A Computer Aided Instructional System and its Application Towards a Course on Snobol4 Programming Language

Edelji

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

Part of the Computer Sciences Commons, and the Curriculum and Instruction Commons

Recommended Citation

https://scholarworks.wmich.edu/masters_theses/1503
A COMPUTER AIDED INSTRUCTIONAL SYSTEM AND ITS APPLICATION TOWARDS A COURSE ON SNOBOL4 PROGRAMMING LANGUAGE

by

Abtin Edelji

A Thesis
Submitted to the
Faculty of The Graduate College
in partial fulfillment of the
requirements for the
Degree of Master of Science
Department of Computer Science

Western Michigan University
Kalamazoo, Michigan
April 1984

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
A COMPUTER AIDED INSTRUCTIONAL SYSTEM AND ITS APPLICATION TOWARDS A COURSE ON SNOBOL4 PROGRAMMING LANGUAGE

Abtin Edelgi, M.S.
Western Michigan University, 1984

Computers are rapidly replacing humans in many tasks. Computer Aided Instructional System is an application of the computer in the teaching of a subject. Currently, a large number of educational institutes use Computer Aided Instructional Systems.

The objective of this thesis is to review some techniques in implementing such systems and to present a Computer Aided Instructional System called TEACH. The emphasis of this thesis is on the type of the directory used to minimize the search time for the TEACH System.
ACKNOWLEDGEMENTS

A very special thanks to my thesis advisor, Dr. Dionysios Kountanis, for his advice and encouragement during the design and implementation of the TEACH system. Thanks also to Dr. Mohammad Meybodi, a member of my committee. In addition, I would like to thank my dear friend, Ms. Eliana L. Silva for her encouragement during the rough months of implementation.

Abtin Edelji
INFORMATION TO USERS

This reproduction was made from a copy of a document sent to us for microfilming. While the most advanced technology has been used to photograph and reproduce this document, the quality of the reproduction is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help clarify markings or notations 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 through an image and duplicating adjacent pages to assure complete continuity.

2. When an image on the film is obliterated with a round black mark, it is an indication of either blurred copy because of movement during exposure, duplicate copy, or copyrighted materials that should not have been filmed. For blurred pages, a good image of the page can be found in the adjacent frame. If copyrighted materials were deleted, a target note will appear listing the pages in the adjacent frame.

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

4. For illustrations that cannot be satisfactorily reproduced by xerographic means, photographic prints can be purchased at additional cost and inserted into your xerographic copy. These prints are available upon request from the Dissertations Customer Services Department.

5. Some pages in any document may have indistinct print. In all cases the best available copy has been filmed.
EDELJI, ABTIN

A COMPUTER AIDED INSTRUCTIONAL SYSTEM AND ITS APPLICATION TOWARDS A COURSE ON SNOBOL4 PROGRAMMING LANGUAGE

WESTERN MICHIGAN UNIVERSITY M.S. 1984

University Microfilms International 300 N. Zeeb Road, Ann Arbor, MI 48106

Copyright 1984 by

EDELJI, ABTIN

All Rights Reserved
PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark. 

1. Glossy photographs or pages
2. Colored illustrations, paper or print
3. Photographs with dark background
4. Illustrations are poor copy
5. Pages with black marks, not original copy
6. Print shows through as there is text on both sides of page
7. Indistinct, broken or small print on several pages
8. Print exceeds margin requirements
9. Tightly bound copy with print lost in spine
10. Computer printout pages with indistinct print
11. Page(s) lacking when material received, and not available from school or author.
12. Page(s) seem to be missing in numbering only as text follows.
13. Two pages numbered. Text follows.
14. Curling and wrinkled pages
15. Other

University Microfilms International

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td></td>
<td>ii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td></td>
<td>vi</td>
</tr>
<tr>
<td>I. INTRODUCTION</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>II. REVIEW OF RELEVANT LITERATURE</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>2.1 Review of CAI Techniques</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>2.1.1 CMI (Computer-Managed Instruction)</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>2.1.2 CAI (Computer-Assisted Instruction)</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>2.1.3 Classification of Computerized Teaching Systems</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>2.2 TEACH System</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>III. EDITOR PROGRAM</td>
<td></td>
<td>13</td>
</tr>
<tr>
<td>3.1 Top Level Commands</td>
<td></td>
<td>13</td>
</tr>
<tr>
<td>3.1.1 EDIT Command</td>
<td></td>
<td>13</td>
</tr>
<tr>
<td>3.1.2 QUIT or EXIT Command</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>3.2 Second Level Commands</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>3.2.1 TEXT Command</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>3.2.2 EXAMPLE Command</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>3.2.3 EXERCISE Command</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>3.2.4 DONE Command</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>3.3 Third Level Commands</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>3.3.1 &quot;I&quot; or Insert Command</td>
<td></td>
<td>17</td>
</tr>
<tr>
<td>3.3.2 &quot;D&quot; or Delete Command</td>
<td></td>
<td>17</td>
</tr>
</tbody>
</table>
### Table of Contents, Con't.

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.3 &quot;P&quot; or Print Command</td>
<td>18</td>
</tr>
<tr>
<td>3.3.4 &quot;R&quot; or Report Command</td>
<td>19</td>
</tr>
<tr>
<td>3.3.5 &quot;S&quot; or Stop Editing Command</td>
<td>19</td>
</tr>
<tr>
<td>3.4 Structure of the Data File</td>
<td>19</td>
</tr>
<tr>
<td>3.5 Directory</td>
<td>20</td>
</tr>
<tr>
<td>3.5.1 Master Directory</td>
<td>21</td>
</tr>
<tr>
<td>3.5.2 Actual Directory</td>
<td>23</td>
</tr>
<tr>
<td>3.5.3 Combination of the Master and the Actual Directory</td>
<td>24</td>
</tr>
<tr>
<td>3.5.4 Performance Analysis of the Inverted Index by Subject Directory</td>
<td>26</td>
</tr>
<tr>
<td>3.6 Logical Design of the EDITOR Program</td>
<td>35</td>
</tr>
<tr>
<td>IV. TEACH MODULE</td>
<td>37</td>
</tr>
<tr>
<td>4.1 List of Commands</td>
<td>37</td>
</tr>
<tr>
<td>4.1.1 LEARN Command</td>
<td>37</td>
</tr>
<tr>
<td>4.1.2 EXAMPLE Command</td>
<td>39</td>
</tr>
<tr>
<td>4.1.3 EXERCISE Command</td>
<td>40</td>
</tr>
<tr>
<td>4.1.4 QUIT or EXIT Command</td>
<td>40</td>
</tr>
<tr>
<td>4.1.5 HELP Command</td>
<td>40</td>
</tr>
<tr>
<td>4.2 Random Selection of a Record</td>
<td>41</td>
</tr>
<tr>
<td>4.3 Logical Design of the TEACH Module</td>
<td>42</td>
</tr>
<tr>
<td>4.4 Logical Design of the TEACH System in General</td>
<td>44</td>
</tr>
<tr>
<td>V. POSSIBLE EXPANSIONS, LIMITATIONS AND MAINTENANCE OF THE TEACH SYSTEM</td>
<td>45</td>
</tr>
<tr>
<td>5.1 Possible Future Expansions</td>
<td>45</td>
</tr>
<tr>
<td>5.1.1 Graphic Capabilities</td>
<td>45</td>
</tr>
</tbody>
</table>
Table of Contents, Con't.

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.2 Artificial Intelligence</td>
<td>46</td>
</tr>
<tr>
<td>5.1.3 Extension of the EDITOR Program's Command Set</td>
<td>48</td>
</tr>
<tr>
<td>5.2 Limitations</td>
<td>48</td>
</tr>
<tr>
<td>5.2.1 Limitation on the Number of Characters</td>
<td>48</td>
</tr>
<tr>
<td>5.2.2 Disk Space Limitation</td>
<td>49</td>
</tr>
<tr>
<td>5.2.3 Graphic Limitation</td>
<td>49</td>
</tr>
<tr>
<td>5.3 Maintenance</td>
<td>49</td>
</tr>
<tr>
<td>APPENDIX A. A SAMPLE RUN OF THE TEACH MODULE AND THE EDITOR PROGRAM</td>
<td>50</td>
</tr>
<tr>
<td>APPENDIX B. LISTINGS OF THE TEACH MODULE AND THE EDITOR PROGRAM AND THE HELP FILES</td>
<td>58</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>89</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.0 Structure of the Editor Program</td>
<td>14</td>
</tr>
<tr>
<td>3.1 External and Internal View of the Data File</td>
<td>20</td>
</tr>
<tr>
<td>3.2 Master Directory</td>
<td>22</td>
</tr>
<tr>
<td>3.3 Actual Directory</td>
<td>23</td>
</tr>
<tr>
<td>3.4 Inverted Index by Subject Directory</td>
<td>25</td>
</tr>
<tr>
<td>3.5 Logical Design of the EDITOR Program</td>
<td>36</td>
</tr>
<tr>
<td>4.0 Structure of the TEACH Module</td>
<td>38</td>
</tr>
<tr>
<td>4.1 Logical Design of the TEACH Module</td>
<td>43</td>
</tr>
<tr>
<td>4.2 TEACH System's Block Diagram</td>
<td>44</td>
</tr>
</tbody>
</table>
CHAPTER I

INTRODUCTION

As the cost of hardware decreases, the use of computers to accomplish tasks previously done only by humans, becomes greatly advantageous. Computer Aided Instructional Systems are rapidly becoming a part of instructional systems in most universities, schools and even kindergartens. The availability of time sharing computers is one of the main factors supporting this rapid growth.

The other factor participating in this vast growth is the use of software techniques which allows individuals with no knowledge of computers to use computers. This thesis reviews the techniques used to implement such systems. Also, the design and the implementation of a Computer Aided Instructional System called TEACH is discussed in detail.

Chapter II deals with reviewing the relevant literature. Some of the computer aided instructional systems' techniques are also discussed in this chapter. TEACH is a computer aided instructional system written in FORTRAN, and it is implemented on a Digital Equipment Corporation (DEC) PDP-10 computer system at Western Michigan University. The emphasis of TEACH system is on the type of directory with which this system is equipped.

The TEACH system consists of a collection of data files, directories and an application program. The data files and the directories are constructed and maintained through an EDITOR program. Chapter III
contains a complete description of the EDITOR program, the directories and the data files. The logical design of the application program and its function and the logical design and implementation of the TEACH System in general, are included in Chapter IV.

The application program, which from now on will be referred to as TEACH module, is an interactive program interfacing with the data base (data files and directories). The TEACH module performs the teaching function of the system. Even though the design and the implementation of the TEACH System, as discussed in this thesis, are complete, there are possibilities for the expansion of the system.

Chapter V describes possible future expansions, limitations and maintenance of the TEACH System. A sample run of the TEACH module and the EDITOR program is included in Appendix A. Appendix B contains listings of the TEACH module and of the EDITOR program. As was mentioned earlier, the system is written in FORTRAN. FORTRAN was chosen because of the author's familiarity with it and its random access capabilities needed for the implementation of the system.
One major application of machines in education is educational television. In its early stages, educational television was often referred to as a "failure" and the reason was the lack of understanding of the medium. This characterization of educational television is probably not valid in today's world.

Isaac I. Bejar states that:

The computer has also had a mixed history in education. While the technology's potential was realized by some of the early innovators, the realities of the marketplace prevented the predicted boom of computers in education.

Because software developers were not sure of the market, they did not strive to produce the best possible product. This led, in turn, to poor acceptance of the software, which fulfilled the developers' own prophecy.

However, the effectiveness of CAI (Computer-Assisted Instruction) now seems established judging by the results of several projects including a large-scale field test being conducted in Los Angeles Public Schools by the Educational Testing Service for the National Institute of Education. On the market side, the cottage industry that has provided educational programs is now beginning to be supplemented by established publishers. Together, these trends suggest that at last technology is gaining a foothold in education.¹

Computers have been used in the area of education for some time now. The use of computers for instructional purposes has been by

elementary schools, middle schools, high schools and colleges. It has also been used by the armed forces personnel and by corporate executives. Computers have helped to fulfill different needs for instruction in all of the areas mentioned. An account of this statement is included later in this chapter.

Kearsley, Hillelsohn and Seidel reported that:

a recent survey conducted by the ACM Committee on Computing in Secondary/Elementary Schools found that of 974 school districts in the U.S., '90 per cent of these districts used computers; 74 per cent of this usage was instructional in nature. Instructional use of computers had increased from 13 per cent in 1970 to 74 per cent in 1980. Most of the programs being used involved a drill-and-practice strategy, were written in BASIC, and were acquired from outside sources.'

Two ways in which computers have been used are computer-assisted instruction (CAI) and computer-managed instruction (CMI). Kalisch states that, "CAI involves direct instruction by computer. CMI deals with instructional management and, for the most part, does not provide the instruction via computer." Kalisch also points out Mitzel's classification of CMI into three levels:

1. Test information tabulation systems;
2. Test scoring, information tabulation; and
3. Level 2 systems with real time interface between student and computer.

---


The descriptions of the levels indicate the use of a computer in each, but moreover underscore the vast differences CMI systems may have. The Level I system serves a clerical function. Level 2 additionally provides information to the student, although not via a computer terminal. Level 3 describes the most sophisticated present day level of CMI. Not only does the Level 3 provide the clerical and student feedback functions, but communicates directly to the student via computer terminal.

Kalisch also describes some CMI systems being used in different colleges and universities. Brigham Young University has a system for Computer-Assisted, on-line, University wide Testing (SCOUT). It is a Level 2 CMI system according to Mitzel's classification. Florida State University has a system similar to SCOUT. After using an on-line CMI testing for several years, they changed to off-line testing because of saturation of the computer system (CDC 6500) and because of economics.

Control Data Corporation has implemented CMI systems employing more of the CMI components discussed previously, than either SCOUT or Florida State's system. The system is part of the Control Data PLATO Computer system, most commonly used for CAI. The CMI system, abbreviated PCMI (PLATO CMI), does not require an author or instructor to select various options, input objectives and items, select strategies, and configure a CMI course.

The Air Force Human Resources Laboratory, in its Technical Training Division, has been involved with a CMI system as part of the Advanced Instructional System (AIS). This system is very similar to SCOUT and Florida State's CMI systems.

---

6 Ibid., p. 216.
Computer-Assisted Instruction (CAI) has also been diversely used. There has been some reluctance to CAI on the part of college instructors, for the fact that the quality of instruction is inferior and for the fear that computers would replace them. Byerly says that these fears have been proven as groundless as similar fears of printed books, television, programmed texts and audio-visual aids. . . . Recent reports from diverse sources indicate that the computer has not revolutionized any field, especially English.7

There are several uses of CAI in teaching English; one being that it can teach students how to read and write. For students in the early levels of learning English, the computer can provide new ways of presenting a new word. Today, many schools are using CAI as a supplement to instruction.

