USING MODEL 204 DATABASE MANAGEMENT SYSTEM TO BUILD A COMPUTER ASSISTED TEST CONSTRUCTION SYSTEM FOR A FORTRAN COURSE

BY

HUNE-CHI FUH

Bachelor of Science in Engineering Science

National Chung Kung University

Tainan, Taiwan

Republic of China

1976

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the Degree of
MASTER OF SCIENCE
May, 1983

USING MODEL 204 DATABASE MANAGEMENT SYSTEM TO BUILD A COMPUTER ASSISTED TEST CONSTRUCTION SYSTEM FOR A FORTRAN COURSE

Report Approved:

_____
Report Adviser

_____

_____

_____
Dean of Graduate College

TABLE OF CONTENT

Chapter Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The per unit cost of computing has dropped rapidly in the past two decades. This cost reduction is attributed primarily to the advances in computing hardware, particularly the dramatic improvement in computational speed. Even if comparisons are made only over the past decade, it can still be said that the cost of computing has decreased significantly.

It seems, however, that the growth and improvements in the field of educational computer usage have not kept pace with the opportunities provided by these favorable hardware and cost changes. The pedagogical strategies involved and the types of educational computing being used have not changed significantly. Furthermore, the percentage of college and university students who are exposed to educational computing has not grown as rapidly as predicted.

The availability of a computer-assisted test construction (CATC) system through computer networks has just recently begun to emerge as a novel and powerful tool for the individual classroom teacher. More than one hundred computer-accessible question banks have been built at schools and colleges in the United States. Although their

use seems to be increasing, only a tiny minority of faculty are regular users; the great majority make either marginal use of them or no use at all. This situation may exist due to old habits which are hard to change, a lack of awareness about the existence of the question banks, or the fact that many faculty members are simply intimidated by the computer and believe that anything connected with it requires an operating knowledge of sophisticated machines. In addition to the above reasons, economic factors play a major role. Many institutions still feel CATC represents add-on cost in an academic institution and they simply cannot afford it, even when the actual cost of human istructors may be greater.

According to Lippey(1), there are two fundamentally different approaches to the CATC system. The first system (CATC-R) is based on the ability of the computer to facilitate retrieval, in other words, the computer merely manipulates a categorized database of complete questions with or without answers, but does not actively create or add to the content of the individual questions. The second system (CATC-G) involves the computer in a true generation role, developing a set of algorithms which, when carried out by the computer, will result in a complete question. The approaches to building a CATC system vary greatly across a broad spectrum of utilization philosophies, subject matter areas, and the computer facility.

The CATC system's general aim is to reduce the time an instructor must spend on the mechanical processes of writing, reproducing, and grading examinations. A properly designed and operated CATC system would allow more time for two of the instructor's most important functions: selecting examination content and interpreting student performance. The computer can assume the duties of word processor, grader, and statistician. The scoring and statistical procedures are basically administrative functions of the computer and as such are not the principal focus of CATC. Future research may be extended to these functions and more.

The purpose of this paper is to give an indepth description of the design philosophy and the features of this design which will build a CATC-R FORTRAN question bank. The Model 204 database management system is used to build this question bank. In Model 204, there need be no fixed format for any record in the file and the field in the record can be added, deleted and changed very easily. For example, a new field can be added to records even though that field was not in the original record. A completely new type of record can be added to a file at any time without any change in the file structure. The system provides the user with a very easy-to-use query language to express both simple and complex data retrieval and update operations. In the future, this CATC system's application could be applied elsewhere, for example, other area's question bank for any other academic discipline can be further developed based on

this project. The query language of Model 204 enables the instructor with no programming background to access the question bank, this is the major reason in selecting Model 204 to build this question bank. Model 204 also provides interactive processes to manipulate the question in the bank at a terminal. A variety of security features can be implemented to provide a question bank with protection against unauthorized use of the account, files, records, fields, and procedures.

CHAPTER II

LITERATURE REVIEW

CATC systems have been under development for the last decade. The literature in this field mainly describes methodological approaches and logistical concerns. Three primary sources in professional literature offer reviews of the use of CATC in a diversity of testing environments. The first of these sources is the March 1973 issue of the Journal of Educational Technology, which is devoted to the topic of test construction. The second source is the book Computer-Assisted Test Construction, edited by Gerald Lippey (1) and published by Educational Technology Publications in 1974. The most recent source is a paper presented at the Proceedings of 14th Annual Convention of the Association for Educational Data Base Systems in 1976 by Jesse M. Heines (2). The following review will describe the most important and well known CATC systems.

