for inclusion in Master's Theses by an authorized administrator of ScholarWorks at WMU. For more information, please contact maira.bundza@wmich.edu.
DEVELOPMENT AND APPLICATION
OF COMPUTER ASSISTED INSTRUCTION
FOR MINI-COMPUTERS

by

Thomas A. Burne

A Thesis
Submitted to the
Faculty of The Graduate College
in partial fulfillment
of the
Degree of Master of Arts

Western Michigan University
Kalamazoo, Michigan
August 1977
ACKNOWLEDGEMENTS

I would first like to express my sincere appreciation to Dr. Arthur Snapper for both the opportunity to work in his laboratory and for his guidance and support.

I would like to thank my wife, Nancy, for her encouragement during the course of my studies and for her help in the editing and typing of this manuscript.

Finally, I would like to thank Dr. Douglas Anger who initiated and cultivated my interest in the study of the experimental analysis of behavior, and who helped to shape my goals through his friendship.

Thomas A. Burne
INFORMATION TO USERS

This material was produced from a microfilm copy of the original document. While the most advanced technological means to photograph and reproduce this document have been used, the quality is heavily dependent upon the quality of the original submitted.

The following explanation of techniques is provided to help you understand markings or patterns which may appear on this reproduction.

1. The sign or “target” for pages apparently lacking from the document photographed is “Missing Page(s)”. If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting thru an image and duplicating adjacent pages to insure you complete continuity.

2. When an image on the film is obliterated with a large round black mark, it is an indication that the photographer suspected that the copy may have moved during exposure and thus cause a blurred image. You will find a good image of the page in the adjacent frame.

3. When a map, drawing or chart, etc., was part of the material being photographed the photographer followed a definite method in “sectioning” the material. It is customary to begin photoing at the upper left hand corner of a large sheet and to continue photoing from left to right in equal sections with a small overlap. If necessary, sectioning is continued again — beginning below the first row and continuing on until complete.

4. The majority of users indicate that the textual content is of greatest value, however, a somewhat higher quality reproduction could be made from “photographs” if essential to the understanding of the dissertation. Silver prints of “photographs” may be ordered at additional charge by writing the Order Department, giving the catalog number, title, author and specific pages you wish reproduced.

5. PLEASE NOTE: Some pages may have indistinct print. Filmed as received.

Xerox University Microfilms
300 North Zeeb Road
Ann Arbor, Michigan 48106

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
MASTERS THESIS

BURNE, Thomas Alan
DEVELOPMENT AND APPLICATION OF
COMPUTER ASSISTED INSTRUCTION FOR
MINI-COMPUTERS.

Western Michigan University, M.A., 1977
Psychology, experimental

Xerox University Microfilms, Ann Arbor, Michigan 48106
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>1</td>
</tr>
<tr>
<td>II</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>13</td>
</tr>
<tr>
<td>III</td>
<td>18</td>
</tr>
<tr>
<td>IV</td>
<td>20</td>
</tr>
<tr>
<td>V</td>
<td>35</td>
</tr>
<tr>
<td>VI</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>39</td>
</tr>
<tr>
<td>Appendix A</td>
<td>39</td>
</tr>
</tbody>
</table>
INTRODUCTION

Traditional educational techniques show little evidence of respect for or utilization of the results of research in the field of learning. Education, the branch of technology which is most concerned with the learning process, does not appear to observe the basic principles of learning developed through research in the experimental analysis of behavior.

Three notable shortcomings in traditional educational technology are especially evident. First, in most schools today, the contingencies in effect are not based on positive control, but, in fact, fall under the area of aversive control. Skinner (1959) pointed out that the student behaves primarily to escape the threat of a series of minor aversive events, such as instructor disapproval, peer criticism, etc. Arriving at the correct answer is in itself an insignificant event amid the inevitable by-products of aversive control, such as boredom, anxiety, and aggression. Secondly, the contingencies of reinforcement utilized are far from optimal. Unless explicit mediating behavior is determined in advance, the lapse of a few seconds between the response and reinforcement will destroy the effect. Traditionally, exam feedback is delayed days to weeks, in light of which it is extremely surprising to note that this system has had any effect at all. A third shortcoming is the lack of an efficient shaping system to develop the desired response through a series of short steps, each systematically reinforced.

The experimental analysis of behavior is concerned with contingencies
which control behavior. The major components of this are the situation in which the behavior occurs, the behavior itself, and the consequences of the behavior (Skinner, 1959). The majority of these principles and techniques have been developed in the animal laboratory. In the laboratory, the environment can be regulated to optimize the desired conditions. Such factors as deprivation and available stimuli can all be regulated by the experimenter. The response, i.e., the behavior itself, can also be easily defined, manipulated, and detected by the experimenter. Finally, in the animal laboratory, the programming of the consequences of behavior can be done with great accuracy.

Through manipulation of these contingencies, the animal can be taught to discriminate. A discrimination occurs when an animal has been trained to make one response in the presence of one stimulus, and another response in the presence of a different stimulus. This discriminative performance in the animal laboratory can be modified by manipulating the reinforcement contingencies associated with each stimulus.

Michael (1974) pointed out that the essential contingencies in effect in the animal laboratory are also in effect in human learning, i.e., acquiring verbal behavior. In most courses, what the student learns is to emit one response in the presence of one stimulus, and another response in the presence of a different stimulus. Michael noted that college learning in an animal training paradigm would be a form of multiple-stimulus, multiple-response discrimination learning.

However, in human learning, the three components of animal learning are not as simple to identify. The environment is not controlled to as great an extent as it is in animal studies. The animal establishes a rep-
ertoire by emitting responses to environmental stimuli, followed by exper­
imentor arranged consequences contingent upon performance. In human
learning, this sequence is not as easily controlled. Defining and conse­
quating a response is not a simple, one-step process. A student makes a
response when reading a text. What are the the consequences of the be­
havior? The student may listen to lectures and read textbooks, but the
reinforcement for such behavior is not as direct as is the pellet of food
used to reinforce a hungry animal. It has been said that students attend
college (a response) to receive an education (a reinforcer). However, it
is in the steps taken to obtain that education that the principles of op­
erant behavior and task analysis come into effect.

The concept of task analysis has emerged in response to the pressures
of practical instructional design. Task analysis examines the properties
of a behavior which is to be learned and the sequencing of the component
tasks. Task analysis first identifies the necessary prerequisite skills
which the learner must possess before being able to successfully complete
the task or objective (Bunderson, 1970). This sequence is repeated for
each task or objective.

Michael (1974) summarized the differences between the animal labora­
tory and the most common form of college learning, i.e., studying a text,
and found two major differences. First of all, Skinner's stimulus-response­
consequence learning formula is difficult to apply. It is simple in the
animal laboratory to define performance in terms of stimuli, responses,
and their consequences. However, these elements do not appear to exist
in any obvious form in the textbook learning situation. Secondly, the
learner has control over the presentation of the stimulus material in the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
college learning situation, whereas this is not the case in the animal laboratory.

A number of educational theories and systems have been developed based upon Skinner's analysis of the pertinent variables necessary for learning. The first of these was Skinner's teaching machine approach. From Skinner's point of view, education was still based on aversive control, the by-products of which were early drop-out, mental fatigue, forgetfulness, and apathy. In spite of progressive trends to utilize available reinforcers, Skinner felt that the bulk of educational technology was not as effective as possible. Conditioned reinforcers, such as grades and diplomas, were not directly contingent on behavior in an effective manner. Skinner stated that to properly arrange the number of contingencies which could be used in a single experimental session would require some type of mechanical help. The human organism is equally, if not more sensitive to the precise contingencies studied in the animal laboratory. To program these contingencies would require instrumental assistance.

Skinner's early teaching machine, as described in Cumulative Record (1959), consisted of a box with a window through which the student could read a question or problem printed on paper. In order to respond, the student set a numbered slide to the appropriate answer choice and turned a knob on the side of the machine. Reinforcement was immediate because the knob turned only if the attempted answer was right. Wrong answers were counted, but had no effect. One of the advantages of Skinner's teaching machine was that the student had to pass through a carefully designed sequence of steps in acquiring a complex behavior. Each step was small,
and moved the student closer to the designated behavior. A disadvantage to this early machine was that it could handle only multiple choice questions, thereby requiring no originality on the part of the student. Skinner later developed another teaching machine which allowed the student to write in the answer.

Skinner indicated several ways in which a teaching machine can be effective in much the same way as a one-to-one private tutor. First of all, there is constant interaction between the program and the student. Unlike textbooks, this induces sustained activity. Secondly, the machine requires complete comprehension of every point, either frame by frame or lesson by lesson, before the student can continue. Texts, on the other hand, proceed without this, and can leave a student behind. Thirdly, sequencing presents only that material which the student is ready for, is most likely to understand, and can respond to correctly. But the most effective aspect of the teaching machine is its immediate reinforcement of each correct response in order to efficiently shape the targeted behavior.

The next type of educational technology to be developed was the contingency managed approach, which sprang from Keller's (1968) lead. "Contingency is a relationship between a behavior and its consequences. Contingency management, therefore, means changing behavior through the regulation of this relationship" (Berman, 1971). Contingency management has come to stand for the applied use of reinforcement theory.

Contingency Managed Education (CME) techniques have been applied to teaching in psychology (Malott and Svinicki, 1969; Ferster, 1968; Lloyd and Knutzen, 1969; McMichael and Corey, 1969). CME involves the use of educational objectives. Frequent sampling of students' repertoires com-
bined with extensive remediation are key features. These objectives, like the teaching machines, increase the student's contact with the study materials, thereby increasing studying. The objectives can also direct the student to key points while requiring reading of the entire material. General objectives add relevance to specific instructional objectives (Clark, 1972).

The advantages of CME lie primarily in the demonstrated effectiveness of this approach to education. Another advantage is the inexpensiveness of its implementation. The cost of writing and copying objectives is by far less than the expense involved in the construction of a teaching machine or the use of additional textbooks as a learning supplement. Objectives can be easily revised as a result of performance analysis. However, as Michael (1969) pointed out, even excellent materials have little effect on the student if used improperly.

A disadvantage of CME is that its implementation in a large class requires the use of multiple choice questions for grading purposes. Another disadvantage inherent to CME is that students are forced to receive information in a "locked step" fashion which bores the fast learner and loses the slow learner.

Another approach to instruction is the Personalized System of Instruction (PSI), which was designed to facilitate learning and to avoid some of the problems inherent in the traditional lecture-readings format when applied to large college classes (Hess, 1974). PSI was initially referred to as the Keller Plan (Keller, 1968). Keller used the Skinner learning model and the techniques of operant conditioning to formulate what is now referred to as PSI. In the PSI approach, units of instruction are small,
and are assisted by objectives and/or study guides designed to call atten-
tion to important points. PSI requires mastery of all materials presented,
and has been shown to be an effective instructional method in 16 studies
(Hursh, 1976). An advantage of PSI is that it allows students to proceed
at their individual rates. However, this individualized pace can also be
a disadvantage as some students may not finish the entire sequence of
materials. Another advantage of PSI is its inexpensive implementation, a
characteristic of CME as well. Multiple choice questions are widely used
in PSI in order to facilitate immediate consequences in large classes.