Byerly cites many colleges and universities using CAI systems: Scottsdale Community College in Arizona, Claremont Colleges in California, University of Notre Dame, Dartmouth College, Brown University, etc.8 These colleges and universities are using CAI systems as a supplement to teach English. The results have been good, but their development has been slow, because of resistance on the part of the instructors, fearing that CAI will substitute for or eliminate them. Another problem results from the fact that many instructors are not capable of programming a computer.

8Ibid., p. 284.
An area where CAI systems have been proven very effective is in aiding Spanish-speaking migrant students. Saracho describes a study that she did examining the effects of computer-assisted instruction on Spanish-speaking migrant children. This study involved two groups: a group that participated in the CAI designed by the Computer Curriculum Corporation, and a control group that did not. The students who used the CAI program achieved higher scores on the comprehensive tests of basic skills than the students who did not use the CAI system.

CAI systems have also been useful as an instructional tool for the deaf. Fox states that:

The Learning Technology Institute commenced work on this project in October 1977. The basic objective established was to conduct development of the computer controlled projection device and materials of instruction in a frame of reference which would permit early and widespread implementation in order to achieve the maximum benefit of this technology for vocational instruction of deaf students.

Computer-based education has also been used in prison schools. A system called PLATO Corrections Project (PCP) was developed and implemented in Illinois and Minnesota. This work, reports Siegel,

was carried out by the PCP staff at the Computer-based Education Research Laboratory (CERL), University of Illinois, under contract with the Illinois Department of Corrections School District #428.


Siegel reports that the system was used by 2,000 students and 120 staff in three Illinois prisons. The network was expanded to the Minnesota Department of Corrections. Siegel's overview of this project shows that the PCP system is feasible, attractive, and effective in correctional education.

In the past ten years, computers have become widely used in businesses and industries for many types of applications. Kearsley, Hillelsohn, and Seidel have made a survey on micro-computer-based training in business and industry. They contacted, by mail or telephone, training directors or managers of 160 major corporations, to find out the scope of micro-computer usage in the training domain. Fifty-six corporations responded. Fifty per cent indicated that they use computers for training and sixteen per cent indicated that they were investigating the possible use of computers for training.¹²

Kearsley, et al., believe that the use of computers for training purposes will become increasingly important. Past trends indicate that the use of computers for both training and educating will continue to increase. Computers are already being used in businesses, educational institutions, libraries and finally, in homes. Their use in education and the other areas will continue to further expand.

Bork states that:

Computer technology, over the next twenty-five years, will completely transform our educational system. While computers are widely talked about as a learning device, the amount of use of computers at the present

time for learning could be considered at the level of trivia. Gradually over the next quarter century this situation will change drastically, until the computer becomes the major delivery system at all levels of education.\textsuperscript{13}

2.1 Review of CAI Techniques

The use of computers for instructional purposes is described with various terms. The most popular ones are CAI (computer-assisted instruction or computer-aided instruction), and CMI (computer-managed instruction). However, the techniques such as: CBI (computer-based instruction), CAL (computer-aided learning), CAT (computer-assisted testing), etc., have also been employed in using computers as an instructional tool. There are obvious overlaps among these categories. Because of these overlaps, classification of a computerized instructional system into either of these categories will not be very distinct, but is still possible.

2.1.1 CMI (Computer-Managed Instruction)

A computerized instructional system classified into the CMI category refers to a system in which the computer is used as a monitor and as an information system for designing instructions generally produced in other ways. Usually a CMI system is equipped with a CAT (computer-assisted testing) system plus computer-managed reporting. Computer-assisted testing performs test generation and scoring in

which the tests are then analyzed and reports for the teacher will be
provided by the computer-managed reporting. Most CMI systems are
operated off-line in a batch type mode and the student does not
actually interact with the computer.\textsuperscript{14}

\textbf{2.1.2 CAI (Computer-Assisted Instruction)}

CAI System refers to a computerized instructional system in
which the student actually interacts with the computer. This inter­
action of the student with the computer can be of either a tutorial
form or a drill-and-practice type.\textsuperscript{15}

\textbf{2.1.2.1 The Tutorial Technique.} In this technique, the student
has considerable freedom in selecting the text to be studied. The
text is usually arranged in levels of increasing difficulty. The
student is evaluated and scored, then according to the analysis, the
system transfers control to the proper level. In such systems, the
student might ask the system for different approaches to a particular
problem or helpful hints for problem solving.

\textbf{2.1.2.2 The Drill-and-Practice Technique.} In this technique,
there are sequences of multiple choice questions arranged in frames.
The student is allowed to select one of the limited number of possible
responses. The responses are evaluated and according to the analysis,

\textsuperscript{14}T. E. Kieren, Research on Computers in Mathematics Education
(Columbus, Ohio: The Ohio State University Press, ERIC Information
Analysis Center for Science, Mathematics and Environmental Education,

\textsuperscript{15}Ibid., p. 2.
the system's logical flow of the lesson will alter. In the event of a correct response the system transfers control to a logically higher frame. In case of an incorrect answer, additional practice is provided.

2.1.3 Classification of Computerized Teaching Systems

Computer-assisted instructional systems can be classified as either "selective" or "generative" categories.

2.1.3.1 Selective Teaching System. "Selective" teaching systems usually consist of a large data-base where the contents are randomly accessible. In such systems, the entire process is pre-programmed. Every question is associated with a list of possible responses contained in the data-base. Most of the earlier work of CAI was done in a selective nature. The fact that the data-base should contain all possible responses to a question can be considered a drawback for selective systems. An obvious practical limitation of such systems would be the large capacity of the computer that is needed to store the entire data-base.

2.1.3.2 Generative Teaching System. "Generative" teaching systems are organized in a different way than the "selective" ones. As opposed to "selective" systems that are based on a pre-programmed data-base, "generative" teaching systems are usually equipped with a series of powerful algorithms to generate problems and answers. Such systems are best suited to the teaching of mathematically-oriented
subjects such as physical sciences, chemistry and logic. Within these subjects, there is a set of defined relations on which the algorithms can be constructed. As the questions and the possible responses are generated by the system itself, it is no longer necessary for the teacher to go through the process of anticipating and pre-programming such responses.

2.2 TEACH System

The technique used to implement the "TEACH" System is that of "tutorial" technique, because of the fact that the user (student) has considerable freedom in selecting the materials to be studied. However, the "TEACH" system can be classified under "SELECTIVE" computer-assisted instructional systems, since the system is interfaced with a data-base of learning materials. In this data-base, every question is associated with a list of possible answers, and the whole process is pre-programmed.
CHAPTER III

EDITOR PROGRAM

The purpose of the EDITOR program is to enable the user (teacher) to set up the data-base (data files and directories) which shall be accessed by the TEACH module. The data-base will contain the TEXT, EXAMPLE and EXERCISE files and directories in this files. The function of the EDITOR program is to construct any of these files through a simple dialog with the user (teacher) and "automatically" set up an inverted index by subject directory to the corresponding file. In many CAI systems, the directory, including the pointers to the records in the TEXT, EXAMPLE or EXERCISE file, has to be constructed "manually" by the teacher, through the use of an available TEXT EDITOR (for instance, TECHO or SOS).

The EDITOR program can be run by using the monitor command, .RUN EDITR2. Once the EDITOR program is activated, there are three sets of commands available: top level commands, second level commands, and third level commands. Figure 3.0 illustrates the structure of the EDITOR program.

3.1 Top Level Commands

The top level commands are EDIT and QUIT (EXIT).

3.1.1 EDIT Command

Upon invoking the EDIT command, the system will require the answer
Figure 3.0

Structure of the Editor Program
to the question: OLD or NEW? If the file (data file) to be edited already exists, then "OLD" is the right answer. If the intention is to create the file (data file) and edit it (or portions of it), then "NEW" is the proper response.

Next the system will ask the user to enter the name of the file to be edited. If the file is an old file, where there has been a name associated with it at the time of the creation, then in this case, the same name should be given to the system as the name given to the file. If a new file is being created, the user will have to provide the system with a new name.

The name of the file could be any legal file name consisting of, at the most, ten characters. The first six or less characters used as the name, is followed by a period and then followed by three characters as an extension. From this point on, the system is ready to accept the second level commands.

3.1.2 QUIT or EXIT Command

This command terminates the execution of the EDITOR program.

3.2 Second Level Commands

The second level commands are TEXT, EXAMPLE, EXERCISE and DONE. These commands can be used once the system is in the EDIT mode and after the name of the file to be edited has been provided to the system.
3.2.1 TEXT Command

The TEXT command enables the user to edit the text portion of the data file. When the TEXT command is issued, it allows the user to use the third level commands, which are the actual editing commands to edit the TEXT portion of the data file. The third level commands are discussed in detail in Section 3.3.

3.2.2 EXAMPLE Command

The EXAMPLE command permits the user to edit the EXAMPLE portion of the data file. Invoking this command enables the user to use the third level commands, which as mentioned before, are the actual editing commands to edit the EXAMPLE portion of the data file.

3.2.3 EXERCISE Command

This command enables the user to edit the EXERCISE portion of the data file. Once this command is issued, the third level commands can be used to edit the EXERCISE portion of the data file.

3.2.4 DONE Command

This command is used to transfer control from the second level to the top level.

3.3 Third Level Commands

The third level commands are the actual editing commands and they are used to edit any one of the three portions of the data
file: TEXT, EXAMPLE and EXERCISE. These commands are: "I", "D" and "P", followed by an integer number, "R" and "$".

3.3.1 "I" or Insert Command

This command is used to insert a new record in a file. Once this command is issued, the user will be asked to enter the subject of the TEXT, EXAMPLE or EXERCISE depending on which portion of the data file the user is working with. Next, the system requests the user to enter the TEXT, or the EXAMPLE, or the EXERCISE, again, depending on which portion of the data file is being edited.

The contents of each portion of the data file (TEXTS, EXAMPLES, EXERCISES), are classified according to the subjects that they include. There is a directory associated with each portion of the data file. The directory structures and the performance analysis of the system are discussed in later sections.

NOTE: From this point on, a TEXT, an EXAMPLE, or an EXERCISE is simply referred to as a record (data).

3.3.2 "D" or Delete Command

This command should be used if the intention of the user is to delete a record. The delete command should be given in the form of "D", followed by a record (data) number. A record number is an integer number. Even though this is a restricted requirement, at the same time, for such a structure, it is the only way of associating a unique key with each record.

A record (data) candidate for deletion is displayed by the system
in which the user is asked to either confirm or cancel the delete re-
quest. This is to make sure that the record to be deleted is the cor-
rect one. As soon as a record is deleted, the remaining records in
that portion of the data file (TEXT, EXAMPLE or EXERCISE portion)
are renumbered. For instance, if there are five records in a portion
(TEXT, EXAMPLE, EXERCISE) of the data file and record one is deleted,
the record which previously was numbered as record two is now numbered
as record one and so on. This does not require any record movements
since only the pointers to the records are adjusted and renumbered.

The reason that records in each portion of the data file are num-
bered is to associate a unique key with each one of them. A subject
cannot be considered as a key for a record since there might be sever-
al records regarding the same subject. In other words, a subject can-
ot be an identifying key for a record since several other records
might be identified by the same subject. For instance, if there are
two records as follows:

<table>
<thead>
<tr>
<th>Record #</th>
<th>Subject</th>
<th>Record Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>apple</td>
<td>some apples are red</td>
</tr>
<tr>
<td>2</td>
<td>apple</td>
<td>some apples are delicious</td>
</tr>
</tbody>
</table>

The subject, apple, cannot be considered as a primary key since both
records, 1 and 2, are identified by it. Therefore, a record number
can only be associated with one record so it can be used as the pri-
mary key to identify a record.

3.3.3 "P" or Print Command

This command is used to print a record. The letter "P", followed
by a record number, causes the required record to be printed on the screen.

3.3.4 "R" or Report Command

If the user desires to see the contents of a portion of the data file which is being edited, this command should be used. The command, "R", generates a report on that portion of the data file being edited. This report contains all records and their corresponding subjects in that portion of the data file.

3.3.5 "$" or Stop Editing Command

The command, "$", is the stop editing command. Once this command is issued, the system transfers control to the second level.

3.4 Structure of the Data File

There are three files constituting the data file. These three files are the TEXT, EXAMPLE, and EXERCISE files. At the external level (the user's view), all three files are considered as portions of a single data file, which has been given a single file name.

However, internally (the system's view), the three files are distinct and each has a name associated with it. By considering all three files as portions of a single data file, the user can access any of the three files with one file name (name of the data file), and can specify which portion of the data file he or she wants to access. Otherwise, the user has to deal with three different file names for the TEXT, EXAMPLE and EXERCISE files. The advantage
resulting from such a structure is to enable the user to access all three files by a single file name rather than three different file names. Figure 3.1 demonstrates the external and the internal view of the data file.

![Diagram of data file structure](image)

Figure 3.1

External and the Internal View of the Data File

3.5 Directory

Every portion (TEXT, EXAMPLE, EXERCISE) of the data file is equipped with an inverted index by subject directory. This directory is composed of a master and an actual directory. Directories are useful when there is a considerable amount of data to be processed. They save time by totally or partially eliminating sequential operations.

The biggest drawback of inverted directories is the slowness of the insert and delete operation. However, the greatest advantage of
such directories is the fast record retrieval capability. The disk space occupied by the directories is directly dependent on the amount of data.

As was mentioned earlier, directories can be absolutely useless if there is not a "large" amount of data available. Usually, for a particular application, analyses and calculations are performed to find out whether there is a need for a directory and what type would serve the situation the best.

The inverted list directory is best suited for the TEACH system because of the fact that a user (student) who is using the system mainly performs record retrieval. For instance, a request by the user (student) to see a particular text, example or exercise, involves a record retrieval. Since inverted list directories are capable of fast record retrieval, they serve the situation the best. The performance analyses and measurements of such an inverted list directory, which is employed in TEACH system, is discussed in detail in later sections.

3.5.1 Master Directory

The master directory groups the subjects, of which the records are about, in alphabetical order, assuming that every possible subject starts with a letter from the twenty-six letters of the alphabet. This scheme groups all the subjects together starting with a particular letter of the alphabet, by actually grouping the pointers to these subjects. For instance, pointers to subjects; ZAP, ZIP, and ZOP, are all grouped together under the subjects starting with the letter Z.
The index to the master directory are the twenty-six letters of the alphabet. Figure 3.2 demonstrates a master directory of this kind.