A "CALL CARDS" SYSTEM

Salisnjak (3) assembles tests from item cards stored in a filing cabinet. Questions are stored in a computer according to question number (from 1 to 999) with two additional digits: one to identify the instructor and the

other to identify his department. The instructor is given a printed list of the questions and a punched "call card" for each question. He arranges these call cards according to chapters in a textbook. Other cross-references can be achieved by placing other copies of call cards under additional headings. To prepare a test, the instructor selects the desired questions, noting the question numbers. Appropriate call cards are then selected and arranged in the desired order; the sequence of the call cards determines the sequential numbering of the questions on the test printout.

TEST GENERATOR

This system was developed by Bailey (4) at Oklahoma State University primarily to reorder the test questions for a Fortran course. The test questions are selected and punched on cards, then they are put together with control cards to indicate what questions will be used. Tests can be generated according to three options: (1) the same set of questions may be printed in random order, (2) alternating tests may contain all even-numbered or all odd-numbered questions, or (3) some randomly selected subset of questions may be printed. The third option allows considerable flexibility. For example, suppose that there are 25 questions, but each test is to have 10 questions. The questions may simply be selected randomly from the 25 questions. It is also possible to select random subsets of subsets. Thus, a subset of 10 questions might also be

obtained by randomly selecting 3 from the first 10, 2 from the next 5 and 5 from the last 10.

For each test generated a record is written onto a 'test key' file in which the numbers, in the original set of questions, of questions chosen are placed in the order in which they appear on the test. For each test printed, two lines of comments or instructions are printed at the top of the test. More lengthy messages or a lengthy program segment can also be printed at the top of each test. Each student's responses are punched, along with his name, test number, and four-digit code. These cards are used as input to a test grading program which grades them and lists the test results.

THE COMPUTER GENERATED REPEATABLE TEST SYSTEM (CGRT)

This system was developed by Prosser (5) at Indiana University primarily to provide frequent testing for personalized student instruction in large classes. The classification and selection of questions is very simple. This system provides a means for generating multiple forms of the same test. Jensen has used it to generate 4000 different forms for a class of 1500 students. He allows students to take a test on a specific topic as often as they like and counts only the highest grade. His philosophy in this approach is that "...one should ask only what one wishes the student to know, but ask it in so many different ways the student cannot learn the items without learning the concept."

THE MENTREX SYSTEM

MENTREX introduced by Libaw (6) is a commercially developed, computer-based system that not only provides testing but also a sophisticated scoring service with various learning aids, and which might appropriately be used in support of computer-managed instruction. It is sufficiently flexible and adaptable to be useful in a wide variety of instructional system settings. This flexibility and adaptablity provides a wide range of possible ways to build a CATC system. There are two methods of test construction: one is to select questions through different major classifications and subclassifications based on Bloom's Taxonomy. The other way to construct tests is to fill out a special optical scanning form, stating the parameters of the test desired. Number and categories of questions desired also can be specified. Hence, the selection of questions usually requires looking through carefully indexed catalogs of test questions. As Libaw states, "The system permits the use of multiple choice items augmented by two types of Data Sets containing charts, graphs, diagrams, drawings, descriptions of experiments, etc. One type of data set consists of narrative data that can be printed out by the computer. The other makes use of a simple "hybrid" technology. The computer will key an item to a drawing which is reproduced, off-line, on a thermal spirit duplicating or direct offset master. All items based on data sets will automatically pull the appropriate data

out of the file for the student to refer to when answering the question."

THE CLASSROOM TEACHER SUPPORT SYSTEM (CTSS)

The Classroom Teacher Support System introduced by Toggenburger (7) and Lippey (8) was developed as an experimental computer application under a joint-study agreement between the Los Angeles City Unified School District and the IBM corporation. CTSS is now installed in several institutions and a number of question banks have been developed for use with it. This system constructs multiple choice examinations according to teacher specified criteria such as course, category, difficulty level, behavioral level, and key words. The system can also work with "macro" questions, i.e., a set of questions which always appear together and may be preceeded by tables, charts, graphs, and so forth, printed in the test. The system attempts to obtain all of the questions requested according to the criteria specified. If there are not enough eligible questions available, the program first ignores any behavior level and then any difficulty level in order to obtain the number of questions indicated in the request. Further control over test content may be exercised by the provision that a teacher's own questions can be added to the test and that selected questions supplied may be suppressed.