Programmed Instruction (PI) is still another educational approach.
PI has been defined as "a planned sequence of experiences, leading to
proficiency in terms of stimulus-response relationships" (Espich and
Williams, 1967). "Planned sequence" refers to the instructor's sequencing
of the material to be learned. "In terms of stimulus-response relation-
ships" refers to Skinner's definition of the components of learning, i.e.,
a stimulus in the environment, a response by the student, and consequences
contingent upon the response. PI and Skinner's teaching machine follow
Skinner's analysis of learning as closely as any educational technique
described thus far, assuming that answering a frame correctly is actually
reinforcing. PI forces the student to participate in the learning pro-
cess by requiring him to respond to frequent questions. PI is primarily
implemented with written textbooks where the student writes the answer in
the textbook or in an accompanying workbook. PI allows the student to
proceed at his individual rate without the traditional classroom distrac-
tions. Programmed materials are broken down into small steps which have
been described as "optimally sized increments" (Pipe, 1966). Small steps
for learning, combined with active participation on the part of the student and immediate knowledge of results, make PI one of the most potent forms of instruction. One drawback of PI is that there are no contingencies which assure that the student responds to each frame. The answers are usually readily available, and thus some students may not follow the sequencing determined by the instructor. The major drawback of the implementation of PI has been the great cost of developing an effective program. Programming skills require a great deal of time to develop, and PI is only as effective as the writing technique of the instructor (Espich and Williams, 1967).

Computer Assisted Instruction (CAI) is the most recently developed educational technology. CAI is, in essence, programmed instruction written by a person and taught by a computer. CAI consists of interactions between the machine and the learner, with the computer functioning as the instructor (Morgan, 1969). In CAI, programmed text is entered into the computer by the instructor. Students then interact with the instructional material via some input-output device, such as a teletype, Decwriter, or video screen. In this interaction, the computer presents text material and prints a question covering that text. The student then types an answer to the question into the computer. The computer then compares the student's answer with those already programmed by the instructor. The sequence of learning is determined by the instructor when the lesson is first written, and is followed by the computer. Both the training material and the instructional logic are stored in the computer.

The two main formats of CAI are drill and practice, and tutorial CAI. Both formats are response contingent, but differ in terms of the amount of computer involvement in instructional decision making. Drill and practice
CAI is the most basic form, and was the first developed. It was developed in 1964 in California elementary schools by Suppes (Suppes, Jerman and Brian, 1968). Drill and practice is the simplest with respect to computer involvement, and consists of material presented in a linear format. A linear format is one in which the student's performance has no effect upon instructional sequencing. Drill and practice is the most widely used form of CAI, and is an effective instructional method (Suppes, 1969; Suppes, Jerman and Groen, 1965; Fishman, Keller and Atkinson, 1969). The other level of CAI is tutorial or dialogue CAI. Tutorial CAI allows the student to construct natural language responses, and thus exercise control over the learning sequence via responses. Tutorial programs have the capabilities of making instructional branching contingent upon a single response or upon a subset of the student's response history (Atkinson, 1967).

There are many advantages of CAI. It has all of the positive features of PI, including individualized rate of response. CAI also provides immediate feedback to the student's responses. CAI allows for sequencing of instructional materials by the instructor. Unlike PI, CAI requires the student to make a response in order for the lesson to continue. CAI also serves as a pretest for the student to assure that the student is ready to proceed with the sequence of learning. CAI, as noted, allows branching to occur. Branching refers to selection of material which is determined by the student's performance. CAI also provides automatic scoring, which is a major component of the approach. Scoring greatly facilitates revision of programmed instruction writing style. This points to another advantage of CAI as compared with PI. This advantage is the ease with which lessons written on a computer can be modified. To alter programmed
texts requires revision of an entire textbook. Multiple choice questions and questions requiring original answers are both possible using CAI. Cheating can be controlled for in CAI, because, unlike PI, the answers are not readily available to the student. Most subjective reports on CAI indicate that students find interaction with the computer to be reinforcing.

The disadvantages of CAI center around the availability of a computer to implement the program. The cost of a computer is much greater than the cost of mimeographing objectives, as in CME. Also, as with PI, CAI is very costly in time to implement. Writing effective CAI requires many revisions of style. One other drawback of CAI is that most instructors possess typing skills, but may not possess the skills required to enter text into a computer.

CAI also contains some elements of Computer Managed Instruction (CMI). CMI uses the computer to help the instructor administer and guide the instructional process, but relies upon separate hardware and learning materials. CMI has been shown to be the most cost effective method of sequencing instructional materials when dealing with large classes (Hansen, 1970; Laham, 1973; Fromer, 1972). The scoring and branching features of CAI incorporate the basic logic of CMI.

There are a large number of CAI programs in use today. Three of the most successful and widely known are the Stanford Project, the PLATO System, and the GNOSIS System. The Stanford Project was developed at the Institute for Mathematical Studies in the Social Sciences Department at Stanford University. The initial CAI program used a PDP-1 computer. This has now been replaced by an IBM 1500 System which consists of an IBM 1800 Central Processing System with bulk storage maintained on
tape and disk, a station controller, and other peripheral devices. PLATO, (Programmed Logic for Automated Teaching Operations), was developed at the University of Illinois. Instruction in over 30 subjects is available in PLATO, including a visual display which can simulate operant experiments. PLATO utilizes a large central computer. GNOSIS, which is available on the PDP-10 computer, incorporates the features of tutorial CAI.

Recent hardware developments with remote input-output terminals make CAI more feasible than ever before with respect to cost (Knezevich and Eye, 1970). The cost of a computer system has been declining rapidly since 1970. As shown by Knezevich and Eye (1970), with the rising costs of professors' salaries and the declining costs of computers, CAI is an even more cost effective method of instruction. CAI is most cost effective with small classes and self-instruction (Suppes, 1977). The special languages required to communicate with the computer have become very user oriented. A computer programmer is no longer required to create CAI lessons. A computer naive user can be guided by the CAI system to mastery of a language (Gerard, 1969).

In spite of the great proliferation of CAI languages available today, no CAI language is widely available for the PDP-8 mini-computer. The PDP-8 is widely used in psychological laboratories, with over 100 users in the SKED Users Group alone. CAI would be useful to document laboratory procedures, and to serve as a pretest to student utilization of the computer for research purposes. Learning computer skills on a computer provides a desensitization which can be beneficial in acquiring computer usage skills. CAI for the PDP-8 computer will be useful only if a CAI language is developed which is compatible with the PDP-8, and if that CAI language can be
easily implemented by non-sophisticated users.

The purpose of the present study is to develop a CAI language for the PDP-8 computer which can be easily learned. This study will develop such a language, and will document it using lessons written in the language. Subjects will then be exposed to the lessons and their associated contingencies. The CAI language will be considered effective if the subjects are able to write instructional sequences on their own after exposure to the lesson.
METHOD

Subjects

Twelve students served as subjects. The subjects' computer experience varied from naive to experienced, with the majority of subjects being naive.

Apparatus

All programming was done on a PDP-8e computer, manufactured by Digital Equipment Corporation (DEC).\(^1\) Input-output was handled by a Decwriter, also manufactured by DEC. An RK8 disc served as the OS/8 device. The software used to create the CAI program and the lessons consisted of U/W FOCAL\(^2\) and DEC's Symbolic Editor.

The following equipment is necessary to use TEACH:

1. A PDP-8 computer with 8k or more of memory and an OS/8 device.
2. A U/W FOCAL source.
3. An input-output device.
4. A copy of TEACH.

Procedure

The language chosen to create a CAI language for PDP-8 computers was U/W FOCAL. The reasons behind this choice were several. First of

---

\(^1\)DEC is the trademark of Digital Equipment Corporation of Maynard, Massachusetts.

\(^2\)U/W FOCAL can be obtained from Jim Van Zee, Department of Chemistry, BG-10, University of Washington, Seattle, Washington.
all, U/W FOCAL contains the necessary features. Secondly, it is easily obtainable. Finally, programs developed in U/W FOCAL can easily be converted into other languages including either FORTRAN IV or BASIC.

The first step in writing a computer program is to develop a flowchart. Such a flowchart traces all of the basic components necessary for the program, without reference to a particular language format. The flowchart in Figure 1 is that from which TEACH was developed. It is a flowchart which can be adapted to any language. It is an outline of how an instructional lesson would operate. A lesson in CAI is a series of sequentially numbered frames. Each frame consists of text, a question concerning the text, and a series of right and wrong possible answers.

As shown in Figure 1, the program begins by initializing all variables, i.e., setting them equal to zero. The program asks the subject's name, and allows this information to be typed in. The subject's name and the date are stored in an output file called SCORE.FD. Next, the available lesson options are printed out, and the subject is allowed to choose one. The subject's choice is recorded in SCORE.FD, and the requested lesson file is opened for input. The first character is sampled by the program to determine whether or not it is a %. The first character in a CAI lesson written in TEACH must be a %. If this is not the case, the program prints teacher error, the subject's score file is closed, and the lesson ends. If the first character is a %, the program then selects the next character. This character is checked against other characters. If the character is a T, the program inputs and prints characters until a ? is found. The program then searches for a % and checks the next character. If that character is an R, the subject is allowed
to enter his answer to the question which terminated the input of text. The subject's answer is terminated by a carriage return.

Next, the first programmed right answer is read in and compared with the subject's answer. If the two are the same, the number of the answer given by the subject is recorded in SCORE.FD, a congratulations message is printed to provide the subject with immediate feedback, and counter C is set equal to zero. The program then goes on to the next frame.

If, on the other hand, the subject's answer does not match the right answer with which it was compared, the next character is input. That character is checked by the program to determine if it is another possible right (R) answer. If a right answer is found, the program inputs the possible answer and compares it with the subject's answer. This continues until no more right answers remain, or until a correct match occurs.

If no possible right answers are found, the program then checks for possible wrong (W) answers. Wrong possible answers are then compared, one at a time, with the subject's answer. This can be terminated by a match of the subject's answer with one of the possible wrong answers. In this case, the program then searches for the corresponding %&. The %& allows a remedial message to be printed to the subject, clarifying the error. The number of the wrong answer is then recorded in SCORE.FD, and counter C is incremented by one. If C equals three, the subject has missed three consecutive questions, since every correct answer sets C back to zero. If C equals three, the subject is told that too many errors have been made, and the lesson ends. After a matched wrong answer,
the program then goes on to the next frame.

If the subject gives an answer which does not match any of the possible answers, the program searches for a %S, which then initiates branching. Branching refers to a portion of text which may or may not be presented in a lesson depending upon the occurrence of another event. That event is the %S. After a %S, additional frames will be presented until either a matched answer or no additional %S options are found.

Two other characters should also be defined. The %J signifies the end of a lesson, and is placed at the end of a lesson. When encountered, the %J ends the lesson, prints out the subject's score, closes the output file, and saves this file in the computer for future reference by the instructor. The other character which is used in TEACH but has not yet been discussed, is the %E. This character is used when writing lessons. The %E can be placed at the end of a frame so that each frame can be tested individually prior to testing by the subjects. The %E is also a somewhat standard character in CAI, and is used to detect errors in the instructor's writing of the program.

The FOCAL program TEACH was developed from the flowchart in Figure 1. Figure 2 is an actual copy of TEACH in U/W FOCAL. The commands used in U/W FOCAL are summarized in Table 1. For additional information on U/W FOCAL, consult the U/W FOCAL manual.\(^1\)

Next, CAI lessons documenting TEACH were written using the Symbolic Editor program to enter text into the computer. Three lessons were written, and are shown in Appendix A. The lessons were then tested by subjects.

\(^1\)ibid.
Testing consisted of subjects running lessons one and two in Dr. Arthur Snapper's laboratory. The computer was started for the subjects, and TEACH was called into core. From this point, the subjects were not disturbed until the lessons were completed. The subjects were allowed to work at their own pace. The programmed writing style used in the lessons was revised several times based upon analysis of the subjects' results. Following these revisions, a criterion was established which required the subject to complete the lessons and to write a CAI lesson based on knowledge gained in lessons one and two. This was felt to be a satisfactory demonstration of the program's effectiveness as an instructional tool, and the ease with which the CAI program TEACH can be learned by those lacking an extensive computer background.
RESULTS

The program TEACH was first written with each frame of a lesson as an individual file on the computer. This permitted complex branching. However, it quickly filled up the available directory space on the computer, and thus was not a feasible design for a PDP-8 CAI language.