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>. .</td>
<td>30, 41, . . .</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>. .</td>
<td>32, 43, . . .</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>. .</td>
<td>42, 62, . . .</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>. .</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Z</td>
<td>. .</td>
<td>31, 61, . . .</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.2
Master Directory

Each integer number in the right side of Figure 3.2 corresponds to a portion or to a unit of the file called block. Blocks, one to twenty-six, contain letters A through Z. Block numbers associated with a letter, indicate where in the actual directory the subjects starting with the corresponding letter can be found. For example, block numbers 30 and 41 in front of the letter A, indicate that blocks 30 and 41 in the actual directory contain subjects starting with the letter A. In order to retrieve a record given a subject, the master directory is accessed first. In the next section, the actual directory is discussed.
3.5.2 Actual Directory

The actual directory contains the actual pile file pointers. There are pile file pointers associated with every subject in the actual directory. These pointers correspond to locations (blocks) in the pile file (Data Bank), where the actual records have been stored. Figure 3.3 illustrates an actual directory of this type.

| 30 And Gates                      | 1, 2, 3 |
| 31 Zap                           | 4, 5    |
| 32 Boolean Functions              | 7, 8    |
| 41 Analog to Digital Conversion   | 6, 9    |
| 42 Computer Software              | 13, 14  |
| 43 Boolean Expressions            | 10, 12  |
| 60                                |        |
| 61 Zip                            | 11, 15  |
| 62 Computer Hardware              | 16, 17  |

Figure 3.3  
Actual Directory
### 3.5.3 Combination of the Master and the Actual Directory

The master and the actual directories are combined to produce the inverted index by subject directory. The two directories are parts of the same file. Figure 3.4 shows an inverted index by subject directory of this kind. An example of employing the combination of the master and the actual directory toward finding a record would be a search for a record corresponding to the subject "computer hardware". Considering Figure 3.4, a binary search is performed on the master directory to find the starting letter of the subject which, in this case, is "C".

Once letter "C" is found in the master directory, the block numbers associated with it are recorded. In this case, the block numbers are 42 and 62. Blocks 42 and 62 of the actual directory are checked to see which or if any of them contain the subject "computer hardware". In this particular example, block number 62 of the actual directory contains the subject "computer hardware". The block numbers associated with the subject "computer hardware" in the actual directory are the locations or the blocks in the pile file where the actual records corresponding to the subject "computer hardware" have been stored. In this case, blocks 16 and 17, in the pile file, contain records corresponding to the subject "computer hardware".
FIGURE 3.4

Inverted Index by Subject Directory
3.5.4 Performance Analysis of the Inverted Index by Subject Directory

3.5.4.1 Data Retrieval. Given a subject, the first letter of this subject is used to search the master directory. A binary search is performed on the master directory to find the matching letter. Once the matching letter is found, the pointers or block numbers associated with it correspond to the blocks in the actual directory containing the subjects starting with this letter.

The mentioned blocks in the actual directory are searched to find the given subject. Upon finding the given subject in the actual directory, the block numbers associated with it indicate where in the pile file (corresponding data file) records regarding this subject are to be found. Since there are twenty-six letters available in the English alphabet, the probability of any of these twenty-six letters occurring is 1/26. For instance, the probability of the occurrence of the letter "A" is 1/26. At the same time, the probability of occurrences of a subject starting with any of these twenty-six letters is 1/26.

As it is known, the computing time for a binary search is:

<table>
<thead>
<tr>
<th>Successful Search</th>
<th>Unsuccessful Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Θ (1), Θ (log₂n), Θ (log₂n)</td>
<td>Θ (log₂n)</td>
</tr>
<tr>
<td>best average worst</td>
<td>best average worst</td>
</tr>
</tbody>
</table>

Since we are concerned with a successful search in the master directory (assuming that every possible subject starts with one of the 26 letters of the alphabet), on the average, we have log₂n or log₂26.
letter comparison. The average number of accesses is computed to approximately \(4.7 \ (\log_2 26 \approx 4.7)\) or at most to 5 accesses.

If there are \(X\) pointers or block numbers associated with the letter of the alphabet found in the master directory, there will be \(X\) subjects in the actual directory starting with this letter. To find the given subject in the worst case, all these \(X\) blocks in the actual directory have to be searched. In other words, in the worst case, \(X\) accesses should be made to the actual directory. If the subject exists in the actual directory, one more access is made to the pile file to retrieve the record corresponding to this subject.

Maximum number of accesses needed to retrieve a record corresponding to a subject, given the subject and provided the subject exists in the actual directory, would be:

\[
\begin{align*}
\text{Master Directory} & \quad 5 \\
\text{Actual Directory} & \quad X \\
\text{Pile File} & \quad 1 \\
\text{Total} & \quad X + 6 \text{ accesses}
\end{align*}
\]

Where \(X\) is the number of blocks in the actual directory, containing subjects starting with the same letter of the alphabet as the given subject.

If the subject does not exist in the actual directory, maximum number of accesses made would be:

\[
\begin{align*}
\text{Master Directory} & \quad 5 \\
\text{Actual Directory} & \quad X \\
\text{Total} & \quad X + 5
\end{align*}
\]
Thus, a message indicating the absence of the given subject in the actual directory would be issued to the user. For further refinement of the analysis, a case with uniform probability of distribution is considered.

In assumption, there are 520 different subjects equally distributed among the twenty-six letters of the alphabet. Therefore, there will be twenty subjects starting with each one of the twenty-six letters of the alphabet. Since we have assumed that there are twenty subjects starting with the letter A, twenty with the letter B, and so on, the actual directory will have twenty subjects for each one of the twenty-six letters of the alphabet. So, at the most, twenty accesses are made in the actual directory to find the given subject. Finally, one access is made in the pile file to retrieve the record. The number of accesses made overall is:

- Master Directory 5
- Actual Directory 20
- Pile File 1

26 accesses

Consider if there were no directory available, a sequential search would have to be performed to find a subject and to retrieve a record. Since there are 520 subjects, a sequential search in the worst case would take 520 accesses. Comparing 520 accesses to twenty-six, there is a saving of 494 accesses. As the number of subjects increases, the saving will be greater.

3.5.4.2 Adding Data. To add a record to the pile file, the
master directory is searched first. A binary search is performed on the master directory to find the starting letter of the given subject. Once this letter is found, the pointers or block numbers associated with it are recorded.

These pointers, or block numbers, are checked in the actual directory to see if the subject already exists in the file. If the subject is already in the actual directory, another pointer will be added to the list of its pointers which will correspond to the new record in the pile file. If the subject is not already in the actual directory, first, there will be a pointer, or block number (PI), added to the master directory in the appropriate list of pointers or list that corresponds to the letter of the alphabet with which the subject begins. Secondly, the subject will be written in the proper block (PI) in the actual directory. Finally, there will be a pointer associated with this subject which shall correspond to the record regarding this subject in the pile file.

After updating the directory, the new record is added to the pile file. In general, the number of accesses in the master directory, the actual directory, and the pile file would be:

1. If the subject already exists in the actual directory, then the binary search in the master directory, to find the starting letter of the subject, takes a maximum of $5$ accesses ($\log_2 26 \approx 5$). To find the subject in the actual directory, a maximum of $X$ accesses would be needed assuming there are $X$ subjects starting with the same letter of the alphabet as the given subject.
There will be one more access needed to rewrite the subject with the list of pointers, including the new pointer corresponding to the new record, which is to be added in the pile file. In the pile file, one access is made to read (find) the available block which will contain the record to be added. The available block is accessed to determine the next available block (the link of the available block). One access will be made to write the new available block and one access will be made to write the record to be added. Therefore, the overall number of accesses is:

- Maximum number of accesses in the master directory: 5
- Maximum number of accesses in the actual directory: \(X + 1\)
- Maximum number of accesses in the pile file: 4

Total: \(X + 10\)

2. If the subject is not in the actual directory, then initially the number of accesses in the master directory would be \(\log_2 26 \approx 5\). To find out if the subject does not exist in the actual directory, \(X\) accesses are needed, assuming there are \(X\) subjects starting with the same letter of the alphabet as the given subject. Going back to the master directory and adding a pointer (PI) in the list of pointers corresponding to the starting letter of the subject, takes one more access.

This pointer corresponds to the new subject which will be added in the actual directory. Going back to the actual directory and the addition of the new subject along with a pointer, corresponding to where in the pile file the record regarding this new subject is going.
to be, takes one more access. The new subject and the associated pointer are added to the actual directory in the block P1.

In the pile file, we will have four accesses. The first access is to read where the available block is. Next, the available block is accessed to determine the next available block (the like of the available block). The third access is to write the new available block. The fourth access is to write the new record. The overall number of accesses are:

- Maximum number of accesses in the master directory: 6
- Maximum number of accesses in the actual directory: \( X + 1 \)
- Maximum number of accesses in the pile file: 4

Total: \( X + 11 \)

For the sake of demonstration, the same case as in 3.5.4.2, with uniform probability of distribution is considered. With the same assumption of having 520 different subjects equally distributed among the twenty-six letters of the alphabet (twenty subjects begin with each one of the twenty-six letters of the alphabet), the number of accesses in the master directory, the actual directory and the pile file would be as follows:

a. If the subject already exists in the actual directory, then the maximum number of accesses in the master directory would be \( \log_2 26 \approx 5 \). To find the subject in the actual directory, a maximum of twenty accesses would be needed, since there are twenty different subjects starting with each one of the twenty-six letters of the alphabet. There will be one more access.
needed to rewrite the subject with the list of pointers including the new pointer corresponding to the new record that is to be added in the pile file.

In the pile file, one access is made to read (find) the available block which will contain the record to be added. The available block is accessed to determine the next available block (the link of the available block). One access will be made to write the new available block and one access will be made to write the record to be added.

Maximum number of accesses in the master directory 5
Maximum number of accesses in the actual directory 21
Maximum number of accesses in the pile file 4

Total 30

b. If the subject is not in the actual directory, initially the number of accesses in the master directory would be \( \log_226 \approx 5 \).
To find out that the subject does not exist in the actual directory, twenty accesses are needed (since there are twenty different subjects starting with each one of the twenty-six letters of the alphabet).

Going back to the master directory and adding a pointer (PI) in the list of pointers corresponding to the starting letter of the subject, takes one more access. This pointer corresponds to the new subject which will be added in the actual directory.

Going back to the actual directory and the addition of the new subject along with a pointer, corresponding to where in the pile
file the record regarding the new subject is going to be, takes one more access. The new subject and the associated pointer are added to the actual directory in the block P1.

In the pile file, we will have four accesses. The first access is to read where the available block is. Next, the available block is accessed to determine the next available block (the link of the available block). The third access is to write the new available block. The fourth access is to write the new record.

Maximum number of accesses in the master directory 6
Maximum number of accesses in the actual directory 21
Maximum number of accesses in the pile file 4
Total 31

3.5.4.3 Deleting Data. To delete a record from the pile file, the master directory is searched first. A binary search is performed on the master directory to find the starting letter of the subject, corresponding to the record which is to be deleted. Once this letter is found, the pointer or block numbers associated with it are recorded.

These blocks, in the actual directory, are searched to find the given subject corresponding to the record that is to be deleted. Upon finding the required subject, the particular pointer associated with it, which corresponds to the record in the pile file that is to be deleted, is changed to zero. Finally, the only required change in the pile file is updating the available block.

In general, the maximum number of accesses in the master directory
would be $\log_2 26 \approx 5$. A maximum of $X$ accesses is needed in the actual directory to find the required subject. One more access is needed in the actual directory to rewrite the subject with the proper pointer changed to zero. In the pile file, all we have to do is to update the available block.

One access is made to read the current available location (Al). The second access will be made to the block where the record to be deleted is stored, and the block number associated with the current available location (Al) is written as a link in this block. The third access is made to write the new available location which previously contained the record to be deleted.

The overall number of accesses in general is:

- In the master directory 5
- In the actual directory $X + 1$
- In the pile file 3

Total $X + 9$

Again, for the sake of illustration, a case with uniform probability of distribution is considered. Assuming that there are 520 different subjects, equally distributed among the twenty-six letters of the alphabet (twenty different subjects begin with every one of the twenty-six letters of the alphabet, the maximum number of accesses in the master directory would be $\log_2 26 \approx 5$. A maximum of twenty accesses is needed in the actual directory to find the required subject. One more access is needed to write the subject with the proper pointer changed to zero. In the pile file, all we have to do is to update the available location.
One access is made to read the current available location (A1). The second access will be made to the block where the record to be deleted is stored; and the block number associated with the current available location (A1) is written as a link, in this block. The third access is made to write the new available location which previously contained the record to be deleted so that the overall number of access would be:

- In the master directory 5
- In the actual directory 21
- In the pile file 3
- Total 29

3.6 Logical Design of the EDITOR Program

The structure of the EDITOR program is as follows in Figure 3.5.
Figure 3.5

Logical Design of the EDITOR Program
CHAPTER IV

TEACH MODULE

The TEACH module is the module performing the teaching function of the system. This module interfaces with the data base constructed by the EDITOR program. Through the use of the TEACH module, materials contained in the data base are randomly accessible by the user (student). The technique employed by the module to accomplish teaching, is that of "tutorial" technique, where the user (student) has considerable freedom in selecting the materials to be studied.

This module can be run by using the monitor command: .RUN TEACH. Once the module is activated, there are five commands that can be used. These commands and further information about the module are discussed in the following sections. Figure 4.0 illustrates the top down structure of the TEACH module.

4.1 List of Commands

The available commands are: LEARN, EXAMPLE, EXERCISE, QUIT (EXIT), AND HELP.

4.1.1 LEARN Command

This command is given if the user's purpose is to learn about a subject. Once this command is issued, the user is asked to enter the subject which is to be learned. If there are several texts available regarding the specified subject, one is randomly selected and displayed.
Figure 4.0

Structure of the TEACH Module
If there is only one text regarding the subject, then this text is selected and displayed.

At this time, the user will have the option of learning more about the same subject. If there are more texts available regarding this same subject, another one will randomly be selected and displayed. If there are no other texts available, the request will be denied and a message will be issued. If the user is not interested in learning further about the same subject, or if there are no more texts available about the specified subject, then the system transfers control to the top level where any of the five commands (LEARN, EXAMPLE, EXERCISE, QUIT [or EXIT], and HELP) can be used again.

4.1.2 EXAMPLE Command

The EXAMPLE command is used if the user's intention is to see some examples of a subject. Once the command EXAMPLE is issued, the user is asked to enter the subject of the example. If there is only one example available about the specified subject, it will be displayed.

In case there is more than one example available, one is randomly selected and displayed. At this point, the user will be given the option of seeing more examples about the same subject. If seeing more examples about the same subject is not of the user's interest, or if there are no more examples available regarding the specified subject, the system will transfer control to the top level wherein any of the five commands can be given again.
4.1.3 EXERCISE Command

This command is issued when the user wishes to see and to try some exercises regarding a subject. Upon invoking this command, the user is asked to associate a subject with the exercise. If there is only one exercise available regarding the specified subject, it will be displayed. If there is more than one exercise that exists, then one is randomly selected and displayed. Once an exercise is displayed, the user will have to provide an answer (or a solution) to it. This answer, or solution, is checked by the system for correction and the user will be informed of whether the answer given was right or wrong.

If the answer provided by the user is a wrong answer, the right answer will be displayed. At this point, the user will be given the option of trying more exercises with the same subject. In case the user does not wish to see more exercises regarding the same subject, or in case of unavailability of more exercises, the system goes back to the top level where, again, the five commands (LEARN, EXAMPLE, EXERCISE, QUIT [or EXIT], and HELP) can be used.