TEST CONSTRUCTION AND ANALYSIS PROGRAM (TCAP)

The Test Construction and Analysis Program introduced by Baker (9) employs a terminal which allows the instructor to interactively examine, add, and delete questions in order to construct a test with desired statistical characteristics. The questions are classified by the following parameters: keyword describing the question, question difficulty level, question discriminating power, number of prior uses of the question and date of the most recent use. The test requested is defined by a set of general parameters that all questions in the test must satisfy. These parameters are: keywords, the number of questions, upper and lower bound values for difficulty level and discriminating power, upper limit to the number of prior uses of the questions, and the cut-off date for the most recent use of the question. This process can be repeated for up to 10 content areas. Upon fulfillment of the final area definition, the computer displays a table containing the number of questions requested per area, the number found per area, predicted value of the test mean and variance and reliability coefficient. At this point, the instructor can interactively add and delete questions within each area of the test in order to balance the content emphasis and adjust the predicated test statistics to a more desired value.

EDUCATIONAL TESTING SERVICE (ETS)

Educational Testing Service described by Epstein (10) has a long history of development of question classification and selection systems. ETS currently has a CATC system to help select questions from its huge question banks. The system does not print tests, but simply returns question numbers that fit specified characteristics. The ETS technique does not differ conceptually in any basic way from other CATC systems, but it is remarkable for the depth and detail of its classification structure. Each question record includes a question ID number, a question classification, history of its use, up to five sets of statistics, code for security level and current activity, and twelve 15-character keywords. When tests are requested from the computer, the statistical specification provided must include acceptable ranges of means and standard deviations of difficulty and discrimination indices. Question selection is also guided by the number of questions needed from each category. The computer surveys the entire pool, then produces a reduced question pool of all those questions that could be eligible for the test, and then assigns a priority number to each question according to a complex procedure. This selection procedure insures that the pool remains as balanced as possible in order to maximize the number of parallel tests that can be assembled.

STUDENT ORIENTED CLASSROOM ANALYSIS AND TEST EVALUATION SYSTEM (SOCRATES)

SOCRATES introduced by Geisler (11), Seely & Willis (12), Gray-Shllberg & Willis & Seely (13), Willis (14), Johnson & Willis & Moore & Seely (15) has been available to the faculty of the nineteen campuses of the California State Universities and Colleges since the fall of 1975. The availability of the question banks through computer networks in this system provides a powerful tool for the individual classroom teacher. Tests can be ordered by telephone and delivered by courier or generated by batch mode from a terminal at any campus. The generation of questions is completely under the teacher's control. No knowledge of programming on a computer is required to take maximum advantage of the system. SOCRATES retrieves questions by subject category, difficulty level, behavior level, and key word cross references. Questions which require the use of additional material (enhanced questions) and questions which occur together (macro questions) are allowed. Questions may be deleted and replaced with ease. A test may contain up to one hundred fifty questions. Parallel tests and multiple versions of the same test may be generated easily. The resulting tests may be directed to the user's campus to be produced on the local high speed printer or they may be printed on Diablo or Qume daisy-wheel printers that permit lower-case characters, subscripts, and superscripts. Finally, most test questions can be scored automatically,

giving the teacher raw scores with ranks and two different levels of question analysis.

PROBGEN

In a CATC-R system, the computer does not solve the problems, answers are generated only if the problems have been solved previously and the answers have been included in the question bank. In contrast to this approach, a CATC-G system employs the computational power of the computer to assign values to the variables within problems and to solve the result. The PROBGEN system introduced by Collins and Duff (16,17,18) can process any numerical problem which can be systematically generalized into an answer question. It is applicable only to the generation of quantitative problems: one program or subroutine must be written for each type of problem. Using interactive mode, PROBGEN prompts the user to provide the generalized solution to each problem, any major restriction which might apply, and the limits on all variables for the text of each problem. Following this sequential question-and-answer dialog, PROBGEN then writes or assembles the actual problem that will create the desired number of non-identical problems of the defined type. Each test can consist of one or more problems of as many types as are input to PROBGEN.