TEACH was then changed to its present format, in which each entire lesson constitutes an individual file on the computer. TEACH allows remedial messages following matched wrong answers and limited branching in the present format, as well as a detailed scoring system. Two lessons were then written to document and test TEACH. It should be noted that a third lesson describing the Symbolic Editor was made available, and was adapted from a tested and revised lesson available in GNOSIS on the PDP-10 computer. Lesson four, which is shown in Figure 3, was written as a short model lesson to assist subjects in writing their own CAI lessons.

Lessons one and two were each tested by 12 subjects. Testing was conducted in such a manner that after an error pattern was observed in subjects' scores, testing was halted and the error was revised, based upon the data received from the subjects. The criterion for revisions was somewhat subjective, but generally followed these three rules:

1. If subjects consistently missed a particular frame, the frame was rewritten.
2. If subjects consistently gave the same wrong answer to a particular frame, the frame was searched for the misleading section and revised.
3. If the lessons were completed by the subjects with no consistent errors, the effectiveness of these lessons was tested by requiring the subjects to write a short CAI lesson.

Six subjects tested the lessons in their early format, and the data from
these tests led to revisions of lessons one and two. Also as a result of the data collected from these subjects, additional right and wrong answers were added which had not been initially anticipated. %S branching options were also added to frames to which wrong answers were consistently given, and to frames in which the content was particularly important.

The next six subjects completed the revised lessons, and no error patterns were detected. However, passages verbally reported as confusing were revised, and additional unanticipated answers were added based upon the subjects' results. Of the final six subjects to test lessons one and two, four were able to write instructional sequences of their own. One subject dropped out of the study due to time limitations. The final subject required extensive work on the Symbolic Editor, a delay which made it impossible to include that subject's results in the study, due to the termination of the study.
DISCUSSION

The results of the present study indicate that TEACH, a CAI language developed for the PDP-8 computer, is easily learned and can be effectively used as an instructional technique.

CAI fits Skinner's stimulus-response-consequence learning theory best of all the educational methods discussed. In CAI, the stimulus is a discrete frame to which the student must respond in order for the lesson to continue. All responses have immediate consequences contingent upon the correctness of the response. CAI was used in Psychology 535, Digital Control, and the students were required to complete unit feedback forms. Of those completed, approximately 80 per cent were highly positive with respect to the use of CAI.

TEACH can be incorporated into a useful contingency managed approach to the acquisition of laboratory skills. To be allowed to conduct an experiment involving computers, the student would first be required to complete CAI modules on the computer which would teach the required computer skills. Such a method of training would be much more efficient and standardized than the current method of verbal instruction. Supplemental materials could also be coordinated with CAI when such materials were available. This was done in Psychology 535 where CAI modules written in GNOSIS to teach the Symbolic Editor were supplemented by DEC's Introduction to Programming.¹

Another possible use of TEACH would be the establishment of a departmental CAI library. This library would be composed of short lessons written by students covering various skill areas, such as logarithms, matching law, and other important areas not always familiar to all students. The students writing the various CAI lessons would derive additional personal benefit from such a library in the additional knowledge gained from actually teaching the lessons.

One of the major improvements made in TEACH over other CAI languages, such as GNOSIS, is the scoring system. Data is the best method of effectively revising programmed instruction writing style. With GNOSIS, the instructor must search through the student's entire print-out, which is often very lengthy, to find the errors made so that poor frames can be revised. With TEACH, all answers are numbered, and an accurate listing of the student's answer to each question is presented to the student, and is also saved on the computer for the instructor's evaluation.

There are several constraints which must be followed when writing lessons using TEACH. Due to the fact that a text frame is terminated by a question mark (?), this symbol cannot appear in the text except as a terminator. This is shown in Frame 1 of Figure 3. The question mark (?) appears in the only possible place. The per cent (%), as mentioned, is a major control character, and should only be used as shown in Figure 3. Finally, a %J must be the last line of a lesson in order for the lesson to stop after the final frame.

There are several minor drawbacks of TEACH. Branching is very limited, and no repeat option is available. Repeat in GNOSIS allows a frame to be repeated any number of times until it is answered correctly. This can be
done in TEACH, but would involve using the %S option over and over, re-
typing the entire frame each time. In addition, GNOSIS can scan for extra
characters in answers. This means that if the correct answer to a ques-
tion was the single word "relay", and the answer typed was "the relay
card", GNOSIS calls the answer correct and prints a message that the extra
words were "the card". This can also be done to a limited extent using
TEACH. TEACH allows the instructor to list potential wrong answers so
that if there was a high probability that an answer would contain an extra
word, that answer could be scanned for, and, if found, a remedial message,
coordinated with the particular wrong answer, could be given to the stu-
dent. This would involve use of the %& option. Figure 3 demonstrates
the use of %&.

The primary goal of this study was the development of a CAI lan-
guage for the PDP-8 computer. As with all computer programs, revisions
and improvements can always be made.

TEACH was written in U/W FOCAL because U/W FOCAL is the easiest and
most powerful language for a beginning programmer. In addition, U/W FOCAL
is easily transferable to BASIC, FORTRAN IV, and Machine Language. BASIC
would be a good language in which to write TEACH because it is available
on micro-computers. However, there are many different versions of BASIC
for micro-computers, a fact which would limit the program's usefulness.
FORTRAN IV is widely transferable and could be used to write a CAI lan-
guage. However, the optimal language into which to convert TEACH would
be Machine Language. All languages, such as BASIC, FORTRAN IV, and FOCAL,
are originally programmed in Machine Language. Thus, writing TEACH in
Machine Language would eliminate the need for FOCAL, and would speed up

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
the operation of the program.

CAI has the potential to use what is currently known about behavior analysis to generate an instructional sequence. Through testing and improvements, CAI can be the best educational technique available as the costs of computers decrease and the materials available for CAI increase.
Table 1: U/W FOCAL commands.
<table>
<thead>
<tr>
<th>Command</th>
<th>Abbreviation</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASK</td>
<td>A</td>
<td>Allows a variable to be set equal to a number from the keyboard.</td>
</tr>
<tr>
<td>COMMENT</td>
<td>C</td>
<td>Anything typed following a &quot;C-&quot; is not read by the computer and has no effect on the program.</td>
</tr>
<tr>
<td>IF</td>
<td>I</td>
<td>Allows the program to branch depending on the occurrence of a character.</td>
</tr>
<tr>
<td>TYPE</td>
<td>T</td>
<td>Prints (outputs) text.</td>
</tr>
<tr>
<td>ZERO</td>
<td>Z</td>
<td>Used at the start of a program to set all variables equal to zero.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Abbreviation</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIN()</td>
<td>F</td>
<td>Inputs a single character from a lesson file.</td>
</tr>
<tr>
<td>FIND()</td>
<td>F</td>
<td>Inputs characters from a file up to a specified character.</td>
</tr>
<tr>
<td>FOUT</td>
<td>F</td>
<td>Outputs characters.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Abbreviation</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN INPUT</td>
<td>0 I</td>
<td>Opens input from either a separate file, such as a lesson, or from the TTY (keyboard).</td>
</tr>
<tr>
<td>OPEN RESTORE INPUT</td>
<td>0 R I</td>
<td>Opens input from a previously opened file; input begins at point where input last left off.</td>
</tr>
<tr>
<td>ECHO</td>
<td>E</td>
<td>Allows text to be printed.</td>
</tr>
<tr>
<td>OPEN OUTPUT</td>
<td>0 O</td>
<td>Opens a new output file.</td>
</tr>
<tr>
<td>OPEN RESTORE OUTPUT</td>
<td>0 R O</td>
<td>Opens output to a previously opened output file.</td>
</tr>
<tr>
<td>OUTPUT CLOSE</td>
<td>0 C</td>
<td>Closes an output file.</td>
</tr>
</tbody>
</table>
Figure 1: Flowchart from which TEACH was developed.
Figure 2: The U/W FOCAL program TEACH.
C/U-W-FOCAL: TEACH 07/15/77

01.01 ZIC-ZEROS ALL VARIABLES
01.02 C-VARIABLES USED: M+i,W,J,E,*,%C,R,N
01.03 C-M IS THE LESSON NUMBER
01.04 C-J RIGHT AND WRONG ANSWER POSSIBILITIES
01.05 C-J SIGNS THE END OF THE LESSON
01.06 C-X ALLOWS YOU TO PRINT REMEDIAL TEXT AFTER A WRONG
01.07 C-C COUNTS NUMBER OF MISSES
01.08 C-A STORES STUDENT ANSWER
01.09 C-B STORES PROGRAMMED ANSWER
01.10 T "TYPE YOUR NAME* FOLLOWED BY A CARRIAGE RETURN*"
01.20 0 0 SCORE
01.21 T "HERE ARE YOUR OPTIONS!!!"
01.22 0 I CAILIB  (X FIND(154))  TTY ! 0 TTY !
01.30 A 'WHICH LESSON DO YOU WANT?(ANSWER MUST BE A DIGIT)
01.40 A 'LESSON NOT FOUND-START OVER!
01.50 X FIND(165) C-GETS NEXT %
01.60 S X=FIN() C-GETS NEXT CHAR.
01.70 F K=1 S A(K)=0
01.80 T 'V C-ECHOS RUBOUT
02.01 C-STUDENTS ANSWER IN INPUT
02.10 F K=1 S A(K)=FIN()
02.20 S K-1 GOTO 2.20
02.30 S (A(K)-255) C-RUB0UT
02.40 S (A(K)-141) C-CR TERM. ANSWER
02.50 S K=K+1 GOTO 2.20
03.01 C-TEXT PRINTING LOOP
03.10 A'THIS IS A DEFAULT LOOP FOR WHEN NO LEGIT. CHAR. FOLLOWS A
03.20 0 I CAILIB  (X FIND(191))  TTY ! 0 TTY !
03.30 0 I 0 COUT(FIND(154)) 0 C! L EfC-PRINTS SCORE OF STUDENT
03.40 0 I 0 COUT(FIND(154)) 0 C! L EfC-INPUTS TEXT AND ASKS QUESTION
03.50 0 X FIND(165) C-GETS NEXT %
03.60 S X=FIN() C-GETS NEXT CHAR.
03.70 F K=1 S A(K)=FIN()
03.80 S (A(K)-255) C-RUB0UT
03.90 S (A(K)-141) C-CR TERM. ANSWER
04.01 C-TEACHER ERROR LESSON
04.10 T "GREAT YOU ARE RIGHT!"
04.20 X FIND(165) C-GETS NEXT CHAR.
04.30 F K=1 S A(K)=FIN()
04.40 S (A(K)-255) C-RUB0UT
04.50 S (A(K)-141) C-CR TERM. ANSWER
04.60 S K=K+1
04.70 G 4.20
04.80 T 'END OF LESSON!'
04.90 R
05.01 C-TEACHER ERROR LESSON
05.10 A'THIS IS THE END OF LESSON LOOP-CODED WITH A 'J'
05.20 0 I CAILIB  (X FIND(191))  TTY ! 0 TTY !
05.30 0 I 0 COUT(FIND(154)) 0 C! L EfC-PRINTS SCORE OF STUDENT
05.40 T 'EXITING FROM LESSON'
05.50 R

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
10.01 C-PROGRAMMED ANSWER READ IN AND STORED
10.05 A R/F-C-SETS R FOR RECORDING
10.10 D R I
10.15 S N=1
10.20 S B(N)=FIN()
10.25 I (B(N)-141) >5.256C-CHECKS FOR CR WHICH TERMINATES ANSWER
10.30 Y N
10.35 GOTO 10.20