4.1.4 QUIT or EXIT Command

This command terminates the execution of the program.

4.1.5 HELP Command

This command causes the help file to be displayed. The help file contains information about the TEACH module and how to run this module.
4.2 Random Selection of a Record  
(TEXT, EXAMPLE, or EXERCISE)

If there are several records (TEXTS, EXAMPLES or EXERCISES) regarding a subject specified by the user, one is randomly selected. This is done by the aid of a random number generator. The range of the number to be randomly selected is given to the random number generator and a number in this range is selected.

The operation of this random number generator is based on the PDP-10's built-in clock. If a random number generated corresponds to a record that has already been selected, another random number is generated. This operation is continued until a random number is generated which will correspond to a record that has not yet been selected. To show how random numbers are related to records and how a random number generated will correspond to a record, the following example is considered: Assume there are five records available regarding the subject "NAND GATE." Once the subject is specified by the user, the system performs a search to find all records corresponding to this subject. Upon finding these five records, they are numbered one through five by the system.

Thus, the range given to the random number generator would be one to five. If number three is the random number generated, then the third record of the five will be selected. If the record three has already been selected, then another number in the range one of five is selected.

This search is guaranteed to halt because of the fact that if all the records corresponding to a subject have been selected, then
the request for a random selection of a record corresponding to this subject is denied. In other words, the request for a random selection of a record is not carried out until the availability of a record is assured.

### 4.3 Logical Design of the TEACH Module

The TEACH module is an application program that can be interfaced with a data base (data files and directories) constructed by the EDITOR program. This program is an interactive program and is general enough to be applied to any data base regarding any subject in any area, as long as the data base is constructed by the EDITOR program. The TEACH module is general since its application is not only limited to the data base considered in this thesis. As was mentioned earlier, this module can be interfaced with any data base constructed by the EDITOR program, no matter what the subject is.

In this thesis, TEACH module has been interfaced with a data base concerning the subject: Introduction to the SNOBOL4 Programming Language. However, if the EDITOR program is applied towards constructing a data base concerning Chemistry, the TEACH module can still be interfaced with this data base to teach chemistry.

Figure 4.1 shows the general logical design of the TEACH module.

---

16 The EDITOR Program is a general program and can be applied to construct a data base regarding any desired subject in any area. See Chapter III, EDITOR Program.
Figure 4.1

Logical Design of the TEACH Module
4.4 Logical Design of the TEACH System in General

The combination of the EDITOR program, data base (data files and directories), and the TEACH module constitute the TEACH System. The whole system can be viewed as a small data base management system. Figure 4.2 demonstrates the TEACH System's block diagram.

The user (student) can gain access to the data base only through the TEACH module. The EDITOR program is the module maintaining the data base, thus, the user will have to generate the data base through the use of the EDITOR program.
CHAPTER V

POSSIBLE EXPANSIONS, LIMITATIONS AND MAINTENANCE OF THE TEACH SYSTEM

The design and implementation of the TEACH System (EDITOR program and TEACH module) is complete, as described in Chapter III and IV. However, there are several possibilities for expanding the system. This chapter deals with the possibilities for future expansions, limitations and maintenance of the TEACH System.

5.1 Possible Future Expansions

Graphic capabilities and Artificial Intelligence are two of the features that can be added to the TEACH System. Another possibility for the expansion of the system would be the extension of the EDITOR program's command set.

5.1.1 Graphic Capabilities

Depending on the application of the TEACH System, the addition of the graphic feature would probably be the most useful expansion because of the fact that in many areas of science, a picture or a diagram can be more useful and understandable than written explanations. For example, the application of the TEACH System towards a course in Computer Graphics or Electronics, which mostly deals with diagrams, would certainly require the graphic feature.

This feature could be added to the system by the addition of...
proper routines or possibly by interfacing the TEACH System with a graphic package. It should be mentioned that once the system is equipped with such a feature the use of a graphic terminal would be required.

5.1.2 Artificial Intelligence

The TEACH System can become a "smart" system through the use of Artificial Intelligence techniques. As it was mentioned in earlier chapters, records (TEXTS, EXAMPLES, and EXERCISES) are classified according to the given subjects. In other words, the subject is specified first, and then the record regarding the subject.

Specifying the subject can be eliminated if the system itself could look for keywords in the records and find the subject which they include. An Artificial Intelligence technique that can be employed in this case would require providing the TEACH System with a list of all possible keywords concerning the area to which the system is being applied. For instance, in this thesis, the TEACH System is applied to a course in SNOBOL4 Programming Language, thus, a list of possible keywords which have to be provided to the system would be LEN, TOB, RTAB, QUICKSCAN MODE, FULLSCAN MODE, ARB, IDENT, etc.

Therefore, if there is a record stated as: "IDENT is a function of two arguments which may be of any data type," this technique performs a search on the record, recognizes the keyword "IDENT", and classifies the stated record under the subject IDENT.

It is possible that a query fails simply because of the wrong ordering of the subjects in the query. For example, any of the
following:

"IDENT and DIFFER"
"DIFFER and IDENT"
"IDENT, DIFFER"
"DIFFER, IDENT"

should result in a successful directory search and should enable the user (student) to retrieve information on "IDENT and DIFFER". However, only one of the above would result in a successful directory search, and that is the one which was given as the subject at the time of the construction of the directory.

For instance, assuming "IDENT and DIFFER" were given as the subject corresponding to record $X$, at the time of construction of the data base, therefore, if a user wishes to retrieve the record $X$, the subject specified to the system should be "IDENT and DIFFER". In other words, neither "DIFFER and IDENT", "IDENT, DIFFER", nor "DIFFER, IDENT" would result in retrieval of record $X$.

The elimination of the problem is another possibility for expanding the TEACH System through the use of Artificial Intelligence techniques. An Artificial Intelligence technique that can be employed to overcome this problem would involve reordering the words contained in the subjects and searching the directory with different combinations of the words constituting the subject.

The TEACH module can be extended to perform additional tasks such as: testing the users (students), determining the areas where they are weak, and then providing additional instructions in these areas.
5.1.3 Extension of the EDITOR Program's Command Set

The command set of the EDITOR program can be extended to perform additional tasks, such as: alteration of an existing record, and replacing an existing record by another. Even though such extensions are not closely related to the TEACH System, they can be considered as some possible future extensions to the system.

5.2 Limitations

5.2.1 Limitation on the Number of Characters

The maximum number of characters allowed in a record is 999. This limit can be increased by linking additional blocks to the blocks containing the record. The other limit concerns the number of characters allowed in a subject. A subject corresponding to a record could be the combination of several subjects. In this case, the subjects forming the mentioned subjects which are separated by semicolons, corresponding to a record or records, can at most be 199 characters long.

The two limits 999 and 199 characters are suitable for most applications of the TEACH System. As far as the number of the subjects starting with a letter of the alphabet and the number of records corresponding to a subject concerned, a limit is not of importance. Linking techniques have been used to overcome these limitations.
5.2.2 Disk Space Limitation

The other limitation concerns the disk space. The amount of disk space associated with the data base (directories and data files), increases as the amount of data increases. However, this limitation is always eliminated to an extent by allocation of more disk space to the area where the TEACH System is residing. The allocation of additional disk space can be obtained from the computing center.

5.2.3 Graphic Limitation

Finally, displaying diagrams could also be considered as another limitation. Since the TEACH System is not equipped with a graphic package, neither built in nor interfaced, diagrams have to be shown by using the available terminal symbols, like dot, dash, etc.

5.3 Maintenance

As far as the maintenance is concerned, the data base (data files and directories) is always maintained, since the EDITOR program is the module constructing it. There are two help files being maintained by the system on disk. One concerns the EDITOR program and the other is for the TEACH module. These help files contain detailed explanations of the EDITOR program's commands and information about the TEACH module and its commands. They are residing on disk under FOR21. DAT, and FOR22. DAT. They can be maintained and updated through the use of available TEXT editors such as: SOS or TECHO.
RUN TEACH

PLEASE ENTER THE FILE NAME: SNOBOL.RAN

IF YOU NEED HELP TYPE HELP !!

WAITING FOR YOUR COMMAND: HELP

TOP LEVEL COMMANDS ARE LEARN, EXAMPLE, EXERCISE, QUIT (EXIT).

IF LEARNING ABOUT A SUBJECT IS THE PURPOSE, COMMAND LEARN SHOULD BE

<table>
<thead>
<tr>
<th>TYPES OF STATEMENTS</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBJECT PATTERN</td>
<td>Pattern matching</td>
</tr>
<tr>
<td>SUBJECT REPLACEMENT</td>
<td>Replacement statement</td>
</tr>
</tbody>
</table>

THE OPERATION OF EXAMINING STRINGS FOR THE OCCURRENCE OF SPECIFIED SUBSTRINGS (I.E. PATTERN MATCHING) IS FUNDAMENTAL TO THE SNOBOL LANGUAGE. PATTERN MATCHING CAN BE SPECIFIED IN TWO TYPES OF STATEMENTS: (1) THE PATTERN MATCHING STATEMENT, AND (2) THE REPLACEMENT STATEMENT. THE PATTERN MATCHING STATEMENT HAS THE FORM: SUBJECT PATTERN, WHERE THE TWO FIELDS ARE SEPARATED BY AT LEAST ONE BLANK.

DO YOU WISH TO SEE MORE TEXT REGARDING THIS SUBJECT(S) (Y N)? Y

SORRY ALL TEXT REGARDING THIS SUBJECT(S) HAVE ALREADY BEEN SELECTED!

WAITING FOR YOUR COMMAND: LEARN

ENTER THE SUBJECT(S): B R E A K $
BREAK IS PRIMITIVE FUNCTION WHOSE VALUE IS PATTERN STRUCTURES THAT MATCHES RUNS OF CHARACTERS.

DO YOU WISH TO SEE MORE TEXT REGARDING THIS SUBJECT(S) (Y N)? Y
SORRY ALL TEXT REGARDING THIS SUBJECT(S) HAVE ALREADY BEEN SELECTED!
WAITING FOR YOUR COMMAND » LEARN

ENTER THE SUBJECT(S) » EVAL

EVAL IS A PRIMITIVE FUNCTION WHOSE ARGUMENT MUST BE AN UNEVALUATED EXPRESSION, STRING, INTEGER, OR REAL. IF THE ARGUMENT IS AN UNEVALUATED EXPRESSION, THE EXPRESSION IS EVALUATED TO OBTAIN THE VALUE OF EVAL. IF THE ARGUMENT IS A STRING, THE VALUE OF EVAL IS THE VALUE OF THE EXPRESSION REPRESENTED BY THE STRING. INTEGER AND REAL ARGUMENTS ARE SIMPLY RETURNED AS VALUE WITHOUT MODIFICATION.

DO YOU WISH TO SEE MORE TEXT REGARDING THIS SUBJECT(S) (Y N)? N
WAITING FOR YOUR COMMAND » EXAMPLE

ENTER THE SUBJECT(S) » PATTERN MATCHING STATEMENTS

IF TRADE = 'PROGRAMMER', THE STATEMENT: TRADE 'GRAM' EXAMINES THE VALUE OF TRADE FOR AN OCCURRENCE OF GRAM.

DO YOU WISH TO SEE MORE EXAMPLE REGARDING THIS SUBJECT(S) (Y N)? N
WAITING FOR YOUR COMMAND » EXAMPLE

ENTER THE SUBJECT(S) » EVAL

IN THE STATEMENTS E = '5 + 6', SUM = EVAL(E) THE VALUE OF SUM WOULD BE 11.

DO YOU WISH TO SEE MORE EXAMPLE REGARDING THIS SUBJECT(S) (Y N)? N
WAITING FOR YOUR COMMAND » EXAMPLE

ENTER THE SUBJECT(S) » BREAK

MAKE SURE SUBJECT(S) OR SUBJECTS SEPARATED WITH A ;) IS 199 OR LESS CHARACTERS LONG AND ENDS WITH A $ ! ! !

ENTER THE SUBJECT(S) » BREAK
NON-EXISTING SUBJECT(S) ! ! !
WAITING FOR YOUR COMMAND » EXERCISE
PLEASE TYPE IN THE CORRECT COMMAND !!!
WAITING FOR YOUR COMMAND>>EXERCISE
ENTER THE SUBJECT(S)>>EVAL$
WHAT DOES THE FOLLOWING OUTPUT STATEMENT PRODUCE? E = '5+6' OUTPUT
= EVAL(E) ? PLEASE SPECIFY YOUR ANSWER IN NUMERIC FORM.
ENTER YOUR ANSWER >>11$
CORRECT !
DO YOU WISH TO SEE MORE EXERCISE REGARDING THIS SUBJECT(S) (Y N)?Y
WHAT WOULD BE THE VALUE FOR RESULT CONSIDERING THE FOLLOWING: E = '5 + 3
- 6' *RESULT = EVAL(E) ?
ENTER YOUR ANSWER >>3
MAKE SURE ANSWER IS 99 OR LESS CHARACTERS LONG AND
ENDS WITH A % SIGN !!
ENTER YOUR ANSWER >>3%
WRONG ! RIGHT ANSWER IS: 2
DO YOU WISH TO SEE MORE EXERCISE REGARDING THIS SUBJECT(S) (Y N)?N
WAITING FOR YOUR COMMAND>>QUIT
End of Execution
CPU time 4.12 Elapsed time 9:12.56
EXIT

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

.RUN EDITR2
IF YOU NEED HELP USING THIS TEXT-EDITOR TYPE HELP!!
WAITING FOR YOUR COMMAND>>HELP
TOP LEVEL COMMANDS ARE EDIT AND QUIT(EXIT).
IF THE PROGRAM IS INSTRUCTED WITH AN EDIT COMMAND THE QUESTION OLD OR
NEW IS ASKED. IF THE FILE TO BE EDITED HAS ALREADY BEEN CREATED, THIS
FILE IS CONSIDERED AN OLD FILE AND IN SUCH CASE OLD SHOULD BE TYPED
IN.

TO CONTINUE TYPE C; OR ELSE TYPE E >>C
IF THE INTENTION IS TO CREATE A NEW FILE AND EDIT IT, THEN NEW SHOULD
BE TYPED IN.
IN EITHER OF THE ABOVE CASES YOU ARE ASKED TO PROVIDE THE NAME OF
THE FILE TO BE EDITED, ONCE THE SYSTEM IS IN EDIT MODE, COMMANDS
AVAILABLE ARE TEXT, EXAMPLE, EXERCISE, DONE.

TO CONTINUE TYPE C; OR ELSE TYPE E >>C
TEXT COMMAND ENABLES ONE TO EDIT THE TEXT PORTION OF THE FILE.
EXAMPLE COMMAND ENABLES ONE TO EDIT THE EXAMPLE PORTION.
EXERCISE COMMAND MAKES THE EDITING OF THE EXERCISE PORTION OF THE
FILE POSSIBLE AND DONE COMMAND CAUSES THE SYSTEM TO TRANSFER CONTROL
TO THE TOP LEVEL.