CHAPTER III

DESIGN PHILOSOPHY AND SYSTEM FEATURES

Introduction

The Department of Computing and Information Sciences of The Oklahoma State University offers a 3-hour introductory programming FORTRAN course. At present, the course has 4 large lecture sections per semester, with a total of over 700 students. As the instructor's workload increases, it becomes very difficult to generate tests for different sections. Even when a large quantity of well-constructed test questions have been sufficiently prepared in advance, the task of typing and producing the examination copies is tedious. In addition, where technical terms are not familiar to the typist, the chances of misspelled or misrepresented questions usually increase and the time required for typing is lengthened. If a FORTRAN question bank were constructed, the task would be made easier for everyone involved.

The project described in this paper involved the design and implementation of a CATC system which satisfies the needs described in the previous paragraph. The approach is similiar to TCAP (Test Construction and Analysis Program) as described in the Chapter II. The system which resulted from

this project enables instructors to build, edit, query and maintain a test question bank. The system is based entirely on the Model 204 database management system.

An instructor selects a question from the question bank as follows. First a scanning procedure is invoked which creates a small question subpool. This subpool contains test questions which satisfy the attributes prescribed by the instructor. The test questions then are selected from this subpool and assign to the test. Hence, this question bank requires a very large number of test questions. Up-date activities such as adding to and modifying the questions are also included as a part of the system which was developed by this project. This project does not include security protection. However, in the future the system can very easily be made to include security features. Another anticipated variation would be to have the system allow user to submit tests to a series of programs that will reformat the tests and store them on a disk file. This file could be retrieved at a terminal by the students as a test, or could be used for mass production of the test.

The following topics which relate to building and using this question bank will be discussed in this chapter:

(1) Entering questions: This section describes the steps which compose records to be stored in the question bank.

(2) Organization of question bank: This section describes the organization and storage structure of the question bank within the Model 204 system.

(3) Initialization of question bank: This section describes the steps which use the Fast Load Utility to load questions into the bank.

(4) Question bank maintenance: This section describes the operations required to maintain the question bank.

(5) Automatic test construction: This section describes the automatic generation of the test questions.

Entering Questions

Introduction

A variety of ways exist to record information in a form that can be read by a computer system. The following topics describe the steps which compose the records to be stored in the system.

(1) Field description: the field name meanings and field characteristics.

(2) Record format: the formats of records which are stored in this system.

(3) Data compression: the mechanism to save storage space and to get the original form of a question.

(4) Substitute variables: the mechanism to substitute variable values into a question.

(5) Input data format: format used to input a large number of questions into the question bank.

Field Description

The most elementary data unit in the system is called a field. Each field has a name and value. Hence, to build a question bank, we must first decide the fields to be assigned to each question. One of these fields provides a unique identification for each question, others may be used by the test generation program to direct the selection of questions, or used by the instructor to score the student's answers, etc. The meaning of the record field names are summarized in Figure 1. In the future, new fields may be added very easily by the DEFINED command.

Field characteristics are shown in Table I; the first column is the field name, the second and the third columns indicate the kind of value stored in each field, the fourth column indicates the length of each field, and the fifth column indicates the ranges of corresponding values.

The TYPE field indicates the type of question; its values range from 1 to 5 representing, in order, true-false, multiple choice, completion, syntax error, and essay questions.

The TOPIC field represents the chapter and section in the textbook to which the question belongs, whose value ranges from 1 to 9999. (To save storage space, some TOPIC field values need only 3 bytes to store them, so the first character of section value may be blank.)

The CHAPTER field represents the chapter in the textbook to which the question belongs, whose value ranges from 1 to 99.

The LEVEL field indicates the difficulty level of the question, with the field value 1 representing the easiest questions, and the field value 5 represents the hardest questions. Field values from 2 to 4 represent middle difficulty levels.

The NO field stores the question ID number, whose value ranges from 1 to 9999.

The TIME field represents the estimated minutes required to finish the question; its value ranges from 1 to 99.

The FLAG field indicates whether or not substitute variables are to be used. When the value of the FLAG field is equal to 1, the question has substitute variables, if it not equal to 1, it does not have substitute variables.

The CONTENT field stores the actual question text.

The SQ field stores the sequence number of question text.