13.01 C-COMPARISON ROUTINE FOR (R) COMES HERE FROM 5.25
13.10 I (A(K)-B(K)) >13.20 <13.50 IF A R MATCH
13.20 R/F-GOES TO 5.30 IF A R MATCH
13.30 BREAK 13.60
13.60 O R I IX FIND (165)
13.70 GOTO 5.186C-TO CHECK FOR NEXT R

14.01 C-COMES HERE IF NO MORE (R) ANSWERS
14.10 A W/C-SETS NUMBER
14.20 G=16C-SETS VARS. N EQUAL TO 1
14.25 S B(N)=FIN()
14.30 I (B(N)-141) >17.106C-AFTER CR GO TO 17
14.35 Y N
14.40 GOTO 14.25

15.01 C-STANDARD WRONG OPTION (IF NO MORE (W) ANSWERS)
15.10 O R I I! "WRONG" !I! 0 TTY:IC-RECORDS IN SCORE.FD
15.12 BREAK 15.15
15.15 O I TTY:EXT "WRONG" !I!
15.20 I C,I (C-3) 15.30. 16:10>15.306C-IF 3 WRONG ANS. IN ROW LESF. ENDS
15.30 O R I I! (X-211) 2:20:8:16:2.206C- CHECKS FOR X5 FOR BRANCHING

16.01 C-IF TOO MANY ERRORS IN A ROW
16.10 O I TTY:EXT "TOO MANY ERRORS" !I!
16.20 T "YOU HAVE MADE 3 ERRORS IN A ROW!!!" !I!
16.25 T "YOU WILL HAVE TO START LESSON * HIT ' OVER" !I!
16.30 GOTO 7.10

17.01 C-LOOP FOR (W) MATCH RETURN AND REM. MESS. & CHECKS (C)
17.10 F K=1,1,16O 18
17.11 O R I I! "WRONG" !I! 0 TTY:IC-RETURNS HERE ON (W) MATCH
17.12 F K=1,1,601S B(K)=0
17.20 O I B=0
17.21 O I TTY:EXT "WRONG!!" !I! IC-PRINTS WRONG
17.30 O R IX FIND (165)IC-GETS X
17.35 S X=FIN()!I (X-166) 17.40.17.60.17.40IC-REMEDIAL MESSAGE
17.40 I (X-215) 17.45.5.01 17.45IC-CHECKS FOR T
17.45 I (X-202) 17.50.6.10.17.50IC-CHECKS FOR J
17.50 I (X-197) 17.30.7.10.17.30IC- CHECKS FOR E
17.60 O R I IX FIND (165)IT IC-INPUTS X5 TEXT AND WAITS AT ?
17.65 Y C,I (C-3) 17.70. 16.10.17.70IC-INC. C AND CHECK C
17.70 O R I I! X=FIN()
17.75 I (X-212) 17.80.5.01 17.80IC- CHECKS FOR T
17.80 I (X-202) 17.85.6.10 17.85IC- CHECKS FOR J
17.85 X FIND (165)GOTO 17.70

18.01 C-COMPARISON ROUTINE FOR (W)
18.10 I (A(K)-B(K)) >18.20 <18.50 IF (W) MATCH
18.20 R/F-RETURNS TO 17.11 IF (W) MATCH
18.50 BREAK 18.60IC-COMES HERE IF (W) DOES NOT MATCH
18.60 O R I IX FIND (165)IC-GETS X
18.65 O R IX FIND (165)X=FIN()!I (X-215) 15.10 14.20 15.10
Figure 3: A sample CAI lesson written in TEACH.
XT01 THIS IS A SAMPLE LESSON TO ILLUSTRATE HOW TO WRITE A CAI LESSON. THE LESSONS THAT YOU WRITE SHOULD FOLLOW THIS FORM. ALL YOU DO IS CREATE LESS.FD AND THEN BEGIN TYPING. EVERYTHING IN THIS LESSON WAS TYPED BY ME. YOU SHOULD ALWAYS RUN YOUR LESSONS FIRST TO CHECK FOR ERRORS.
I NUMBER QUESTIONS TO CORRESPOND WITH THE TEXT THAT THEY COVER. QUESTIONS MUST BE TERMINATED BY A QUESTION MARK.
01- WHAT TERMINATES A QUESTION?
XR11 ?
XR12 QUESTION MARK
XR13 A QUESTION MARK
XR14 A ?
XR15 A CARRIAGE RETURN
XR16 % NO, A QUESTION MARK. THE ? TERMINATES BOTH THE QUESTION AND THE TEXT.
XR17 %R12 QUESTION MARK
XR18 XR14 A QUESTION MARK
XR19 %R1 . 3 A QUESTION MARK
XR20 XR14 A ?
XR21 A CARRIAGE RETURN
XR22 % NO, A QUESTION MARK. THE ? TERMINATES BOTH THE QUESTION AND THE TEXT.
XR23 %R12 QUESTION MARK
XR24 XR14 A QUESTION MARK
XR25 %R1 . 3 A QUESTION MARK
XR26 XR14 A ?
XR27 A CARRIAGE RETURN
XR28 % NO, A QUESTION MARK. THE ? TERMINATES BOTH THE QUESTION AND THE TEXT.
XR29 %R12 QUESTION MARK
XR30 XR14 A QUESTION MARK
XR31 %R1 . 3 A QUESTION MARK
XR32 XR14 A ?
XR33 A CARRIAGE RETURN
XR34 % NO, A QUESTION MARK. THE ? TERMINATES BOTH THE QUESTION AND THE TEXT.
XR35 %R12 QUESTION MARK
XR36 XR14 A QUESTION MARK
XR37 %R1 . 3 A QUESTION MARK
XR38 XR14 A ?
XR39 A CARRIAGE RETURN
XR40 % NO, A QUESTION MARK. THE ? TERMINATES BOTH THE QUESTION AND THE TEXT.
XR41 %R12 QUESTION MARK
XR42 XR14 A QUESTION MARK
XR43 %R1 . 3 A QUESTION MARK
XR44 XR14 A ?
XR45 A CARRIAGE RETURN
XR46 % NO, A QUESTION MARK. THE ? TERMINATES BOTH THE QUESTION AND THE TEXT.
XR47 %R12 QUESTION MARK
XR48 XR14 A QUESTION MARK
XR49 %R1 . 3 A QUESTION MARK
XR50 XR14 A ?
XR51 A CARRIAGE RETURN
XR52 % NO, A QUESTION MARK. THE ? TERMINATES BOTH THE QUESTION AND THE TEXT.
XR53 %R12 QUESTION MARK
XR54 XR14 A QUESTION MARK
XR55 %R1 . 3 A QUESTION MARK
XR56 XR14 A ?
XR57 A CARRIAGE RETURN
XR58 % NO, A QUESTION MARK. THE ? TERMINATES BOTH THE QUESTION AND THE TEXT.
XR59 %R12 QUESTION MARK
XR60 XR14 A QUESTION MARK
XR61 %R1 . 3 A QUESTION MARK
XR62 XR14 A ?
XR63 A CARRIAGE RETURN
XR64 % NO, A QUESTION MARK. THE ? TERMINATES BOTH THE QUESTION AND THE TEXT.
XR65 %R12 QUESTION MARK
XR66 XR14 A QUESTION MARK
XR67 %R1 . 3 A QUESTION MARK
XR68 XR14 A ?
XR69 A CARRIAGE RETURN
XR70 % NO, A QUESTION MARK. THE ? TERMINATES BOTH THE QUESTION AND THE TEXT.
XR71 %R12 QUESTION MARK
XR72 XR14 A QUESTION MARK
XR73 %R1 . 3 A QUESTION MARK
XR74 XR14 A ?
XR75 A CARRIAGE RETURN
XR76 % NO, A QUESTION MARK. THE ? TERMINATES BOTH THE QUESTION AND THE TEXT.
XR77 %R12 QUESTION MARK
XR78 XR14 A QUESTION MARK
XR79 %R1 . 3 A QUESTION MARK
XR80 XR14 A ?
XR81 A CARRIAGE RETURN
XR82 % NO, A QUESTION MARK. THE ? TERMINATES BOTH THE QUESTION AND THE TEXT.
XR83 %R12 QUESTION MARK
XR84 XR14 A QUESTION MARK
XR85 %R1 . 3 A QUESTION MARK
XR86 XR14 A ?
XR87 A CARRIAGE RETURN
XR88 % NO, A QUESTION MARK. THE ? TERMINATES BOTH THE QUESTION AND THE TEXT.
XR89 %R12 QUESTION MARK
XR90 XR14 A QUESTION MARK
XR91 %R1 . 3 A QUESTION MARK
XR92 XR14 A ?
XR93 A CARRIAGE RETURN
XR94 % NO, A QUESTION MARK. THE ? TERMINATES BOTH THE QUESTION AND THE TEXT.
XR95 %R12 QUESTION MARK
XR96 XR14 A QUESTION MARK
XR97 %R1 . 3 A QUESTION MARK
XR98 XR14 A ?
XR99 A CARRIAGE RETURN
XR100 % NO, A QUESTION MARK. THE ? TERMINATES BOTH THE QUESTION AND THE TEXT.
REFERENCES


Morgan, R. M. A review of educational applications of the computer, including those in instruction, administration and guidance: A series two paper from ERIC at Stanford. ERIC Clearinghouse on Educational Media and Technology, Stanford, California, August, 1969, 13.


Suppes, P. The role of psychological models in computer-assisted instruction. Behavior Research Methods & Instrumentation, April, 1977, 9, 66.


APPENDIX A

THREE LESSONS WRITTEN IN TEACH

Lesson 1: How to use CAI
Lesson 2: How to write CAI lessons
Lesson 3: Basic Symbolic Editor commands
WELCOME TO LESSON 1 OF THE PDP-8 COMPUTER ASSISTED INSTRUCTION SERIES. TO INTERACT WITH THIS COMPUTER YOU WILL NEED TO FOLLOW A FEW SIMPLE RULES. THE FIRST THING THAT YOU MUST KNOW IS HOW TO ANSWER QUESTIONS FROM THE COMPUTER. AN ANSWER IS TERMINATED BY A CARRIAGE RETURN. THUS, ALL OF YOUR ANSWERS MUST BE LESS THAN ONE LINE LONG AND MUST BE FOLLOWED BY A CARRIAGE RETURN.

Q1- WHICH KEY TERMINATES YOUR ANSWERS TO THE COMPUTERS QUESTIONS?