TO CONTINUE TYPE C; OR ELSE TYPE E >>E
WAITING FOR YOUR COMMAND >> EDIT

OLD OR NEW OLD
ENTER THE OLD FILE NAME >> SN Q O B O L . R A N
EDITING OLD FILE NAME: SN Q O B O L . R A N
WAITING FOR YOUR COMMAND (TEXT, EXAMPLE, EXERCISE, DONE) >> TEXT
WAITING FOR YOUR COMMAND (I, DN, FN, R,$) >> I
ENTER SUBJECT(S) >> SAMPLE$
ENTER TEXT  >> THIS IS A TEXT SAMPLE$
WAITING FOR YOUR COMMAND (I, DN, FN, R,$) >> P 3 0

SUBJECT: EVAL

TEXT : EVAL IS A PRIMITIVE FUNCTION WHOSE ARGUMENT MUST BE AN UNEVLAU-
ATED EXPRESSION, STRING, INTEGER, OR REAL. IF THE ARGUMENT IS AN UNEVLAU-
ATED EXPRESSION, THE EXPRESSION IS EVALUATED TO OBTAIN THE VALUE OF EVAL. IF TH
E ARGUMENT IS A STRING, THE VALUE OF EVAL IS THE VALUE OF THE EXPRESSION
REPRESENTED BY THE STRING. INTEGER AND REAL ARGUMENTS ARE SIMPLY RETURNED
AS VALUES WITHOUT MODIFICATION.

WAITING FOR YOUR COMMAND (I, DN, FN, R,$) >> P 3 1
SUBJECT: INTEGER

TEXT: IT IS FREQUENTLY DESIRABLE TO TEST WHETHER THE VALUE OF A VARIABLE IS AN INTEGER. THE PREDICATE TEST INTEGER(X) SUCCEEDS IF THE VALUE OF X IS AN INTEGER AND FAILS OTHERWISE.

WAITING FOR YOUR COMMAND (I, DN, FN, R, $) » P32

SUBJECT: SAMPLE

TEXT: THIS IS A TEXT SAMPLE

WAITING FOR YOUR COMMAND (I, DN, FN, R, $) » D32

PLEASE CONFIRM BY TYPING KILL IF THE RECORD TO BE DELETED IS THE FOLLOWING; ELSE TYPE CANCEL!

SUBJECT: SAMPLE

TEXT: THIS IS A TEXT SAMPLE

CONFIRM (KILL OR CANCEL?) » KILL

WAITING FOR YOUR COMMAND (I, DN, FN, R, $) » P32

ERROR! NON-EXISTING RECORD NUMBER!!

WAITING FOR YOUR COMMAND (I, DN, FN, R, $) »$

WAITING FOR YOUR COMMAND (TEXT, EXAMPLE, EXERCISE, DONE) » EXAMPLE

WAITING FOR YOUR COMMAND (I, DN, FN, R, $) » I

ENTER SUBJECT(S) » EXAMPLE$

ENTER EXAMPLE » THIS IS AN EXAMPLE SAMPLE$

WAITING FOR YOUR COMMAND (I, DN, FN, R, $) » F30

SUBJECT: INTEGER

EXAMPLE: INTEGER(X) SUCCEEDS FOR X = 3 AND X = '3' BUT FAILS FOR X = 'IN'T' AND X = '3.0'.

WAITING FOR YOUR COMMAND (I, DN, FN, R, $) » P31
SUBJECT: EXAMPLE

EXAMPLE: THIS IS AN EXAMPLE SAMPLE

WAITING FOR YOUR COMMAND(I, DN, RN, R, $) >> D31

PLEASE CONFIRM BY TYPING KILL IF THE RECORD TO BE DELETED IS THE FOLLOWING;
ELSE TYPE CANCEL!

SUBJECT: EXAMPLE

EXAMPLE: THIS IS AN EXAMPLE SAMPLE

CONFIRM (KILL OR CANCEL?) >> KILL

WAITING FOR YOUR COMMAND(I, DN, RN, R, $) >> P31

ERROR: NON-EXISTING RECORD NUMBER!!

WAITING FOR YOUR COMMAND(I, DN, RN, R, $) >> $

WAITING FOR YOUR COMMAND(TEXT_EXAMPLE, EXERCISE, DONE) >> EXERCISE

WAITING FOR YOUR COMMAND(I, DN, RN, R, $) >> I

ENTER SUBJECT(S) >> SAMPLE

ENTER ANSWER TO THE EXERCISE >> FALSE

ENTER EXERCISE >> THIS IS AN EXERCISE SAMPLE.

WAITING FOR YOUR COMMAND(I, DN, RN, R, $) >> P30

SUBJECT: INTEGER

EXERCISE: TRUE

Y = 5? PLEASE ANSWER TRUE OR FALSE?

WAITING FOR YOUR COMMAND(I, DN, RN, R, $) >> P31

SUBJECT: SAMPLE

EXERCISE: FALSE

THIS IS AN EXERCISE SAMPLE.

WAITING FOR YOUR COMMAND(I, DN, RN, R, $) >> D31
PLEASE CONFIRM BY TYPING KILL IF THE RECORD TO BE DELETED IS THE FOLLOWING;
ELSE TYPE CANCEL!
SUBJECT: SAMPLE
EXERCISE: FALSE

THIS IS AN EXERCISE SAMPLE.

CONFIRM(KILL OR CANCEL?) > KILL
WAITING FOR YOUR COMMAND(I, DN, PN, R, $) > P31
ERROR! NON-EXISTING RECORD NUMBER!!
WAITING FOR YOUR COMMAND(I, DN, PN, R, $) >$
WAITING FOR YOUR COMMAND(TEXT, EXAMPLE, EXERCISE, DONE) > DONE
WAITING FOR YOUR COMMAND > EXIT

End of Execution
CPU time 5.99 S Elapsed time 13:53.00
EXIT
APPENDIX B

LISTINGS OF THE TEACH MODULE AND THE EDITOR
PROGRAM AND THE HELP FILES
55 YEAReryl, FOR

DESCRIPTION OF L, H A I T F K G K A" L

I

32 C2 THE LINE

WRITE(219,20)

20 FORMAT(' PLEASE ENTER THE FILE NAME>>",")

READ(211,41) FILENA

21 FORMAT(10)

READ(251,22) INUM, IAVAIL

22 FORMAT(12)

IF(INUM NE.0) GOTO 25

CALL LINE

WRITE(24)

24 FORMAT(' THERE IS NOT EXIST ANY FILE! FIRST CREATE THE FILE THEN "

RUN THIS PROGRAM!' )

STOP

25 DO 25 I=1,INUM+1

READ(251,23) FILE,UNITNO

26 FORMAT(10,12)

IF(FILE .EQ. FILENA) GOTO 30

CALL LINE

WRITE(313)

31 FORMAT( ' NOEXISTING FILE!!'")

GOTO 32

30 CALL LINE

WRITE(1501)

1501 FORMAT( ' IF YOU NEED HELP TYPE HELP !!'")

1505 CALL LINE

WRITE(1502)

1502 FORMAT( ' WAITING FOR YOUR COMMAND>>>",")

READ(211,41) COMMAND

IF(COMMAND .EQ. 'HELP' OR COMMAND .EQ. 'LEARN' OR COMMAND .EQ.

* EXAMPLE' OR COMMAND .EQ. 'EXERCISE' OR COMMAND .EQ. 'EXIT' OR

* COMMAND .EQ. 'QUIT') GOTO 1503

CALL LINE

WRITE(1504)

1504 FORMAT( 'PLEASE TYPE IN THE CORRECT COMMAND !!!!")

GOTO 1505

1503 IF(COMMAND .EQ. 'LEARN') CALL LEEXEX(UNITNO,"TEXT",RAND1,ICONT1)

IF(COMMAND .EQ. 'EXAMPLE') CALL LEEXEX(UNITNO+2,"EXAMPLE",RAND2,

*ICONT2)
IF(COMMAND .EQ. 'EXERCISE') CALL LEXER!UNITNO,"EXERCISE",*RANDB,ICONT)
IF(COMMAND .EQ. 'EXIT' OR COMMAND .EQ. 'QUIT') STOP
IF(COMMAND .EQ. 'HELP') CALL HELP
GOTO 1555
END

C THIS SUBROUTINE OBTAINS THE SUBJECT

SUBROUTINE ACQUIR(SUBJ)
INTEGER SUBJ
CALL LINE
WRITE(S)
1 FORMAT('ENTER THE SUBJECT(5)>"",$)
READ(S) SUBJ,INM=1,200
2 FORMAT(200A1)
DO 3 I=1,200
3 IF(SUBJ(I) .EQ. "$") RETURN
CALL LINE
WRITE(S)
5 FORMAT('MAKE SURE SUBJECT OR SUBJECTS SEPARATED WITH A ;) IS 19 OR LESS CHARACTERS LONG')
WRITE(S)
6 FORMAT('AND ENDS WITH A $ !!!')
GOTO 7
END

C THIS SUBROUTINE DISPLAYS A RECORD GIVEN THE SUBJECT WHICH C C THE RECORD IS REGARDING OF.

SUBROUTINE DISPLAY(UI,UNIT,ARIA,F,C,RAND,ICONT)
INTEGER UNIT,ARIA(20),SECTION(270),N,SELECT(100),ENSEL(50)
INTEGER S,TEXT(200),TERM(270),TEMP(270),RAND(50),DATA(3)
INTEGER UNITNO
DOUBLE PRECISION CODE
DATA(1)="(XI,"
DATA(3)="AI")
IF(SIGN .EQ. "N") GOTO 999
OPEN(UNIT=UNIT,DEVICE="DSK",ACCESS="RANDOM",RECORD SIZE=410)
IF(IP .NE. 1) GOTO 100
READ(UNIT='ARA(I),50 SECTION(I),I=1,270)
50  FORMAT(100A1,170I3)
   N=0
53  DO 51 I=1,SECTEN(101)
      IF(SECTEN(100*I+2) .EQ. 0) GOTO 51
      N=N+1
      SELECT(N)=SECTEN(100*I+2)
      N=N+1
      SELECT(N)=SECTEN(101*I+2)
51  CONTINUE
      IF(SECTEN(270) .EQ. 0) GOTO 52
      READ(UUNIT,'SECTEN(270),50) (SECTEN(IJI),IJI=1,270)
      GOTO 53
52  IF(N .NE. 0) GOTO 544
      CALL LINE
      WRITE(5,55)
      FORMAT('NON-EXISTING SUBJECT')
      GOTO 999
544  IF(ICONT .EQ. 0) GOTO 54
      IDIG=0
      DO 1111 IR=1,N,2
      DO 1112 IT=1,ICONT
         IF(RAND(IT) .NE. SELECT(IR)) GOTO 1112
      IDIG=IDIG+1
      GOTO 1111
1112  CONTINUE
1111  CONTINUE
      IF(IDIG .NE. N/2) GOTO 54
      CALL LINE
      WRITE(5,931) CODE
      FORMAT('SORRY ALL ',AS,' REGARDING THIS SUBJECT(S) HAVE
             ALREADY BEEN SELECTED')
      GOTO 999
54  S=0
      DO 56 I=1,N,2
         S=S+1
56  TENSEL(S)=SELECT(I)
57  ISS=S
      CALL RANDOM(ISS)
      IF(ICONT .EQ. 0) GOTO 134
      DO 133 I=1,ICONT
      133  IF(RAND(I) .EQ. TENSEL(ISS)) GOTO 57
      ICONT=ICONT+1
      RAND(ICONT)=TENSEL(ISS)
      UUNIT=UUNIT+1
      OPEN(UUNIT,UNIT=UUNIT,DEVICE='DSK',ACCESS='RANDOM',RECORD SIZE=603)
      READ(UUNIT,'TENSEL(ISS),388) (TEXT(IJI),IJI=1,603)
588  FORMAT(600A1,13)
      IF(TEXT(601) .EQ. 0) GOTO 59
      READ(UUNIT,'TEXT(601),60) (TEXT(IJI),IJI=601,1200)
59  IF(ICODE .NE. 'EXERCISE') GOTO 591
CALL EXRCS(TEXT)
GOTO 71
591 M=0
DO 61 IJ=201,1200
IF(TEXT(IJ).EQ. 'Y') GOTO 62
61 IN=IN+1
62 DATA(2)=M
CALL CONVER(DATA(2))
CALL LINE
WRITE(S,DATA) (TEXT(IJ),IJ=201,200+IN)
71 CALL LINE
WRITE(S,67) CODE
DO 67 FORMAT(' DO YOU WISH TO SEE MORE ',AB,' REGARDING THIS SUBJECT?')
79 READS,68 IANS
68 FORMAT(A1)
IF(IANS .EQ. 'Y' .OR. IANS .EQ. 'N') GOTO 69
CALL LINE
WRITE(S,70)
70 FORMAT(' PLEASE ENTER EITHER Y OR N !!!')
GOTO 71
69 IF(IANS .EQ. 'Y') GOTO 544
RETURN
100 N=0
READ(UNIT='ARRA(1),50) (SECTEN(I),I=1,270)
READ(UNIT='ARRA(2),50) (TENSEC(I),I=1,270)
DO 105 IN=1,270
105 TEMP(I)=TENSEC(IN)
108 DO 102 IC=1,SECTEN(101)
104 DO 103 ID=1,TEMP(101)
IF(SECTEN(100+IC+2).NE.TEMP(100+ID+2)) GOTO 103
IF(SECTEN(100+IC+2).EQ. 0) GOTO 103
N=N+1
SELECT(N)=SECTEN(100+IC+2)
N=N+1
SELECT(N)=SECTEN(101+IC+2)
GOTO 107
CONTINUE
103 IF(TEMP(270).EQ. 0) GOTO 107
READ(UNIT='TEMP(270),50) (TEMP(IJ),IJ=1,270)
GOTO 104
107 DO 106 IN=1,270
106 TEMP(IN)=TENSEC(IN)
102 CONTINUE
IF(SECTEN(270).EQ. 0) GOTO 110
READ(UNIT='SECTEN(270),50) (SECTEN(IJ),IJ=1,270)
GOTO 108
110 IF(N .NE. 0) GOTO 111
CALL LINE
WRITE(S,112) CODE
112 FORMAT(' THERE IS NO ',AB,' AVAILABLE REGARDING THIS SUBJECT')
* * *