The SUB field stores the substitute array variables, if any.

The ANS field stores the answer key to the question.

```
TYPE    : type of question.
TOPIC   : question subject.
CHAPTER: chapter associated with question.
LEVEL   : difficulty level.
NO      : question ID number.
TIME    : estimated time to solve question.
FLAG    : whether or not there are substitute
          variables.
CONTENT: question text.
SQ      : sequence number of question text.
SUB     : substitute variables.
ANS     : answer key to the question.
```

Figure 1. Explanation of field names.

TABLE I

FIELD CHARACTERISTICS

FIELD NAME	ALPHANUMERIC	NUMERIC	LENGTH	VALUE RANGE
TYPE		X	1	1 - 5
TOPIC		X(+' ')	4	1 1 - 9999
CHAPTER		X	2	1 - 99
LEVEL		X	1	1 - 5
NO		X	4	1 - 9999
TIME		X	2	1 - 99
FLAG		X	1	0,1
CONTENT	X		255	
SQ		X	1	1 - 9
SUB	X		99	
ANS	X		99	

Record Format

After deciding the fields to be included in the system, the record format should be chosen next. The record format A shown in Figure 2 is used primarily to store the question's features and the first part of the question's text. In Model 204, the string field attribute when stored in the file has one byte of memory storage to indicate its length. The maximum field string length is therefore 255 bytes. However a question's length may exceed 255 bytes, in which case it needs another record to store the excess part. This extended record only needs 3 fields to uniquely identify and store it as shown in record format B shown in Figure 2. The NO field stores the question ID number, the SQ field starting from 2 contains the order of CONTENT field in this question.

Format A: For the first question record.

TYPE	TOPIC	LEVEL	NO	TIME	FLAG	CONTENT	SQ	SUB	ANS

Format B: For the extended question record.

	NO		CONTENT	SQ	

Figure 2. Record format in the file.

Data Compression

In order to save storage space and to get the special form of the question, we use a data compression mechanism. For example, consider a question such as the one shown in

Figure 3. Most of the space in the question is blank and some lines in the question need to started at a new line. We use the special character '@' to represent starting a new line; if there are more than 4 blank spaces then we use the pound sign to delimit an integer count of the number of blank spaces. The actual pound character should be represented by two pound signs (i.e.##). The question shown in Figure 3, using the mechanism which we just described, is stored in memory as shown in Figure 4. We also need a reverse mechanism to transform compressed text back to the original question form. This reverse mechanism is not described in this paper and is left for a future research topic.

```
What values are printed when the following program
is executed?
COLUMN 12345678901234567890123456789012345678901
              READ,A,B,C
              A = B
              B = C
              C = A
              PRINT,A,B,C
                .
                .
       $ENTRY
         7.0  12.0  8.0
       $IBSYS
       //
```

Figure 3: Example question before data compression.

```
What values are printed when the following program
 is executed?@COLUMN 12345678901234567890123456789
01234567890@#13#READ,A,B,C@#13#A = B@#13#B = C@#13
#C = A@#13#PRINT,A,B,C@#14#.@#14#.@#7#ENTRY@#9#7.0
  12.0  8.0@$IBSYS@#7#//
```

Figure 4: Example question after data compression.

Substitute Variables

The instructor may want to change some variables' value in the question. Questions with this feature are indicated by a FLAG field equal to 0. The changeable variables are indicated by the special prefix character '|' and the different variables are assigned with numeric value ranging from 1 to 9 according to their appearance order in the question. The SUB field contains the sets of these values; each set of array values is separated by the special character '↑'. An example question and its SUB field value are shown in Figure 5. In the first usage of this question 10 would be substituted for |1 , and 23 for |2; in the second usage 15 for |1, and 32 for |2; etc.

Suppose memory locations A and B contain the value |1 and |2 respectively. Show the contents of A and B after execution of the following code:@#10#H = A@#10#A = B@#10#B = H@#15#A=@#15#B=#15#C=
SUB:10,23↑15,32↑20,-10↑

Figure 5. Substitute array example and its value.

Input Data Format

Each question is produced according to the classification attribute and typed into computer readable form to load into the question bank as shown in Figure 6. For each question, the first record stores the features of the question and the second record stores the question text. Depending on the question length, it may continue to use the third record format to store the question text. The fifth column of each record indicates which kinds of data format it contains. For example, if a record contains a '*' at the fifth column then the record stores the question features, if it contains a '#' then the record stores the first question text, if it contains a blank the record stores the extended question text.