ZR11 CARRIAGE RETURN
ZR12 THE CARRIAGE RETURN
ZR13 THE CARRIAGE RETURN KEY
ZR14 A CARRIAGE RETURN
ZR15
ZR16 C R
ZR17 C.R.
ZR18 CR
ZW11 PERIOD
ZN NO, YOU DO NOT NEED A PERIOD AFTER YOUR ANSWERS. IN FACT IF YOU ADD EXTRA CHARACTERS YOUR ANSWER WILL BE COUNTED WRONG!!
ZS THE CORRECT ANSWER WAS A CARRIAGE RETURN. SINCE THE CARRIAGE RETURN Terminates ALL ANSWERS, THEN NATURALLY THE MAXIMUM LINE LENGTH IS ONE LINE
WHAT TERMINATES AN ANSWER?
ZR11 CARRIAGE RETURN
ZR12 THE CARRIAGE RETURN
ZR13 THE CARRIAGE RETURN KEY
ZR14 A CARRIAGE RETURN
ZR15
ZR16 C R
ZR17 C R.
ZR18 CR
ZW11 ONE LINE
ZN WELL, YOU WERE ON THE RIGHT TRACK. ONE LINE IS THE MAXIMUM LENGTH OF AN ANSWER. THAT IS DUE TO THE FACT THAT A CARRIAGE RETURN Terminates AN ANSWER AND A LINE.
ZT02 THERE ARE A FEW KEYS THAT YOU NEED TO REFER TO BY TYPING THEIR NAME, DUE TO THE FACT THAT THEY OPERATE DIRECTLY UPON THE PROGRAM. ONE OF THESE CHARACTERS IS THE RUBOUT. IF YOU NOTICE AN ERROR TYPING AND YOU NOTICE IT BEFORE YOU HAVE TYPED ANOTHER CHARACTER, YOU CAN ERASE IT WITH THE RUBOUT KEY. BUT IF YOU HAVE TYPED ANOTHER CHARACTER AFTER THE ERROR, YOU WILL JUST HAVE TO LIVE WITH YOUR ERROR AND LET THE COMPUTER THINK THAT YOU ARE A DUMMY!!
Q2- WHICH KEY ERASES THE LAST CHARACTER TYPED (HINT: TYPE THE WORD BECAUSE IF YOU ACTUALLY TYPE THE CHARACTER IT WILL ERASE ITSELF)?
ZR21 RUBOUT
ZR22 THE RUBOUT
ZR23 THE RUBOUT KEY
ZR24 RUBOUT KEY
ZW21 THE CARRIAGE RETURN
ZN NO, THE CARRIAGE RETURN Terminates YOUR ANSWER. THE RUBOUT KEY CAN ERASE ERRORS IF THE ERROR IS THE LAST CHARACTER THAT YOU TYPED.
ZW22 THE CTRL C
ZN NO, WE WILL DISCUSS CTRL C IN THE NEXT FRAME. THE ANSWER TO THE QUESTION WAS THE RUBOUT KEY.
ZW23 CTRL C
% NO, THE CTRL C WILL BE EXPLAINED IN THE NEXT FRAME. THE RUBOUT KEY CAN ERASE THE LAST CHARACTER THAT YOU TYPE.
% IF YOU ANSWERED BY PRESSING THE RUBOUT KEY, YOU WERE ACTUALLY CORRECT. HOWEVER YOU DID NOT LISTEN TO MY HINT, SINCE, THE RUBOUT ERASES THE LAST CHARACTER TYPED IF YOU PRESSED THE RUBOUT KEY INSTEAD OF TYPING THE WORD RUBOUT THE COMPUTER NEVER "SAW" YOUR ANSWER. TYPE THE WORD COMPUTER AND DELIBERATELY MAKE AN ERROR AND CORRECT IT?

X% COMPUTER

% YOU SHOULD NOT HAVE ADDED THAT FINAL SPACE
% I DON'T KNOW EXACTLY WHAT YOU DID, HOWEVER I DO KNOW THAT YOU MADE AN ERROR. TRY AGAIN TO TYPE COMPUTER AND USE THE RUBOUT TO CORRECT A DELIBERATE ERROR?

X% COMPUTER

X% THE CORRECT ANSWER WAS JUST ONE WORD--- COMPUTER
X% THE NEXT THING THAT YOU SHOULD KEEP IN MIND IS THAT THE CTRL C IS THE "EXIT" CHARACTER. IF YOU WISH TO EXIT FROM THE LESSON, WITHOUT FINISHING, YOU SHOULD PRESS CTRL C. ALSO CTRL C IS TO BE USED IF THE LESSON SHOULD QUIT RUNNING. TO TYPE A CTRL C YOU SHOULD HOLD THE CTRL KEY DOWN AND PRESS THE C KEY.

X% TO EXIT FROM A LESSON WHAT KEY DO YOU PRESS WHILE HOLDING THE CTRL KEY DOWN?

X% THE C
X% THE C KEY

% NO, THE QUESTION ASKED WHICH KEY IN ADDITION TO THE CTRL KEY---THAT'S THE C KEY!!!
% YOU ARE CORRECT IN THAT PRESSING THOSE TWO KEYS SIMULTANEOUSLY WOULD EXIT YOU FROM THE LESSON. BUT THE QUESTION ASKED "WHICH KEY IN ADDITION TO THE CTRL KEY" THE CORRECT ANSWER WAS JUST THE C KEY. YOU SHOULD HAVE ANSWERED BY TYPING THE C KEY. REMEMBER THAT TO ACTUALLY TYPE CTRL C YOU HOLD THE CTRL KEY DOWN WHILE YOU PRESS THE C KEY, BUT YOU SHOULD NEVER USE THE CTRL C UNLESS YOU WANT TO EXIT FROM THE LESSON.

X% WHAT WILL HAPPEN IF YOU ACTUALLY HOLD THE CTRL KEY DOWN AND ALSO TYPE THE C KEY?

X% EXIT FROM LESSON
X% YOU WILL EXIT FROM THE LESSON
X% LESSON WILL END
X% THE LESSON WILL END

% NOTHING
% IF YOU TYPE THE WORD CTRL C NOTHING WILL HAPPEN TO THE PROGRAM, BUT IF YOU ACTUALLY HIT THE TWO KEYS SIMULTANEOUSLY, YOU WILL EXIT FROM THE LESSON.
X% REMEMBER THAT WHEN ANSWERING THE COMPUTER YOU SHOULD RESPOND IN SHORT CONCISE ANSWERS. DO NOT ADD EXTRA WORDS TO YOUR ANSWERS OR THEY WILL BE COUNTED WRONG. ANSWERS MUST MATCH EXACTLY. IF YOU ADD AN EXTRA SPACE OR AN EXTRA WORD TO A CORRECT ANSWER, IT WILL BE COUNTED WRONG.

X% WHEN RESPONDING TO A QUESTION, IF YOU MAKE ONE SPELLING ERROR THAT YOU CAN NOT CORRECT, WILL YOUR ANSWER STILL BE COUNTED CORRECT?

X% NO
X% YES

% NO, IT WOULD BE COUNTED WRONG!!! ALL ANSWERS MUST MATCH EXACTLY. IF YOU TYPE A ANSWER THAT IS COUNTED WRONG, AND YOU THOUGHT IT WAS RIGHT---CHECK FOR SPELLING ERRORS AND OR EXTRA WORDS.
X% IF YOU MAKE THREE ERRORS IN A ROW, THE PROGRAM WILL EXIT YOU FROM THE LESSON. YOU WILL HAVE TO RESTART THE LESSON, SO IT IS VERY IMPORTANT THAT YOU CAREFULLY READ EACH FRAME.

X% HOW MANY ERRORS IN A ROW ARE NECESSARY FOR THE
PROGRAM TO EXIT YOU FROM A LESSON?
$R51 3
$R52 THREE
$R53 THREE ERRORS
$R54 3 ERRORS
$R55 3 OR MORE
$R56 THREE OR MORE
$R57 OVER THREE
$R58 OVER 3
$W51 2
% NO, IF YOU MAKE THREE ERRORS IN A ROW, YOU WILL
HAVE TO START THE LESSON OVER.
$T06 AFTER THE LAST FRAME IN A LESSON, YOUR SCORE WILL BE
PRINTED OUT ON THE TERMINAL. YOUR SCORE IS ALSO
OUTPUT TO A FILE ON THE COMPUTER FOR THE CREATOR OF THE
LESSON TO USE IN REVISING AND EDITING THE LESSON. AS SOON
AS YOUR SCORE IS DONE PRINTING OUT, THE LESSON IS OVER.
Q6- WHAT WILL BE PRINTED OUT FOLLOWING THE FINAL FRAME?
$R61 THE SCORE
$R62 SCORE
$R63 YOUR SCORE
$R64 MY SCORE
$W61 THE LESSON IS OVER
% NO, AFTER THE LAST FRAME, YOUR SCORE WILL BE
PRINTED OUT FOR YOU. AFTER THAT THE LESSON IS OVER.
WELCOME TO LESSON TWO. THIS LESSON WILL TEACH YOU HOW TO WRITE YOUR OWN C.A.I. MODULES. YOU NEED TO BE FAMILIAR WITH A TEXT EDITOR SUCH AS DEC'S SYMBOLIC EDITOR OR TECO. YOU WILL NEED TO BE ABLE TO CREATE LESSON FILES USING ONE OF THE TEXT EDITORS.

Q1. DO YOU KNOW A TEXT EDITOR WELL ENOUGH TO USE IT TO CREATE A LESSON FILE?

[R11 YES
[N11 NO

% DON'T FEEL BAD, THE EDITOR IS NOT TOO HARD TO LEARN. YOU SHOULD FIRST READ PAGES 5-11 TO 5-30 IN DIGITAL EQUIPMENTS INTRODUCTION TO PROGRAMMING. IT IS A NICE SUMMARY OF THE SYMBOLIC EDITOR. AFTER YOU HAVE READ IT, YOU SHOULD RUN LESSON 3 WHICH IS A SUMMARY OF THE MOST IMPORTANT COMMANDS THAT YOU WILL NEED TO CREATE LESSONS FOR C.A.I.

NOW YOU SHOULD HOLD THE CTRL KEY DOWN AND TYPE C. THIS WILL EXIT YOU FROM THIS LESSON.

Q2. TO CREATE A LESSON FOR C.A.I. REQUIRES THAT YOU CREATE TWO FILES. ONE IS THE LESSON ITSELF AND THE OTHER IS A DIRECTORY OF ALL LESSONS. THE DIRECTORY IS CALLED CAILIB.FD.

IN CAILIB.FD YOU TYPE ALL OF THE LESSONS THAT ARE AVAILABLE TO BE RUN. THE LESSONS SHOULD BE LISTED MUCH LIKE THE LESSON OPTIONS AT THE START OF THIS LESSON. ALL YOU HAVE TO TYPE IS THE LESSON NUMBERS AND TITLES. THE PROGRAM WILL DO THE REST!!!

Q2. WHAT IS THE NAME OF THE DIRECTORY FILE?

[R21 CAILIB.FD
[S21 CAILIB
[V21 CAILIBFD

% CLOSE BUT YOU FORGOT THE .FD EXTENSION AND WITHOUT IT, THE COMPUTER WOULD NOT BE ABLE TO FIND YOUR DIRECTORY. THE CORRECT ANSWER WAS CAILIB.FD.

[W22 CAILIBFD
% CLOSE, HOWEVER YOU LEFT OUT THE PERIOD BETWEEN THE B AND THE F. THE NAME OF THE DIRECTORY MUST BE EXACTLY CAILIB.FD.

Q3. THE NEXT FILE YOU CREATE WILL BE THE LESSON. LESSONS ARE NUMBERED SO THAT THE COMPUTER CAN DISTINGUISH THEM FROM ONE ANOTHER. THEY ARE NAMED LESS(M).FD. THE (M) IS A DIGIT. THE NUMBER CORRESPONDS TO WHAT NUMBER THE LESSON IS. FOR EXAMPLE, IF I WERE CREATING MY FIRST LESSON, I WOULD NAME IT LESS1.FD.

Q3. WHAT WOULD YOU NAME A LESSON THAT WAS TO BE THE FOURTH LESSON THAT YOU CREATED?

[R31 LESS4.FD
[S31 LESS 4.FD
[V31 LESS4 FD

% NO, YOU MUST ABBREVIATE THE WORD LESSON. IT MUST BE EXACTLY IN THE FORM LESS4.FD.

[Z32 LESS4.FD
% NO, YOU MUST SHORTEN LESSON TO LESS. THE CORRECT ANSWER WAS LESS4.FD.