GOTO 71
111 IF(IP .EQ. 2) GOTO 132
DO 114 IA=1,IP
S=0
READ(UNIT=ARRA1(IA),50) (SECTOR(1),I=1,270)
DO 117 IN=1,270
TEMP1(IN)=SECTOR(IN)
DO 115 IC=1,N,2
119 DO 116 ID=1,TEMP1(101)
IF(TEMP1(100+(ID-2),.NE. SELECT(IC))) GOTO 116
S=S+1
TENSEL(S)=SELECT(IC)
S=S+1
TENSEL(S)=SELECT(IC+1)
GOTO 118
116 CONTINUE
IF(TEMP1(270),.EQ.0) GOTO 110
READ(UNIT=TEMP1(270),50) (TEMP1(1),IJ=1,270)
GOTO 119
118 DO 120 IN=1,270
TEMP1(IN)=SECTOR(IN)
115 CONTINUE
IF(IS,.NE.0) GOTO 121
CALL LINE
WRITE1(121)
GOTO 71
121 DO 122 I=1,S
122 SELECT(I)=TENSEL(I)
114 N=S
132 GOTO 544
999 RETURN
END

N=0
DO 1 INDEX=INDEX+1,200
IF(SUBJCT(INDEX) .EQ. ']' .OR. SUBJCT(INDEX) .EQ. '*') GOTO 2
N=N+1
1 TEMP(N)=SUBJCT(INDEX)
2 IF(SUBJCT(INDEX) .EQ. ']' ) GOTO 3
FLAG=0
3 LOW=2
HIGH=27
4 MID=(LOW+HIGH)/2
READUNITNO 'IMD',5) (TEMARR(I),I=1,203)
5 FORMAT(A1,20,I3)
IF(TEMARR(1) .EQ. TEMP(1)) GOTO 6
IF(TEMP(1) .GT. TEMARR(1)) LOW=MID+1
IF(TEMP(1) .LT. TEMARR(1)) HIGH=MID-1
GOTO 7
6 ILOC=IMID
IF(TEMARR(2) .NE. 0) GOTO 8
CALL LINE
WRITE(5,801)
FORMAT( ' NON-EXISTING SUBJECT(S) ' )
SIGN='N'
RETURN
8 DO 12 I=1,TEMARR(2)
READUNITNO 'TEMARR(2+I) ,10 ) (SECTE(II,II=1,270)
10 FORMAT(A100,1,170)
DO 13 IA=1,100
11 IF(TEMP(IA) .NE. SECTE(IA)) GOTO 12
IP=IP+1
ARAA(IP)=TEMARR(2+I)
IF(FLAG .EQ. 0) RETURN
GOTO 13
12 CONTINUE
IF(TEMARR(2) .NE. 200) GOTO 22
IF(TEMARR(203) .NE. 0) GOTO 23
CALL LINE
WRITE(5,801)
SIGN='N'
RETURN
23 ILOC=TEMARR(203)
READUNITNO 'LOC,5) (TEMARR(I),I=1,203)
GOTO 8
END

This subroutine initializes the TEMARR array TEMP which will contain a subject and the pointers associated with it.
INTEGER TEMP(270)
COMMON /AREA2/ TEMP
DO 1 I=1,100
   TEMP(I)=I
1 CONTINUE
DO 2 I=101,270
   TEMP(I)=0
2 CONTINUE
RETURN
END

SUBROUTINE LINE
WRITE(5,1)
1 FORMAT(1X)
RETURN
END

SUBROUTINE RANDOM(I)
REAL Z
CALL RAREAL(Z)
I=INT(FLOAT(I)*Z)+1
RETURN
END

SUBROUTINE RAREAL(Z)
REAL Z
INTEGER X,Y
CALL TIME(X,Y)
DECODE(5,2,Y) Z
2 FORMAT(F5.1)
Z=(Z/10.0)-FLOAT(INT(Z/10.0))
IF(Z .EQ. 0.0) Z=24.01
RETURN
END
SUBROUTINE COHVER(IU)
ENC0DE(l,1, IU) IU
1 FORMAT(13)
CODE(3,2,10) IU
2 FORMAT(13)
RETURN
END

SUBROUTINE LEEXEXCNUflBER(CODE,RAND,ICONT)
INTEGER NUMBER,SUBJCT(201),ARRA(20),SIGN,RAND(50)
DOUBLE PRECISION CODE
CALL ACQURI(SUBJCT)
IUNIT=NUMBER+1
CALL SEARCH(SUBJCT,IUNIT,SIGN,ARRA,IP)
CALL DISPLAY(IUNIT,SIGN,ARRA,IP,CODE,RAND,ICONT)
RETURN
END