[W33 LESS4FD
% YOU WERE CLOSE, BUT YOU LEFT OUT THE PERIOD BETWEEN LESS4 AND THE FD EXTENSION. THIS WOULD HAVE CAUSED A PROBLEM. THE CORRECT ANSWER WAS LESS4.FD.

[X34 LESS4
% NOT QUITE, YOU FORGOT THE EXTENSION. THE CORRECT ANSWER WAS LESS4.FD.

[Z35 LESS4 FD
% WHEN CREATING FILES, IT IS VERY IMPORTANT THAT YOU NAME YOUR FILES EXACTLY LIKE I SPECIFIED. ALL LESSONS MUST BE NAMED AS FOLLOWS. LESS(M).FD, WHERE THE (M) IS A NUMBER(DIGIT). THEREFORE:
Lesson 1 would be LESS1.FD
Lesson 2 would be LESS2.FD
What would you name the sixth lesson?

You should have named it LESS6.FD.

Each lesson consists of a series of frames. Each frame consists of the following:

1) Text and a question
2) Possible right answers
3) Possible wrong answers
4) Remediation of matched wrong answers
5) Branching if no matched answer

Frames in a lesson are numbered consecutively.

04- What does the first part of a frame in a lesson consist of?

No! The computer would not accept that.

You should have named it LESS6.FD.

Within each frame there is text, a question, possible right answers, possible wrong answers, remediation and branching.

What is the first part of a frame in a CAI lesson?

No, the question wanted to know what the first part of a frame consisted of. The answer was text.

Don't be concerned if you missed that one. All I was doing was explaining the general flow of a CAI lesson.

Within each frame there is text, a question, possible right answers, possible wrong answers, remediation and branching.

70- What is the first part of a frame in a CAI lesson?

No, text is first, then a question

Text can be as long as necessary. That is, it can be more than one line.

The actual text must be preceded by the following:

%t  SPACE

For example, if I were starting the first frame of text in a lesson, I would type:

%t1 WELCOME TO .......

The percent must be first, followed by the letter t, then followed by the number of the frame, and then by a space. After the space, you may type as much text as you have for that frame.

Q5- When beginning each text frame, you must first type a percent sign, followed by a number. Then before you type in text you must type what?

%r51 SPACE
%r52
%r53 A SPACE
%r54 A
%r51 NUMBER

No, first you type a percent sign and then the frame number and then a space.

%t06 Text is terminated by a question and a question mark.

Q6- What terminates text?

%r61 A QUESTION AND A QUESTION MARK
%r62 QUESTION AND QUESTION MARK
%r63 A QUESTION AND QUESTION MARK
%r64 QUESTION QUESTION MARK
%r65 A QUESTION AND ?
%r66 QUESTION AND ?
%r67 ?
%r68 ?
%r69 QUESTION AND ?
%w61 A QUESTION
THAT'S RIGHT, YOU MUST HAVE A QUESTION AT THE END OF TEXT, BUT YOU MUST ALSO HAVE A QUESTION MARK.
ALL TEXT MUST BE FOLLOWED BY A QUESTION AND A QUESTION MARK. THE QUESTION COVERS THE MATERIAL PRESENTED IN THE TEXT. A QUESTION MARK MUST FOLLOW THE QUESTION.
WHAT FOLLOWS THE QUESTION?

QUESTION MARK

A QUESTION MARK

POSSIBLE RIGHT ANSWERS

NO; THE QUESTION MARK MUST BE BEFORE THE RIGHT ANSWERS.
CAUTION!!! SINCE TEXT IS TERMINATED BY A QUESTION MARK, YOU CAN NOT HAVE A QUESTION MARK IN TEXT. YOU WILL HAVE TO REFER TO IT AS I HAVE IN THIS FRAME, BY NAME. THAT MEANS THAT YOU WILL HAVE TO TYPE OUT THE WORDS QUESTION MARK.

CAN YOU HAVE A QUESTION MARK ITSELF IN THE TEXT PORTION OF A FRAME?

NO

YES

NO? YOU MUST SPELL THE WORDS OUT IF YOU WISH TO REFER TO A QUESTION MARK IN THE TEXT PORTION OF A FRAME, ONLY. THIS IS BECAUSE THE QUESTION MARK IS A TERMINATOR FOR TEXT INPUT.
AS YOU LEARNED EARLIER, TEXT CAN BE AS MANY LINES AS NEEDED. THE QUESTION CAN BE MORE THAN ONE LINE ALSO, AS LONG AS IT IS FOLLOWED BY A QUESTION MARK.

WOULD A QUESTION THAT IS THREE LINES LONG BE ACCEPTABLE AS LONG AS IT WAS FOLLOWED BY A QUESTION MARK?

YES

OF COURSE IT WOULD BE ACCEPTABLE. QUESTIONS CAN BE ANY LENGTH.
SO FAR YOU HAVE WRITTEN SOME TEXT AND ASKED A QUESTION ABOUT THE TEXT. NEXT YOU WILL LIST ALL POSSIBLE RIGHT ANSWERS. MOST QUESTIONS WILL HAVE MORE THAN ONE RIGHT POSSIBLE ANSWER.
WHAT FOLLOWS THE QUESTION AND QUESTION MARK?

RIGHT ANSWERS

ALL POSSIBLE RIGHT ANSWERS

POSSIBLE RIGHT ANSWERS

YOU LIST ALL POSSIBLE RIGHT ANSWERS

YOU WILL LIST ALL POSSIBLE RIGHT ANSWERS

ANSWERS

NOT QUITE, FIRST YOU LIST ALL POSSIBLE RIGHT ANSWERS. IT IS VERY IMPORTANT THAT YOU LIST ALL POSSIBLE RIGHT ANSWERS IMMEDIATELY FOLLOWING THE QUESTION AND QUESTION MARK.

ALL POSSIBLE ANSWERS

FIRST ALL POSSIBLE RIGHT ANSWERS.
POSSIBLE RIGHT ANSWERS ARE NUMBERED SIMILAR TO THE WAY TEXT WAS NUMBERED. RIGHT ANSWERS ARE PRECEEDED BY THE FOLLOWING
SPACE
FOR EXAMPLE IF I WERE GOING TO WRITE THREE POSSIBLE RIGHT ANSWERS FOR A QUESTION ON THE THIRD FRAME, I WOULD DO IT AS FOLLOWS:

ANSWER 1

ANSWER 2

ANSWER 3

THE FIRST NUMBER AFTER THE ZR REFERS TO THE FRAME NUMBER. THE SECOND NUMBER REFERS TO THE RIGHT ANSWER POSSIBILITY.
IF A POSSIBLE RIGHT ANSWER WAS LABELED ZR34 WHAT FRAME WOULD IT BE?

THREE

3

FRAME THREE

FRAME 3
ZR105 THIRD
ZR106 THE THIRD FRAME
ZR107 THIRD FRAME
ZR108 NUMBER THREE
ZR109 NUMBER 3
ZW101 FOUR
%R109 NUMBER 3

%R112 NINE
ZW111 SIXTY

%R122 ONE
ZW121 NONE

%R132 NINE
ZR133 UP TO 9
ZR134 NINE OR LESS
ZW131 MORE THAN NINE

%R142 1
ZR143 THE FIRST
ZR144 FRAME ONE
ZR145 FRAME 1
ZR146 THE FIRST FRAME
ZR147 FIRST
ZW141 THREE

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
YOU WOULD BE ON THE FIRST FRAME. THE THREE SIGNIFIES THAT IT IS THE THIRD WRONG POSSIBLE ANSWER FOR THE FIRST FRAME.

NO, IT WOULD BE THE THIRD POSSIBLE WRONG ANSWER ON THE FIRST FRAME.

AFTER EACH WRONG POSSIBILITY, YOU INSERT A REMEDIAL MESSAGE THAT CORRESPOND TO THE ANTICIPATED WRONG ANSWER THAT IS LISTED BEFORE IT. SO WHEN TYPING A LESSON YOU WOULD ARRANGE THE WRONG ANSWERS AS FOLLOWS:

RELY
NO, NOT QUITE, IT IS SPELLED RELAY.

.

ON THE LINE AFTER EACH WRONG POSSIBILITY, YOU TYPE %S FOLLOWED BY AS MUCH REMEDIATION AS NECESSARY. EVERY %W POSSIBILITY MUST HAVE A %F FOLLOWING IT. THE MESSAGE THAT YOU TYPE FOLLOWING THE %F CAN BE MORE THAN ONE LINE LONG

WHAT MUST FOLLOW EVERY %W?

A %S
A %S OPTION
%S FOLLOWED BY A REMEDIAL MESSAGE
REMEDIAL MESSAGE
THATS RIGHT, BUT FIRST THERE MUST BE A PERCENT SIGN FOLLOWED BY A %.
THE % KEY IS TYPED BY PRESSING THE SHIFT KEY AND THE 6 KEY, AND IT IS THE CHARACTER THAT ALLOWS REMEDIAL MESSAGES FOLLOWING EACH WRONG POSSIBILITY. IT MUST BE TYPED AT THE START OF THE LINE IMMEDIATELY FOLLOWING EACH %W POSSIBILITY.

WHAT KEY ALLOWS REMEDIAL MESSAGES FOLLOWING EACH POSSIBLE WRONG ANSWER?
SHIFT 6
SHIFT 6
%S
YOU TYPED THE 6 BUT FORGOT TO HOLD THE SHIFT KEY DOWN WHILE YOU WERE PRESSING IT.
IF A STUDENT GIVES AN ANSWER THAT IS CORRECT, THE PROGRAM GOES TO THE NEXT FRAME. IF THE STUDENT GIVES AN ANSWER THAT IS WRONG, BUT MATCHES ONE OF THE POSSIBLE WRONG ANSWERS THAT YOU HAVE LISTED, THEN THE %S PRINTS OUT THE CORRESPONDING REMEDIATION MESSAGE AND THEN THE PROGRAM GOES TO THE NEXT FRAME. HOWEVER: IF A STUDENT GIVES AN ANSWER THAT DOES NOT MATCH ONE OF THE POSSIBLE ANSWERS THAT YOU LISTED, THE PROGRAM WILL JUST PRINT OUT WRONG!! THERE IS AN OPTION AVAILABLE TO HANDLE THIS SITUATION. IT IS THE %S OPTION. IT ALLOWS YOU TO WRITE ADDITIONAL TEXT AND QUESTIONS WHICH WILL ONLY BE PRINTED WHEN THE STUDENT'S ANSWER DOES NOT MATCH ANY OF THE ANSWERS LISTED. THE %S IS OPTIONAL, UNLIKE THE %F, YOU MAY USE %S ON EVERY FRAME, ON DIFFICULT FRAMES OR NOT AT ALL.

IF A STUDENTS ANSWER MATCHES ONE OF THE WRONG ANSWERS THAT YOU LISTED, WILL THE %S OPTION BE PRINTED?

NO
YES
%F IS ACTIVATED ONLY BY AN ANSWER FROM THE STUDENT THAT DOES NOT MATCH ANY OF THE ONES THAT YOU LISTED.
AFTER YOU INSERT A %S, YOU HAVE BEGUN WHAT IS KNOWN AS BRANCHING. BRANCHING MEANS WRITING TEXT THAT WILL BE PRESENTED IN ONE CASE AND SKIPPED IN ANOTHER. IF A STUDENT GETS A QUESTION RIGHT OR IF HE MATCHES ONE OF THE POSSIBLE WRONG ANSWERS THAT YOU LISTED, THE TEXT FOLLOWING %S WILL NOT BE PRESENTED. BUT IF A STUDENT GIVES AN ANSWER THAT WAS NOT ONE OF THE ONES THAT YOU LISTED THEN THE TEXT FOLLOWING THE %S WILL BE PRESENTED.
Q18- IF YOU USE THE ZS OPTION, AND A STUDENT GIVES THE CORRECT ANSWER TO THE QUESTION, WILL THE ZS TEXT BE PRESENTED?
ZR181 NO
ZW181 YES
ZS NO? THE ZS OPTION TEXT IS ONLY PRESENTED IF THE STUDENTS ANSWER DOES NOT MATCH ANY OF THE RIGHT OR WRONG POSSIBLE ANSWERS THAT YOU LISTED.
Q19- IF A STUDENT GIVES AN UNMATCHED WRONG ANSWER TO A QUESTION? WHERE DOES THE PROGRAM GO?
ZR191 NEXT FRAME
ZR192 NEXT REGULAR FRAME
ZR193 THE NEXT FRAME
ZR194 NEXT REGULAR FRAME
ZR195 THE NEXT REGULAR FRAME
ZW196 NEXT PERCENT S OPTION
ZS NO? THE PROGRAM WOULD FIRST CHECK FOR THE NEXT PERCENT S OPTION BUT IF THERE WAS NOT ANOTHER ONE, THEN THE PROGRAM WOULD GO ON TO THE NEXT REGULAR FRAME.
ZT20 HERE IS AN EXAMPLE OF WHAT THE ZS OPTION WOULD LOOK LIKE AFTER YOU WROTE A FRAME USING IT.
ZR35 RELAY
ZW31 RELY
ZS YOU MISSPelled THE WORD, IT IS RELAY
ZS THIS WOULD BE TEXT THAT YOU WOULD WRITE TO HELP CLARIFY THE PRECEEDING FRAME. TEXT IS HALTED BY A QUESTION AND A QUESTION MARK.
ZR311 RIGHT ANSWERS LISTED
ZW311 WRONGS LISTED
ZS EACH FOLLOWED BY THIS
AS THE EXAMPLE ILLUSTRATED, A ZS "FRAME" LOOKS AND IS WRITTEN JUST LIKE A REGULAR FRAME.
IF THE STUDENTS ANSWER MATCHES ONE OF THE LISTED ANSWERS IN THE S OPTION THEN THE PROGRAM RETURNS TO THE NEXT REGULAR FRAME.
Q20- WHEN WRITING A ZS OPTION, ARE THE POSSIBLE RIGHT ANSWERS OR THE POSSIBLE WRONG ANSWERS LISTED FIRST?
ZR201 RIGHT
ZR202 RIGHT ANSWERS
ZR203 THE RIGHT ANSWERS
ZR204 THE POSSIBLE RIGHT ANSWERS
ZR205 THE POSSIBLE RIGHT ANSWERS ARE LISTED FIRST
ZR206 POSSIBLE RIGHT ANSWERS
ZW201 WRONG
ZS NO? JUST AS IN REGULAR FRAMES ALL RIGHT POSSIBLE ANSWERS ARE LISTED FIRST.
ZW202 WRONG ANSWERS
ZS NO? JUST AS IN A REGULAR FRAME, ALL RIGHT POSSIBLE ANSWERS ARE LISTED FIRST.
ZT21 IF THE STUDENTS ANSWER TO THE QUESTION IN THE ZS OPTION DOES NOT MATCH ANY OF THE ANSWERS LISTED, THEN ONE OF TWO THINGS CAN HAPPEN. IF THERE IS ANOTHER ZS OPTION, YOU CAN USE AS MANY IN A ROW AS YOU NEED, THEN THE TEXT ETC, FOLLOWING IT WILL BE PRINTED. IF THERE IS NOT ANOTHER ZS OPTION THEN THE PROGRAM RETURNS.
TO THE NEXT REGULAR SCHEDULED FRAME.
Q21- CAN YOU HAVE MORE THAN ONE ZS OPTION PER FRAME, IE-IN A ROW?
XR211 YES
XR211 NO
48
% THE ANSWER WAS YES. IF AFTER BRANCHING TO A PERCENT S OPTION, A CORRECT ANSWER BY THE STUDENT AUTOMATICALLY RETURNS THE STUDENT TO THE NEXT FRAME. BUT IF NO MATCH OCCURS THEN IF YOU HAD USED ANOTHER PERCENT OPTION IT WOULD HAVE BEEN PRINTED.
%T22 YOU CAN HAVE AS MANY FRAMES IN A LESSON AS YOU NEED. BUT THE COMPUTER WILL NEED TO KNOW WHICH FRAME YOU INTEND TO BE THE LAST ONE. TO SIGNIFY THE LAST FRAME, YOU TYPE A %J ON THE LINE FOLLOWING THE LAST LINE IN YOUR LESSON.
Q22- ON WHICH LINE IS THE %J TYPED?
XR221 THE LAST
XR222 THE LAST LINE
XR223 ON THE LAST LINE
XR224 ON THE LINE FOLLOWING THE LAST LINE IN YOUR LESSON
XR225 LAST
XR221 THE LAST LINE OF THE LESSON
%&NO, IT GOES AFTER THE LAST LINE IN THE LESSON
XR222 THE LAST LINE IN YOUR LESSON
%& NO, IT GOES AFTER THE LAST LINE IN THE LESSON
%T23 NOW YOU SHOULD BE READY TO WRITE YOUR OWN CAI LESSONS. HERE ARE TWO THINGS THAT WILL HELP YOU.
1) READ THE CAI MANUAL AVAILABLE FROM TOM BURNE.
2) AFTER YOU EXIT FROM THIS LESSON, TYPE THE FOLLOWING:
   TYPE LESS4.FD
   AND THEN PRESS CARRIAGE RETURN.
LAST QUESTION!!!- DID YOU FIND THIS LESSON:
CONFUSING, AWKWARD, BORING, TOO EASY, TOO HARD, INSTRUCTIONAL OR NONE OF THE ABOVE?
XR231 CONFUSING
XR232 BORING
XR233 AWKWARD
XR234 TOO EASY
XR235 TOO HARD
XR236 INSTRUCTIONAL
XR231 NONE OF THE ABOVE
%& I MIGHT AGREE WITH YOU!!!
%J

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
TO WRITE A CAI LESSON YOU MUST KNOW HOW TO USE A TEXT EDITOR PROGRAM SUCH AS THE SYMBOLIC EDITOR. THE SYMBOLIC EDITOR IS ONE OF THE SIMPLEST TEXT EDITORS TO LEARN AND USE. A TEXT EDITOR ALLOWS YOU TO INPUT TEXT INTO THE COMPUTER VIA THE KEYBOARD (DEC WRITER).

Q1- WHAT DOES THE SYMBOLIC EDITOR ALLOW YOU TO DO?

- INPUT TEXT
- TYPE IN TEXT
- TYPE TEXT
- TYPE CAI LESSONS
- TYPE LESSONS
- INPUT LESSONS
- WRITE CAI LESSONS
- WRITE A CAI LESSON
- LEARN AND USE

Q2- WHAT WILL YOU LEARN AND USE THE SYMBOLIC EDITOR FOR? BUT THE SYMBOLIC EDITOR WILL ENABLE YOU TO INPUT TEXT INTO THE COMPUTER AND THUS WRITE CAI LESSONS.

Q2- WHICH MODE WOULD YOU WANT TO BE IN TO TYPE IN THE TEXT OF A LESSON?

- TEXT MODE
- TEXT
- COMMAND

NO, YOU WOULD USE COMMAND MODE TO TAKE YOU TO TEXT MODE WHICH IS WHERE YOU WANTED TO BE TO TYPE IN TEXT.

Q3- WHEN YOU START THE EDITOR, YOU ARE INITIALLY IN COMMAND MODE. ANY ONE LETTER IS INTERPRETED AS A COMMAND. FOR INSTANCE THE FOLLOWING COMMANDS WOULD LIST PORTIONS OF THE TEXT BUFFER.

- L (RETURN) LISTS ENTIRE BUFFER
- 5L (RETURN) LISTS ONLY LINE 5
- 5L7L (RETURN) LISTS LINES 5 TO 7

Q3- WHAT LINES WOULD BE LISTED BY THE FOLLOWING COMMAND:

16,21L (RETURN)?

- LINES 16 THRU 21
- LINES 16,17,18,19,20, AND 21
- LINES 16,17,18,19,20,21
- LINES 16 TO 21
- 16 TO 21
- LINES 16 AND 21

NO, LINES 16,17,18,19,20 AND 21 WOULD HAVE BEEN LISTED BY THE COMMAND 16,21L.

Q4- THIS WOULD LIST LINES 16 THRU 21.

CERTAIN COMMANDS WILL AUTOMATICALLY PUT YOU IN TEXT MODE.

Q4- CERTAIN COMMANDS WILL AUTOMATICALLY PUT YOU IN TEXT MODE. THAT MEANS ANYTHING THAT YOU TYPE FOLLOWED BY A RETURN WILL BE ENTERED INTO A TEXT BUFFER. THESE COMMANDS ARE:

- A APPEND TO THE TEXT BUFFER, AFTER THE LAST LINE OF TEXT
- I INSERT INTO TEXT BUFFER BEFORE SPECIFIED LINE (E.G. SI INSERTS TEXT BEFORE LINE 5)
- ("I" BY ITSELF, INSERTS TEXT BEFORE LINE 1)
- C CHANGE A CERTAIN LINE
- "4C" DELETES LINE 4 AND ALLOWS YOU TO RETYPE IT
TO TERMINATE TEXT MODE AND RETURN TO COMMAND MODE, YOU TYPE
CTRL L (CTRL FORM) OR CTRL G (CTRL BELL).