SUBROUTINE EXERCS(TEXT)
INTEGER TEXT(1200),DATA(3),RESPD0(100)
DATA(1)="(1X,"
DATA(3)="A1")
IN=0
DO 1 IJ=301,1200
IF (TEXT(IJ) .EQ. ' ') GOTO 2
1 IN=IN+1
2 DATA(2)=IN

```
CALL CONVER (DATA(2))
CALL LINE
WRITE(S, DATA) (TEXT(IJ), IJ=301,300+IN)
INM=0
DO 12 INM=201,300
IF(TEXT(M) .EQ. "2") GOTO 8
12 INM=INM+1
CALL LINE
WRITE(S,3)
FORMAT(' Enter your answer >>', *)
READ(S,4) (RESPON(IJ), IJ=1,100)
FORMAT(100A1)
IN=0
DO 5 J=1,100
IF(RESPON(IJ) .EQ. "2") GOTO 6
5 IN=IN+1
CALL LINE
WRITE(S,7)
FORMAT(' Make sure answer is 99 or less characters long and "")
WRITE(S,8)
FORMAT(' Ends with a $ sign '"')
GOTO 8
6 DO 10 IJ=1,1M
IF(RESPON(IJ) .EQ. TEXT(200+IJ)) GOTO 10
DATA(2)=INM
CALL CONVER (DATA(2))
CALL LINE
WRITE(S,11)
FORMAT(' Wrong ! Right Answer is :')
WRITE(S, DATA) (TEXT(IK), IK=201,200+INM)
RETURN
10 CONTINUE
CALL LINE
WRITE(S,13)
FORMAT(' Correct !")
RETURN
END

This subroutine displays the help file.

SUBROUTINE HELP
INTEGER TEXT(70)
ITOTAL=0
30 ICONT=0
10 IF(TOTAL .EQ. 10) GOTO 1000
READ(22,1) (TEXT(I), I=1,70)
1 FORMAT(70A1)
ITOTAL=ITOTAL+1
.TY EDIT2, FOR

DOUBLE PRECISION COMMAND
WRITE(5,1)
1 FORMAT(' IF YOU NEED HELP USING THIS TEXT-EDITOR TYPE HELP!!')
CALL LINE
WRITE(5,2)
2 FORMAT(' WAITING FOR YOUR COMMAND>>','$')
READ(5,2) COMMAND
3 FORMAT(A10)
   IF(COMMAND .EQ. 'HELP') CALL HELP
   IF(COMMAND .EQ. 'EDIT') CALL TEST
   IF(COMMAND .EQ. 'QUIT' .OR. COMMAND .EQ. 'EXIT') STOP
   GOTO 5
END

THIS SUBROUTINE GETS THE FILE NAME TO BE EDITED.

SUBROUTINE TEST
INTEGER ANSWER
DOUBLE PRECISION FILENAME
20 CALL LINE
WRITE(5,1)
1 FORMAT(' OLD OR NEW?', '')
READ(5,2) ANSWER
2 FORMAT(A3)
   IF(ANSWER .EQ. 'OLD' .OR. ANSWER .EQ. 'NEW') GOTO 10
   CALL LINE
   WRITE(5,3)
3 FORMAT(' PLEASE TYPE EITHER OLD OR NEW!')
   GOTO 20
10 CALL LINE
   WRITE(5,4) ANSWER
4 FORMAT(' ENTER THE ',A3,' FILE NAME>>','$')
   READ(5,5) FILENAME
5 FORMAT(A10)
   CALL EDITOR(FILENAME, ANSWER, 'TEST')
   RETURN
END
SUBROUTINE EDITOR(FILEA,ANSWER,CODE)
INTEGER ANSWER,UNITNO,A,RECNUN,CODE
DOUBLE PRECISION FILEA,FILE,COMMAD
OPEN(UNIT=25,DEVICE='985',FILE='FILEIST.RAN',ACCESS='RANDON',
* RECORD SIZE=15)
READ(25*,1,1) INUM,IAVAIL
1 FORMAT(212)
IF(ANSWER .EQ. 'OLD') GOTO 20
IAHS='NEW'
ICODE1=0
IAHS2='NEW'
ICODE2=0
IAHS3='NEW'
ICODE3=0
IF(INUM .EQ. 0) GOTO 16
15 DO 10 I=2,INUM+1
READ(25*1,2) FILE
10 IF(FILE .EQ. FILEA) GOTO 12
GOTO 16
12 CALL LINE
WRITE(5,13)
13 FORMAT(' THIS FILE ALREADY EXISTS!!')
CALL LINE
WRITE(5,14) ANSWER
14 FORMAT(' IF YOU WISH TO CONTINUE,REENTER THE ','A3,' FILE NAME OR
* ELSE TYPE CANCEL><',5)
READ(5,25) FILEA
IF(FILEA .EQ. 'CANCEL') RETURN
GOTO 15
16 INUM=INUM+1
WRITE(25*INUM+1,2) FILEA,IAVAIL,ICODE1,ICODE2,ICODE3
2 FORMAT(16,12,3I1)
UNITNO=IAVAIL
IAVAIL=IAVAIL+4
WRITE(25*1,1) INUM,IAVAIL
GOTO 22
20 IF(INUM .EQ. 0) GOTO 200
DO 21 I=2,INUM+1
READ(25*1,2) FILE,UNITNO,ICODE1,ICODE2,ICODE3
21 IF(FILE .EQ. FILEA) GOTO 22
200 CALL LINE
WRITE(5,14) ANSWER
READ(5,25) FILEA
23 FORMAT(' NON-EXISTING OLD FILE NAME!!')
CALL LINE
WRITE(5,14) ANSWER
WRITE(5,10)
25
IF(FILEA .EQ. 'CANCEL') RETURN
GOTO 20
22 CALL LINE
IF(ICODE1 .EQ. 0) IANS1='NEW'
IF(ICODE1 .EQ. 1) IANS1='OLD'
IF(ICODE2 .EQ. 0) IANS2='NEW'
IF(ICODE2 .EQ. 1) IANS2='OLD'
IF(ICODE3 .EQ. 0) IANS3='NEW'
IF(ICODE3 .EQ. 1) IANS3='OLD'
WRITE(5,100) ANS,FILENA
30 FORMAT( 'EDITING ' ,AS,' FILE NAME' ,A10)
167 CALL LINE
WRITE(5,160) ANS
168 FORMAT( 'WAITING FOR YOUR COMMAND(TEXT,EXAMPLE,EXERCISE,DONE)>'
*,A)
READ(5,165) COMMA
165 FORMAT(A10)
IF(COMMA .EQ. 'TEXT' .OR. COMMA .EQ. 'EXAMPLE' .OR. COMMA .EQ.
  'EXERCISE' .OR. COMMA .EQ. 'DONE') GOTO 77
CALL LINE
WRITE(5,160)
166 FORMAT( 'PLEASE TYPE EITHER TEXT OR EXAMPLE OR EXERCISE OR DONE
   '!!'
)*
GOTO 167
77 IF(COMMA .EQ. 'TEXT') GOTO 600
IF(COMMA .EQ. 'EXAMPLE') GOTO 700
IF(COMMA .EQ. 'EXERCISE') GOTO 800
IF(IANS1 .EQ. 'NEW') ICODE1=0
IF(IANS1 .EQ. 'OLD') ICODE1=1
IF(IANS2 .EQ. 'NEW') ICODE2=0
IF(IANS2 .EQ. 'OLD') ICODE2=1
IF(IANS3 .EQ. 'NEW') ICODE3=0
IF(IANS3 .EQ. 'OLD') ICODE3=1
READ(25',1) IRRM,IRRML
DO 255 JZ=2,IRRML+1
READ(25',256) FILE,UNITNO
255 FORMAT(A10,12)
256 FORMAT(A10,12)
IF(FILE .EQ. 'FILEA') GOTO 257
257 WRITE(25',257) FILEA,UNITNO,ICODE1,ICODE2,ICODE3
RETURN
600 UNITM=UNITNO
604 CALL LINE
WRITE(5,601)
601 FORMAT( 'WAITING FOR YOUR COMMAND(I,OK,PM,R,$)>>',A)
READ(5,602) A,RENUM
602 FORMAT(A,110)
IF(A .EQ. 'I' .OR. A .EQ. 'B' .OR. A .EQ. 'P' .OR. A .EQ. 'R' .OR.
  A .EQ. '$') GOTO 603
CALL ILLEGAL
GOTO 604
603 IF(A .EQ. 'I') CALL ADD(UNITM,IANS1,'TEXT')
IFIA. 'E' CALL DELETE(IUNIT, IAMS1, RECHUM, 'TEXT')
IFIA. 'P' CALL PRINT(IUNIT, IAMS1, RECHUM, 'TEXT')
IFIA. 'R' CALL REPORT(IUNIT, IAMS1, 'TEXT')
IFIA. 'Y' GOTO 167
GOTO 604

700 IUNIT=UNITNO+2
704 CALL LINE
WRITE(S,601)
READ(S,402) A,RECHUM
IFIA. 'I'.OR. A .EQ. 'D'.OR. A .EQ. 'P'.OR. A .EQ. 'R'.D
*R. A .EQ. 'S') GOTO 703
CALL ILLEGAL
GOTO 704

703 IFIA. 'E' CALL ADD(IUNIT, IAMS2, 'EXAMPLE')
IFIA. 'P' CALL DELETE(IUNIT, IAMS2, RECHUM, 'EXAMPLE')
IFIA. 'R' CALL REPORT(IUNIT, IAMS2, 'EXAMPLE')
IFIA. 'Y' GOTO 167
GOTO 704

800 IUNIT=UNITNO+4
804 CALL LINE
WRITE(S,601)
READ(S,402) A,RECHUM
IFIA. 'I'.OR. A .EQ. 'D'.OR. A .EQ. 'P'.OR. A .EQ. 'R'.D
*R. A .EQ. 'S') GOTO 803
CALL ILLEGAL
GOTO 804

803 IFIA. 'E' CALL ADD(IUNIT, IAMS3, 'EXERCISE')
IFIA. 'P' CALL DELETE(IUNIT, IAMS3, RECHUM, 'EXERCISE')
IFIA. 'R' CALL PRINT(IUNIT, IAMS3, 'EXERCISE')
IFIA. 'Y' GOTO 167
GOTO 804
END

 THIS SUBROUTINE DISPLAYS THE LEGAL EDITING COMMANDS
 WHEN AN ILLEGAL EDITING COMMAND IS TYPED IN.

 SUBROUTINE ILLEGAL
 CALL LINE
 WRITE(S,34)
34 FORMATTED_ILLEGAL_COMMAND!_LEGAL_COMMANDS_ARE_AS_FOLLOWS:
 CALL LINE
 WRITE(S,33)
35 FORMATTED I_TO_INSERT_A_NEW_RECORD:
 CALL LINE
 WRITE(S,37)
37 FORMAT(' D FOLLOWED BY A RECORD NUMBER')
38 WRITE(5,NB)
39 FORMAT(' TO DELETE A RECORD')
40 CALL LINE
41 WRITE(5,NB)
42 FORMAT(' P FOLLOWED BY A RECORD NUMBER')
43 WRITE(5,NB)
44 FORMAT(' TO PRINT A RECORD')
45 CALL LINE
46 WRITE(5,NB)
47 FORMAT(' R TO GENERATE A READABLE COPY OF THE QUESTIONS AND SUB
48JECTS AVAILABLE SO FAR')
49 CALL LINE
50 FORMAT(' $DOLLAR SIGN TO STOP EDITING THE CURRENT FILE')
51 RETURN
52 END

SUBJCT(201)=IAVAIL
WRITE(UNITNO'1,SPOT,110) (SUBJCT(IN),IN=1,201)
READ(UNITNO'1,IVAL1,3) ILINK
IVAL1=4
WRITE(UNITNO'1,IVAL1,2) ISOFAR,AVAREC
IF(ILINK .EQ. 0) GOTO 113
IVAL1=ILINK
114 WRITE(UNITNO'1,2) IAVAIL,INOREC
GOTO 120
113 IAVAIL=IVAL1+1
WRITE(UNITNO'1,IVAL1,3) ILINK
GOTO 114
120 GOTO 10
4 IAVAIL=IVAL1+1
WRITE(UNITNO'1,IVAL1,3) ILINK
3 FORMAT(13)
GOTO 8
1 ILINK=0
AVAREC=3
IVAL1=4
WRITE(UNITNO'1,IVAL1,3) ILINK
INOREC=1
WRITE(UNITNO'1,2) IAVAIL,INOREC
WRITE(UNITNO'2,2) INOREC,AVAREC
10 CONTINUE
126 CALL LINE
WRITE(5,11)
11 FORMAT(?) ENTER SUBJECT(S)>>',"
READ(5,12) (SUBJCT(IN),IN=1,200)
12 FORMAT(200A1)
DO 122 IS=1,200
122 IF(SUBJCT(IS) .EQ. '9') GOTO 123
CALL LINE
WRITE(5,124)
124 FORMAT(' MAKE SURE SUBJECT(SUBJECTS SEPARATED WITH ;) IS 199 OR
*LESS CHARACTERS LONG AND,")
WRITE(5,125)
125 FORMAT(' ENDS WITH A $ $!!!! )
GOTO 126
123 CONTINUE
14 FORMAT(13)
CALL INQUIR(UNITNO'1,ANSWER,AVAREC)
IF(CODE .NE. 'EXERCISE') GOTO 20
207 CALL LINE
WRITE(5,201)
201 FORMAT(?) ENTER ANSWER TO EXERCISE >>',"
READ(5,202) (TEXT(IJ,J),IJ,J=1,100)
202 FORMAT(100A1)
DO 203 IKK=1,100
203 IF(TEXT(IKK) .EQ. '2') GOTO 204
CALL LINE
WRITE(5,205)
205 FORMAT(' MAKE SURE ANSWER IS 99 OR LESS CHARACTERS LONG AND')
WRITE(5,206)
206 FORMAT(' END WITH A $ SIGN !!')
GOTO 207
207 CALL LINE
WRITE(5,15) CODE
210 FORMAT(' MAKE SURE EXERCISE IS 99 OR LESS CHARACTERS LONG')
WRITE(5,211)
211 FORMAT(' AND END WITH A $ SIGN !!')
GOTO 204
20 CALL LINE
WRITE(5,15) CODE
15 FORMAT(' ENTER '',AB,'','$')
READ(1,16) (TEXT(IJ),IJ=1,1000)
16 FORMAT(1000A1)
DO 17 I=1,1000
17 IF(TEXT(I) .EQ. '') GOTO 18
CALL LINE
WRITE(5,19) CODE
19 FORMAT(' MAKE SURE '',''AB,'', IS 99 OR LESS CHARACTERS LONG AND
*ENDS WITH A $ !!!!')
GOTO 20
20 IF(I .GT. 400) GOTO 25
WRITE(UNITNO*AVAREC,26) (SUBJECT(IJ),IJ=1,200),(TEXT(IJ),IJ=1,1)
26 FORMAT(600A1)
RETURN
25 READ(UNITNO*1,2) IAVAIL,INREC
WRITE(UNITNO*AVAREC,27) (SUBJECT(IJ),IJ=1,200),(TEXT(IJ),IJ=1,400
*) ,IAVAIL
27 FORMAT(600A1,13)
READ(UNITNO*IAVAIL,3) ILINK
WRITE(UNITNO*IAVAIL,26) (TEXT(IJ),IJ=401,1)
IF(ILINK .EQ. 0) GOTO 28
IAVAIL=ILINK
GOTO 29
28 IAVAIL=IAVAIL+1
WRITE(UNITNO*IAVAIL,3) ILINK
29 WRITE(UNITNO*1,2) IAVAIL,INREC
100 RETURN
END
THIS SUBROUTINE ADJUSTS THE INVERTED INDEX BY SUBJECT

SUBROUTINE INVDIR(UNITNO, ANSWER, AVERAGE)
INTEGER UNITNO, ANSWER, SUBJECT(I), TEMP(270), J, AVERAGE, TEMPARR(203)
INTEGER SECTEN(270), FLAG, N
COMMON /AREA1/ SUBJECT, AREA2, TEMP
OPEN(UNIT=UNITNO, DEVICE='DSK', ACCESS='DIRECT',
      RECNO=INDEX, NERO=100)
  FLAG=0
IF(ANSWER .EQ. 'OLD') GOTO 100
CALL INIT(UNITNO)
ANSWER='OLD'
100 INDEX=0
10 CALL INIT
N=0
DO 1 IINDEX=INDEX+1,200
  IF(SUBJECT(INDEX) .EQ. ' ' OR SUBJECT(INDEX) .EQ. ' ') GOTO 2
  N=N+1
  TEMP(N)=SUBJECT(INDEX)
1   IF(SUBJECT(INDEX) .EQ. ' ' ) GOTO 3
   FLAG=0
3   ILow=2
   IMid=(ILow+Ihighlight)/2
   READ(UNITNO, 'IMID',5) (TEMPARR(I), I=1,203)
5 FORMAT(A1,203I1)
   IF(TEMPARR(1) .EQ. TEMP(1)) GOTO 4
   IF(TEMP(1) .LT. TEMPARR(1)) ILow=IMid+1
   IF(TEMP(1) .GT. TEMPARR(1)) Ihighlight=IMid-1
   GOTO 7
6   ILoc=IMid
   IF(TEMPARR(2) .NE. 0) GOTO 9
   TEMPARR(2)=TEMPARR(2)+1
9   READ(UNITNO, 'IAVAIL',9) IAVAIL
   FORMAT(I3)
   TEMPARR(TEMPARR(2)+2)=IAVAIL
   TEMP(100)=AVERAGE
   WRITE(UNITNO, 'IMID',10) (TEMP(I), I=1,270)
10  FORMAT(100A1,170I3)
   IAVAIL=IAVAIL+1
   WRITE(UNITNO, 'IAVAIL',9) IAVAIL
   WRITE(UNITNO, 'IMID',5) (TEMPARR(I), I=1,203)
   IF(FLAG .EQ. 0) RETURN
   GOTO 11
7   DO 12 I=1,TEMPARR(2)
   READ(UNITNO, 'TEMPARR(2+I)',10) (SECTEN(J), J=1,270)
12  DO 13 I=1,100
   IF(TEMP(1) .NE. SECTEN(I)) GOTO 12
   GOTO 11
   GOTO 11
IF(SECTEM(101) .EQ. 84) GOTO 15
ILOC=TEMARK(2+1)
21
SECTEM(101)=SECTEM(101)+1
SECTEM(100)=SECTEM(101)+2=AIVAREC
WRITE(UNITNO',ILOC,10) (SECTEM(IJ),IJ=1,270)
IF(FLAG .EQ. 0) RETURN
GOTO 11
15
IF(SECTEM(270) .NE. 0) GOTO 17
ILOC=TEMARK(2+1)
20
READ(UNITNO',1,9) IAVAL
SECTEM(270)=IAVAL
WRITE(UNITNO',ILOC,10) (SECTEM(IJ),IJ=1,270)
INUM=1
WRITE(UNITNO',IAVAL,16) INUM,AIVAREC
FORMAT(100X,213)
IAVAL=IAVAL+1
WRITE(UNITNO',1,9) IAVAL
IF(FLAG .EQ. 0) RETURN
GOTO 11
17
ILOC=SECTEM(270)
READ(UNITNO'*ILOC,10) (SECTEM(IJ),IJ=1,270)
IF(SECTEM(270) .NE. 0) GOTO 17
IF(SECTEM(101) .EQ. 84) GOTO 20
GOTO 21
12
CONTINUE
IF(TEMARK(2) .NE. 200) GOTO 22
IF(TEMARK(203) .NE. 0) GOTO 23
READ(UNITNO',1,9) IAVAL
TEMARK(203)=IAVAL
WRITE(UNITNO',ILOC,5) (TEMARK(IJ),IJ=1,203)
INUM=1
IAV=IAVAL+1
WRITE(UNITNO',IAVAL,24) INUM,IAV
FORMAT(1X,213)
IAVAL=IAVAL+1
WRITE(UNITNO',IAVAL,24) INUM,IAV
24
TEMP(101)=1
TEMP(102)=AIVAREC
WRITE(UNITNO',IAVAL,10) (TEMP(I),I=1,270)
IAVAL=IAVAL+1
WRITE(UNITNO',1,9) IAVAL
IF(FLAG .EQ. 0) RETURN
GOTO 11
23
ILOC=TEMARK(203)
READ(UNITNO',ILOC,5) (TEMARK(IJ),IJ=1,203)
GOTO 8
22
TEMARK(2)=TEMARK(2)+1
READ(UNITNO',1,9) IAVAL
TEMARK(TEMARK(2)+2)=IAVAL
WRITE(UNITNO',ILOC,5) (TEMARK(IJ),IJ=1,203)
GOTO 25
END
SUBROUTINE INITIL(UnitNo)
INTEGER UnitNo

OPEN(Unit=UnitNo, DEVICE='DSK', ACCESS='RANDOM',
  RECORD SIZE=610)
ICH='A'
WRITE(UnitNo,'2,1') ICH
FORMAT(A1)
ICH='B'
WRITE(UnitNo,'3,1') ICH
ICH='C'
WRITE(UnitNo,'4,1') ICH
ICH='D'
WRITE(UnitNo,'5,1') ICH
ICH='E'
WRITE(UnitNo,'6,1') ICH
ICH='F'
WRITE(UnitNo,'7,1') ICH
ICH='G'
WRITE(UnitNo,'8,1') ICH
ICH='H'
WRITE(UnitNo,'9,1') ICH
ICH='I'
WRITE(UnitNo,'10,1') ICH
ICH='J'
WRITE(UnitNo,'11,1') ICH
ICH='K'
WRITE(UnitNo,'12,1') ICH
ICH='L'
WRITE(UnitNo,'13,1') ICH
ICH='M'
WRITE(UnitNo,'14,1') ICH
ICH='N'
WRITE(UnitNo,'15,1') ICH
ICH='O'
WRITE(UnitNo,'16,1') ICH
ICH='P'
WRITE(UnitNo,'17,1') ICH
ICH='Q'
WRITE(UnitNo,'18,1') ICH
ICH='R'
WRITE(UnitNo,'19,1') ICH
ICH='S'
WRITE(UnitNo,'20,1') ICH
ICH='T'

READ (*) UnitNo, ICH
CLOSE(UnitNo)


```
WRITE(*,1) ICH
ICH="U"
WRITE(*,1) ICH
ICH="V"
WRITE(*,1) ICH
ICH="W"
WRITE(*,1) ICH
ICH="X"
WRITE(*,1) ICH
ICH="Y"
WRITE(*,1) ICH
ICH="Z"
WRITE(*,1) ICH
ICH=28
WRITE(*,1) ICH
ICH=2
FORMAT(3)
RETURN
END

```