Q4- TYPING AN 'A' WILL PUT YOU IN WHAT MODE?
XR41 TEXT
XR42 TEXT MODE
ZW41 COMMAND
% NO. TYPING AN 'A' WOULD PUT YOU IN TEXT MODE.
ZW42 COMMAND MODE
% NO. TYPING AN 'A' WOULD HAVE PUT YOU IN TEXT MODE.
XT05 THERE ARE TWO SPECIAL CHARACTERS THAT YOU WILL NEED TO KNOW
ABOUT. ONE IS THE RETURN. CARRIAGE RETURN IS THE TERMINATOR
FOR TEXT LINES AND FOR COMMANDS. THIS MEANS THAT ONLY LINES
THAT ARE FOLLOWED BY RETURN ARE ENTERED INTO THE TEXT BUFFER.
LIewise, COMMANDS ARE NOT EXECUTED UNTIL RETURN IS TYPED.
Q5- WHICH KEY IS THE TERMINATOR FOR COMMANDS?
XR51 RETURN
XR52 THE RETURN
XR53 THE CARRIAGE RETURN
XR55 CARRIAGE RETURN
XR56 CR
XR57 C R
XR58
ZW51 CTRL L
% NO. THE CARRIAGE RETURN IS THE TERMINATOR FOR COMMANDS.
XT06 THERE ARE TWO KEYS THAT YOU MAY USE TO CANCEL TEXT OR COMMANDS.
THE FIRST ONE IS THE CTRL/U KEY. IT CAN BE USED BY HOLDING THE CTRL
KEY DOWN, WHILE PRESSING THE U KEY. WHEN IN COMMAND MODE, TO CANCEL A
COMMAND WITH THE CTRL/U, ALL YOU HAVE TO DO IS TO PRESS THE RUBOUT KEY
ONCE. THE COMPUTER WILL TYPE A QUESTION MARK AND WILL BE READY TO
ACCEPT A NEW COMMAND. WHEN IN TEXT MODE, TO CANCEL THE ENTIRE CURRENT
LINE OF TEXT AND LET YOU CONTINUE AS IF THE LINE NEVER HAPPENED,
YOU TYPE CTRL/U.
Q6- WHAT MODE WOULD YOU HAVE TO BE IN TO USE CTRL/U TO CANCEL
A COMMAND?
XR61 COMMAND
XR62 COMMAND MODE
ZW61 TEXT MODE
% NO. TO BE ISSUING A COMMAND, YOU MUST BE IN COMMAND MODE.
ZW62 TEXT
% NO. TO BE ISSUING A COMMAND, YOU MUST BE IN COMMAND MODE.
XT07 THE OTHER KEY THAT CAN BE USED TO CANCEL COMMANDS OR TEXT
IS THE RUBOUT. IN THE EDITOR, A RUBOUT WILL CANCEL A COMMAND IF
YOU TYPE IT BEFORE THE RETURN. AND AS WITH THE CTRL/U IT LEAVES
YOU READY TO ISSUE ANOTHER COMMAND. IN TEXT MODE, A RUBOUT WILL
DELETE THE LAST CHARACTER THAT YOU HAVE TYPED. IT ALLOWS YOU
TO DELETE CHARACTERS ALL THE WAY BACK TO THE FIRST OF THE LINE.
Q7- IF I WERE EDITING TEXT AND I Pressed THE RUBOUT KEY THREE
TIMES IN A ROW ON THE SAME LINE, WHICH MODE WOULD I HAVE TO BE IN?
XR71 TEXT
XR72 TEXT MODE
XR73 THE TEXT MODE
ZW71 COMMAND
% NO. AS SOON AS YOU PRESS RUBOUT IN COMMAND MODE, THE COMMAND
THAT YOU WERE TYPING IS ERASED AND THE COMPUTER WILL TYPE A
QUESTION MARK AND RETURN AND BE READY FOR A NEW COMMAND. BUT IN TEXT
MODE YOU MAY ERASE FOUR CHARACTERS UP TO A WHOLE LINE OF
TEXT WITH THE RUBOUT KEY.
ZW72 COMMAND MODE
% NO. IN COMMAND MODE AS SOON AS YOU PRESS RUBOUT THE COMPUTER
RETURNS AND PRINTS A QUESTION MARK. IN TEXT MODE YOU MAY ERASE
FROM 1 CHARACTER UP THE FIRST ONE TYPED ON THE LINE THAT YOU ARE ON.
XT08 THE SYMBOLIC EDITOR IS AN EDITOR PROGRAM THAT CENTERS
AROUND NUMBERED LINES. THE EDITOR ASSIGNS A LINE NUMBER TO
EVERY LINE (IMPLICITLY-IE. IT DOES IT AUTOMATICALLY). IN COMMAND MODE, TO HAVE THE CURRENT LINE PRINTED, YOU TYPE
(PERIOD) FOLLOWED BY A RETURN. TO HAVE THE LAST LINE OF TEXT
PRINTED YOU TYPE / (SLASH) FOLLOWED BY A RETURN.
IT IS ALSO POSSIBLE TO BE ABLE TO PRINT OTHER LINES
BY USING THESE AS A REFERENCE FOR RELATIVE LINE SPECIFICATIONS
WITHOUT HAVING TO FIND OUT THEIR ABSOLUTE LINE NUMBERS.
AS AN EXAMPLE:
/-SL MEANS PRINT THE LINE THAT IS 5TH FROM THE BOTTOM LINE
Q8- TO PRINT A LINE THAT IS THE CURRENT LINE BUT NOT THE ACTUAL
LAST OR FINAL LINE OF TEXT, WHAT WOULD YOU TYPE (ASSUMING YOU ARE
IN COMMAND MODE)?
ZR81 .L
ZR82 .L FOLLOWED BY A RETURN
ZR83 .L(RETURN)
ZR84 .L RETURN
ZR85 .L
ZR86 .L(RETURN)
ZW81 /L
Z% /L WOULD HAVE LISTED THE LAST LINE IN THE TEXT BUFFER. IF YOU
WERE NOT ON THAT LINE, YOU WOULD THEN HAVE TO HAVE TYPED .L (RETURN).
ZT09 THERE IS NO REASON NOT TO KNOW THE CURRENT LINE NUMBER.
THERE ARE TWO KEYS THAT ARE USED FOR THIS. THE = (EQUALS KEY)
AND THE : (COLON KEY). THEY WORK IN A SIMILAR MANNER AS THE FOLLOWING
EXAMPLES ILLUSTRATE:
= PRINTS CURRENT LINE NUMBER
:: PRINTS CURRENT LINE NUMBER
/ PRINTS NUMBER OF THE LAST (FINAL) LINE
/= PRINTS NUMBER OF THE LAST (FINAL) LINE
ONE THING YOU WILL OFTEN WANT TO DO WITH THE EDITOR IS TO
EXAMINE THE PREVIOUS LINE OR THE FOLLOWING LINE. THIS MUST BE DONE
IN COMMAND MODE. TO DO THIS YOU WOULD TYPE
.-1L OR
.+1L
Q9- TO PRINT THE PREVIOUS LINE WHAT WOULD YOU TYPE?
ZR91 .-1L
ZR92 <
ZR93 SHIFT ,
ZR94 +1L
Z% THAT WOULD LIST THE VERY NEXT LINE. TO PRINT THE PREVIOUS LINE YOU
WOULD TYPE -1L.
ZT10 TO ENTER TEXT INTO THE COMPUTER VIA THE TELTYPE, YOU
WOULD TYPE "A (RETURN)". THIS WOULD ADD TEXT ONTO THE END OF
AN EXITING BUFFER OR AT THE START IF THE BUFFER IS EMPTY.
FOLLOWING TYPING "A (RETURN)* ANYTHING THAT YOU TYPE THAT TERMINATES
WITH A RETURN ENTERED INTO THE BUFFER, UNTIL YOU TYPE A CTRL/L
OR A CTRL/G TO RETURN TO COMMAND MODE.
Q10- WHEN IN COMMAND MODE, WHAT DO YOU TYPE TO ALLOW YOU TO
TO ENTER TEXT?
ZR101 A (RETURN)
ZR102 A FOLLOWED BY RETURN
ZR103 A
ZW101 CTRL/G
Z% NO, THE CORRECT ANSWER WAS THE KEY A FOLLOWED BY RETURN
ZT11 IF YOU WANT TO INSERT TEXT BEFORE A LINE, YOU USE THE I COMMAND.
FOR EXAMPLE THE COMMAND 26I (RETURN) WOULD ALLOW YOU TO INSERT TEXT
BEFORE THE 26TH LINE. THE I COMMAND WITH NO NUMBER PRECEEDING IT, WILL
INSERT TEXT BEFORE THE FIRST LINE OF THE CURRENT TEXT BUFFER.
Q11- TO INSERT TEXT BEFORE THE 3RD LINE OF THE CURRENT BUFFER, WHAT
WOULD YOU TYPE?
ZR111 3I
ZR112 3I (RETURN)
ZR113 3I FOLLOWED BY RETURN
ZW111 A
Z% THAT WOULD ALLOW YOU TO ENTER TEXT ALRIGHT. EXCEPT THE TEXT
WOULD BE PLACED AT THE END OF THE CURRENT TEXT BUFFER.
ZT12 SOMETIMES A LINE IS IN THE WRONG PLACE OR IS FULL OF MANY ERRORS
AND YOU WANT TO DELETE IT, OR PERHAPS A GROUP OF LINES. TO DO THIS
YOU NEED TO USE THE D COMMAND. THE FOLLOWING EXAMPLE SHOWS HOW TO
USE THE D COMMAND:

\[ 6D(RETURN) \text{ DELETES LINE 6} \]
\[ 4\text{D}(RETURN) \text{ DELETES LINES 4,5,6 AND 7} \]

In the first example, the old line 6 is now gone, and all subsequent lines have now been renumbered. In the second case, lines 4,5,6 and 7 have been deleted. And the old line 8 would now be line 4, thru implicit numbering by the computer.

Q12- Give a single command to delete from line 19 thru line 37 of a program?
\[ XR121 \text{ 19,37D} \]
\[ XR122 \text{ 19,37D(RETURN)} \]
\[ XR123 \text{ 19,37D FOLLOWED BY RETURN} \]
\[ XR124 \text{ 19,37 D} \]

% You were close, but you should not have added the space between the 7 and the D.

XT13 Sometimes the simplest way of correcting a line that has mistakes in it, is to delete the line and type a replacement for it. The "C" command is able to accomplish these two objectives. You are in text mode after you type a "C" command until you type a CTRL/G or a CTRL/L.

Q13- To use the "C" command do you have to be in text or command mode?
\[ XR131 \text{ COMMAND} \]
\[ XR132 \text{ COMMAND MODE} \]
\[ XR133 \text{ TEXT} \]
\[ XR134 \text{ TEXT MODE} \]

% No, to give any command requires you to be in command mode.

XT14 The S (search) command is one of the most powerful and useful commands in the editor for correcting errors within a line. "S" is followed by return, and then by the character to be searched. Then the tty(decwriter) prints the portion of the line leading up to and including the search character. If a search character is not found, then the line is not changed. Several things may happen once the character has been found in this way, depending on the key that the user types next:

<table>
<thead>
<tr>
<th>KEY</th>
<th>WHAT HAPPENS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>CTRL/U ENTIRE PRINTED PORTION OF LINE IS DELETED</td>
</tr>
<tr>
<td>2.</td>
<td>RETURN ENTIRE UNPRINTED PORTION OF LINE DELETED (LINE NOW ENDS AT SEARCH CHAR.)</td>
</tr>
<tr>
<td>3.</td>
<td>RUBOUT DELETES SINGLE CHARACTER TO THE LEFT OF THE SEARCH CHARACTER</td>
</tr>
<tr>
<td>4.</td>
<td>ANY CHARACTER FROM KEYBOARD INSERTS CHARACTERS IMMEDIATELY AFTER THE SEARCH CHARACTER</td>
</tr>
<tr>
<td>5.</td>
<td>LINE FEED DIVIDES THE LINE INTO 2 LINES AFTER THE SEARCH CHARACTER</td>
</tr>
<tr>
<td>6.</td>
<td>CTRL/L CONTINUE SEARCHING TO NEXT OCCURRENCE OF THE SEARCH CHARACTER</td>
</tr>
<tr>
<td>7.</td>
<td>CTRL/G CHANGE THE SEARCH CHARACTER TO THE NEXT ONE TYPED, AND SEARCH FOR IT</td>
</tr>
</tbody>
</table>

The entire buffer may be searched by typing S (search character)

A single line may be searched, as in 6S (search character)

A block of lines may be searched, as in 40-50S (search character)

The following questions, questions 14 and 15, will refer to the next line:

Abcdefghijklmn

You will be asked to type the way the line will look when
YOU HAVE EDITED IT IN THE MANNER DESCRIBED IN THE QUESTIONS.
Q14- IF I DO THE FOLLOWING:
  S(RETURN)
  G
  (RETURN)
WHAT IS THE LINE GOING TO LOOK LIKE?
ZR141 ABCDEFG
ZW141 ABCDEFG
% NOT QUITE, THE CORRECT ANSWER WOULD HAVE BEEN
ABCDEFG
BECAUSE THE SEARCH WOULD HAVE PRINTED UP TO AND
INCLUDING THE G.
ZT15 THIS FRAME ONLY CONSISTS OF A QUESTION COVERING THE LINE:
ABCDGFHJKLMN
Q15- WHAT WILL THE LINE LOOK LIKE IF I SEARCH FOR Z?
ZR151 ABCDEFGHJKLMN
ZR152 THE SAME
ZR153 SAME
ZR154 IDENTICAL
ZR155 UNCHANGED
ZW151 ABCDEFGHJKLMNZ
% NO, THE LINE DOES NOT CHANGE IF YOU SEARCH FOR
A CHARACTER THAT IS NOT CONTAINED IN THAT LINE.
ZJ