```
SUBROUTINE LINE
WRITE(5,1)
1 FORMAT(1X)
RETURN
END

```

```
SUBROUTINE INIT
INTEGER TEMP(270)
COMMON /ANEO/ TEMP
DO 1 I=1,100
1 TEMP(I)=.F.
DO 2 I=101,270
2 TEMP(I)=0
RETURN
END

```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
C THIS SUBROUTINE DELETES A RECORD.

SUBROUTINE DELETE(UNITNO, ANSWER, RECNUM, CODE)
INTEGER UNITNO, ANSWER, RECNUM, SUBJCT(21), TEXT(1200), DATA(3)
INTEGER TENVUN(21), TENVEX(20)
DOUBLE PRECISION CODE
OPEN(UNIT=UNITNO, DEVICE='DISC', ACCESS='DIRECT')
* RECORD SIZE = 403
IF (ANSWER .EQ. 'OLD') GOTO 777
CALL LINE
WRITE(5,776)
776 FORMAT(' DELETE CAN NOT BE DONE! THERE IS NO RECORD IN THE FILE!')
RETURN
777 DATA(1)='(IX, '
DATA(2)=0
DATA(3)='A1')
ISPOPT=2
4 READ(UNITNO, 'ISPOT,1) (SUBJCT(IN), IN=1, 20)
1 FORMAT(200I3)
IF(RECNUM .LE. SUBJCT(1)) GOTO 2
IF(SUBJCT(201) .EQ. 0) GOTO 3
ISPOT=SUBJCT(201)
RECNUM=RECNUM-SUBJCT(1)
GOTO 4
3 CALL LINE
WRITE(5,555)
555 FORMAT(' ERROR! NON-EXISTING RECORD NUMBER!!')
RETURN
2 CALL LINE
WRITE(5,5)
5 FORMAT(' PLEASE CONFIRM BY TYPING YES IF THE RECORD TO BE DELETED IS THE FOLLOWING: ')
WRITE(5,115)
115 FORMAT(' ELSE TYPE CANCEL! ')
READ(UNITNO, 'SUBJCT(RECNUM+1),6) (TEXT(IN), IN=1, 601)
6 FORMAT(400A1,13)
DO 116 I=1, 200
116 TEXTEX(I)=TEXT(I)
ITEM=SUBJCT(RECNUM+1)
ITEMS=TEXT(601)
CALL LINE
WRITE(5,7)
7 FORMAT(' SUBJECTS: ',9)
DO 8 I=1, 200
IF(TEXT(I) .EQ. ' ') GOTO 9
8 DATA(2)=DATA(2)+1
9 IF=DATA(2)
CALL CONVER(DATA(2))
WRITE(5,DATA) (TEXT(I), I=1, IN)

DATA(2)=0
CALL LINE
WRITE(S,11) CODE
FORMAT(1X,AB,'$'
DO 12 I=201,600
IF(TEXT(I) .EQ. 'S') GOTO 13
11
DATA(2)=DATA(2)+1
READ(UNIT0,TEXT(601),14) (TEXT(IN).IN=601,1200)
12
FORMAT(600A1)
DO 13 I=601,1200
IF(TEXT(I) .EQ. 'S') GOTO 13
13
I=M=DATA(2)
CALL CONVER(DATA(2))
WRITE(S,DATA) (TEXT(IN).IN=201,IN=200)
CALL LINE
WRITE(S,16)
FORMAT("CONFIRM(KILL OR CANCEL)" )",$'
READ(S,17) IANS
16
FORMAT(AS)
IF(IANS .EQ. "CANCEL") RETURN
SUBECT(I)=SUBECT(I)+1
SUBECT(RECNUM+1)='0'
TSENSUB(I)=1
TSENSUB(IN)=SUBECT(201)
IN=1
DO 18 I=1,200
IF(SUBECT(I) .EQ. '0') GOTO 18
IN=IN+1
TSENSUB(IN)=SUBECT(I)
18 CONTINUE
DO 19 I=N+1,200
TSENSUB(I)=0
WRITE(UNIT9) 'ISPUI,1') (TSENSUB(IN),IN=1,201)
READ(UNIT0,120) IAVAIL,INGREC
19
FORMAT(213)
IF(IITEM2 .NE. 0) GOTO 21
WRITE(UNIT0) 'ITEM1,22') IAVAIL
GOTO 23
20
WRITE(UNIT9) 'ITEM2,22') IAVAIL
FORMAT(22)
WRITE(UNIT9) 'ITEM1,22') ITEM2
21 IAVAIL=ITEM1
INGREC=INGREC-1
WRITE(UNIT0,120) IAVAIL,INGREC
CALL BELINV(UNIT0,1,TEXT,ITEM1)
RETURN
END
THIS SUBROUTINE ADJUSTS THE INVERTED DIRECTORY WHEN A RECORD IS DELETED.

SUBROUTINE DELINI(UUNITO,UNITI,ITEM)
INTEGER TEMTEX(200),UNITINO,ITEM,SECTEM(270),TEMP(270)
INTEGER TEMMARU(203),FLAG,N
COMMON /AREA2/ TEMP
OPEN(UNITI,UUNITO,DEVICE="DSK",ACCESS="RANDOM").
RECORD SIZE=6100
FLAG=1
IINDEX=0

DO 1 N=0
1 INDEX=INDEX+1,200
IF(ITEMEX(INDEX) .EQ. ' ' .OR. TEMTEX(INDEX) .EQ. ' ') GOTO 2
N=N+1
TEMP(N)=ITEMEX(INDEX)
2 IF(ITEMEX(INDEX) .EQ. ' ') GOTO 3
3 FLAG=0
4 IF(ITEME(ITEM) .EQ. '') GOTO 5
5 IF(TEMMARU(ITEM) .EQ. TEMP(ITEM)) GOTO 4
IF(TEMP(ITEM) .GT. TEMMARU(ITEM)) ILOU=IHMID+1
IF(TEMP(ITEM) .LT. TEMMARU(ITEM)) INHIGHL=INHID+1
GOTO 7
6 DO 12 I=1,ITEMAR(2)
ISPOF=TEMMAR(2*1)
READ(UUNITI,ISPOF,10) (SECTEM(IJ),IJ=1,279)
12 IN=1,100
13 IF(TEMP(ITEMA) .NE. SECTEM(ITEMA)) GOTO 12
52 DO 50 IB=1,SECTEM(101)
50 IF(SECTEM(IB*IB=1,SECTEM(101)
51 ISPOF=SECTEM(270)
READ(UUNITI,ISPOF,10) (SECTEM(IJ),IJ=1,279)
GOTO 52
12 CONTINUE
READ(UUNITI,TEMARRU(203),10) (TEMMARU(IJ),IJ=1,203)
GOTO 4
51 SECTEM(100*IB*IB=1)
WRITE(UUNITI,ISPOF,10) (SECTEM(IJ),IJ=1,279)
10 FORMAT(100A1,17013)
IF(FLAG .EQ. 0) RETURN
GOTO 11
END
CEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
C THIS SUBROUTINE CONVERTS THE INTEGER REPRESENTATION OF A
C NUMBER TO ITS CHARACTER REPRESENTATION.

CEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
SUBROUTINE CONVERT(IU)
ENCODE(3,1,IU) IU
1 FORMAT(13)
ENCODE(3,2,IU) IU
2 FORMAT(13)
RETURN
END

CEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
C THIS SUBROUTINE PRINTS A RECORD.

CEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
SUBROUTINE PRINT(UNITNO,ANSWER,RECHNUM,CODE)
INTEGER UNITNO,ANSWER,RECHNUM,SUBJCT(1),TEXT(1,201),DATA(3)
DOUBLE PRECISION CODE
OPEN(UNITNO,DEVICE='DG',ACCESS='RANDOM'
*RECORD SIZE=603)
IF(ANSWER.EQ.'OL0') GOTO 777
CALL LINE
WRITE(1,776)
776 FORMAT('PRINT CAN NOT BE DONE! THERE IS NO RECORD IN THE FILE!
*')
RETURN
777 DATA(1)="(IX,`
DATA(2)=0
DATA(3)="A1"
ISPOT=2
4 READ(UNITNO,ISPOT,1) (SUBJCT(IN)I,IN=1,201)
1 FORMAT(10I3)
IF(RECHNUM.LE.SUBJCT(1)) GOTO 2
IF(SUBJCT(201).LE.0) GOTO 3
ISPOT-SUBJCT(201)
RECHNUM=RECHNUM-SUBJCT(1)
GOTO 4
3 CALL LINE
WRITE(1,555)
555 FORMAT('ERROR!NON-EXISTING RECORD NUMBER!!')
RETURN
2 CALL LINE
READ(UNITNO,SUBJCT(RECHNUM+1),6) (TEXT(IN),IN=1,601)
6 FORMAT(10D41,13)
CALL LINE
WRITE(13,7)
```plaintext
7  FORMAT(' SUBJECT: ',A)
   DO 8 I=1,200
   IF(TEXT(1).EQ. ' ') GOTO 9
8  DATA(2)=DATA(2)+1
9  IM=DATA(2)
   CALL CONVER(DATA(2))
   WRITE(S,DATA) (TEXT(I),I=1,IM)
   DATA(2)=0
   CALL LINE
   WRITE(S,11) CODE
11  FORMAT(I*X,AB,'*','$')
   DO 12 I=201,400
   IF(TEXT(1).EQ. ' ') GOTO 13
12  DATA(2)=DATA(2)+1
   READ(UNITO,TEXT(401),14) (TEXT(IN),IN=601,1200)
14  FORMAT(600A1)
   DO 15 I=601,1200
   IF(TEXT(1).EQ. ' ') GOTO 13
15  DATA(2)=DATA(2)+1
13  IM=DATA(2)
   CALL CONVER(DATA(2))
   WRITE(S,DATA) (TEXT(IN),IN=201,IN+200)
   RETURN
END

THIS SUBROUTINE PRODUCES A READABLE COPY OF THE SUBJECTS
AND THE RECORDS REGARDING THEM.

SUBROUTINE REPORT(UNITO,ANSWER,CODE)
   INTEGER UNITO,ANSWER,DATA(3),SUBJECT(201),FLAG,TEXT(1200),N
   INTEGER FALG
   DOUBLE PRECISION CODE
   OPEN(UNIT=UNITO,DEVICE='DSK',ACCESS='RANDOM',
   *RECORD SIZE=601)
   WRITE(23,101) CODE
101  FORMAT(I*X,AB)
   WRITE(23,102)
102  FORMAT('**********')
      FLAG=0
      N=0
      IF(ANSWER.EQ. '0LB') GOTO 10
   CALL LINE
   WRITE(S,20)
20  FORMAT(' FILE IS EMPTY!!!')
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
RETURN
10 ISPOT=2

READUNITNO(ISPOT,1) (SUBJECT(IN),IM=1,201)
1 FORMAT(20113)
IF(SUBJECT(1) .EQ. 0 .AND. SUBJECT(201) .NE. 0) GOTO 3
IF(SUBJECT(1) .EQ. 0 .AND. SUBJECT(201) .NE. 0) GOTO 5
IF(SUBJECT(1) .NE. 0 .AND. SUBJECT(201) .EQ. 0) GOTO 6
IF(SUBJECT(1) .NE. 0 .AND. SUBJECT(201) .NE. 0) GOTO 7

3 IF(FLAG .NE. 0) GOTO 100
CALL LINE
WRITE(15,20)
RETURN

5 ISPOT=SUBJECT(201)
GOTO 4

6 FLAG=1
7 FLAG=1
DO 66 I=1,SUBJECT(1)
READUNITNO(SUBJECT(I+1),67) (TEXT(IN),IM=1,601)
67 FORMAT(600A1,13)
IF(TEXT(601) .EQ. 0) GOTO 68
READUNITNO(TEXT(601),69) (TEXT(IN),IM=601,1200)
69 FORMAT(600A1)
68 IM=0
DO 70 IA=1,200
IF(TEXT(IA) .EQ. ' ' ) GOTO 71
70 IM=IM+1
71 IA=IA+1
WRITE(23,72) IM
72 FORMAT(' RECORD NUMBER: ',13)
WRITE(23,73)
73 FORMAT(' SUBJECT(S): ',9)
DATA(2)=IM
CALL CONVEXTA(DATA(2))
WRITE(23,DATA) (TEXT(IA),IM=1,IM)
IM=0
DO 74 IA=201,1200
IF(TEXT(IA) .EQ. ' ' ) GOTO 75
74 IM=IM+1
75 WRITE(23,76) CODE
76 FORMAT(IX,9,' ',9)
DATA(2)=IM
CALL CONVEXTA(DATA(2))
WRITE(23,DATA) (TEXT(IA),IM=201,IM+200)
76 WRITE(23,1066)
1066 FORMAT(IX)
IF(FLAG .EQ. 1) GOTO 100
ISPOZ=SUBJECT(201)
GOTO 4
100 CALL LINE
WRITE(15,80)
80 FORMAT(' REQUIRED COPY IS IN FILE FOR23.DAT!!!')
SUBROUTINE HELP
INTEGER TEXT(70)
ITOTAL=0
ICOUNT=0
10 IF(ITOTAL .EQ. 0) GOTO 1000
READ(21,1) (TEXT(I),I=1,70)
1 FORMAT(70A1)
ITOTAL=ITOTAL+1
WRITE(5,1) (TEXT(I),I=1,70)
ICOUNT=ICOUNT+1
IF(ICOUNT .NE. 5) GOTO 10
IF(ITOTAL .EQ. 30) GOTO 1000
CALL LINE
WRITE(5,2)
2 FORMAT(' 10 CONTINUE TYPE C ; OR ELSE TYPE E ' )
READ(5,3) ICNAR
3 FORMAT(A1)
IF(ICNAR .EQ. 'C') GOTO 30
IF(ICNAR .EQ. 'E') GOTO 1000
CALL LINE
WRITE(5,4)
4 FORMAT('PLEASE TYPE EITHER C OR E ! ' )
GOTO 20100Q
RETURN 21
END
TOP LEVEL COMMANDS ARE EDIT AND QUIT(EXIT).

IF THE PROGRAM IS INSTRUCTED WITH AN EDIT COMMAND THE QUESTION OLD OR
NEW IS ASKED. IF THE FILE TO BE EDITED HAS ALREADY BEEN CREATED, THIS
FILE IS CONSIDERED AN OLD FILE AND IN SUCH CASE OLD SHOULD BE TYPED
IN. IF THE INTENTION IS TO CREATE A NEW FILE AND EDIT IT, THEN NEW SHOULD
BE TYPED IN.

IN EITHER OF THE ABOVE CASES YOU ARE ASKED TO PROVIDE THE NAME OF
THE FILE TO BE EDITED. ONCE THE SYSTEM IS IN EDIT MODE, COMMANDS
AVAILABLE ARE TEXT, EXAMPLE, EXERCISE, DONE.

TEXT COMMAND ENABLES ONE TO EDIT THE TEXT PORTION OF THE FILE.
EXAMPLE COMMAND ENABLES ONE TO EDIT THE EXAMPLE PORTION.
EXERCISE COMMAND MAKES THE EDITING OF THE EXERCISE PORTION OF THE
FILE POSSIBLE AND DONE COMMAND CAUSES THE SYSTEM TO TRANSFER CONTROL
to the top level.

IN ANY OF THE TEXT, EXAMPLE OR EXERCISE MODES COMMANDS AVAILABLE ARE:
(1) TO INSERT A NEW RECORD; (2) FOLLOWED BY A RECORD NO. TO DELETE A
RECORD; (P) FOLLOWED BY A RECORD NO. TO PRINT A RECORD; (R) TO GENERATE
A READABLE COPY OF THE SUBJECTS AND MATERIALS(TEXT, EXAMPLE, EXERCISE);
(1) TO STOP EDITING CURRENT PORTION (TEXT, EXAMPLE, EXERCISE) OF THE FILE.
IF IT IS TYPED IN DEPENDING ON WHICH MODE (TEXT, EXAMPLE, EXERCISE) THE
SYSTEM IS IN, YOU ARE ASKED TO PROVIDE THE SUBJECT AND THE MATERIAL
(TEXT, EXAMPLE, EXERCISE). IF D FOLLOWED BY A RECORD NUMBER IS TYPED IN,
THE RECORD TO BE DELETED IS DISPLAYED AND THE USER IS ASKED TO EITHER
CONFIRM OR CANCEL THE REQUEST.
P FOLLOWED BY A RECORD NUMBER CAUSES THE REQUESTED RECORD TO BE
DISPLAYED AND R GENERATES A READABLE COPY OF THE SUBJECTS AND
MATERIALS(TEXT, EXAMPLE, EXERCISE). S TRANSFERS CONTROL TO THE SECOND
LEVEL WHERE DONE TRANSFERS CONTROL TO THE TOP LEVEL WHERE QUIT OR
EXIT TERMINATES THE EXECUTION OF THE PROGRAM.
TOP LEVEL COMMANDS ARE LEARN, EXAMPLE, EXERCISE, QUIT(EXIT).

IF LEARNING ABOUT A SUBJECT IS THE PURPOSE, COMMAND LEARN SHOULD BE TYPED IN.

ONCE THE SYSTEM IS IN LEARN MODE YOU ARE ASKED TO ENTER THE SUBJECT WHICH YOU WANT TO SEE SOME TEXTS REGARDING OF.

IF YOU LIKE TO SEE SOME EXAMPLES REGARDING A SUBJECT, COMMAND EXAMPLE SHOULD BE TYPED IN. COMMAND EXERCISE ENABLES ONE TO SEE SOME EXERCISES REGARDING SUBJECTS. EXERCISES ARE SELECTED ACCORDING TO THE SUBJECTS GIVEN AND YOU ARE TO ANSWER THEM. THESE ANSWERS ARE CHECKED FOR CORRECTNESS. COMMAND QUIT(EXIT) TERMINATES THE EXECUTION OF THE PROGRAM.
BIBLIOGRAPHY