First Card: Question Features.

field	NO	*	TYPE	TOPIC	LEVEL	TIME	FLAG	SUB	ANS
column	1-4	5	6-7	8-11	12-13	14-15	17	18-53	54-250

Second Card: First Question Text.

field	SQ	#	C O N T E N T
column	1-4	5	6-255

Third Card: Extended Question Text.

field	SQ		C O N T E N T
column	1-4	5	6-255

Figure 6. Input data format.

Organization of Question Bank

Introduction

Model 204 provides a very flexible environment for handling large or small amounts of data. Relationships among individual data records are maintained at a logical level through the use of key indices. There need not be any physical linkage among data records. Model 204 uses inverted file retrieval techniques which facilitate fast retrieval of data without requiring expensive scanning of the database itself.

All retrievals and many update operations are performed based on 'field name=value' specification. Attributes are assigned to each field thus defining retrieval options and storage formats. These attributes, which provide a variety of options that suit the user requirements and optimize response, are listed in the following section. A record is a collection of fields. Each record is variable in length and need contain only the fields which pertain to it. A field may occur zero, one, or many times within a single record, and there is no limit to the number of fields in a record. There need be no fixed format for any record. Each record is automatically assigned a unique internal record number, which is used by the system to build index entries for the record. A file is an arbitrary collection of records. Records in a file are normally related to each other by chronology of their creation, or by a particular

sorted key field, or by a particular hashed key field.

To fully understand the meaning of field attributes, it is necessary to understand the structure of a Model 204 file first. Further detail is given in the following section. A Model 204 file is logically divided into five tables or sections. (1) File Control Table: This table keeps the control information of the file. (2) TABLE A: This table contains a dictionary of the field name and its attributes, 'CODED' field values and the values of 'FOR-EACH-VALUE' field. (3) TABLE B: This table contains the retrievable data of the actual records in the file. (4) TABLE C and TABLE D: TABLE C and TABLE D comprise the indexing structure of a Model 204 file. If the field has either the KEY or NUMERIC RANGE or FRV attribute, Model 204 generates entries in these tables. TABLE C entries are made for each distinct value of a KEY or NUMERIC RANGE or FRV field. TABLE D entries are made for each record which contains a particular value if that value occurs in more than one record in the file.

Field Attributes

The field attributes determine how a field may be used and how it is stored internally in the Model 204 file. There is almost no restriction on defining the attributes to a field. The field attributes are listed here, and their meanings are described briefly in the Appendix A. Further details can be found in chapter 2 of File Manager's Technical Reference Manual (19). The underscored attribute

is the default attribute if it is not defined.

(A) Functional Attributes

KEY / NON-KEY

VISIBLE / INVISIBLE

NUMERIC RANGE / NON-RANGE

FOR EACH VALUE(FRV) / NON-FRV

DEFERABLE / NON-DEFERABLE

LEVEL

These field attributes determine how a field can be used in query language and how they affect the retrieval speed to access the records. For example, when a field is defined with KEY attribute, Model 204 makes special entries in the index structure. During retrieval, the system goes directly to the appropriate index entry to find which records satisfy the selection criteria without searching through other records in the file. On the contrary, when a field is defined with NON-KEY attribute, retrieval is done by searching through the whole file (TABLE B) sequentially to find which records satisfy the selection criteria. All of these attributes are described in Appendix A.

(B) Representation Attributes

CODED / NON-CODED

BINARY / STRING

MANY-VALUED / FEW-VALUED

UPDATE IN PLACE / UPDATE AT END

OCCURS

LENGTH

PAD

These field attributes determine how a field is physically stored in the Model 204 file. The field values are stored in TABLE B in one of three formats depending upon the selection of STRING/BINARY or CODED/NON-CODED field attributes. The choice affects space requirements and the time required for updates. For example, when a field is defined with the CODED attribute, the character string is stored in TABLE A and a four-byte value code pointing to that character string is stored in the logical record in TABLE B. If the average length for CODED field values is more than four characters, space is saved when there are several records which all contain the same value. The coding and decoding of these value codes takes time and may slow down updates and retrievals. All of these attributes are described in Appendix A.

The nondefault field attributes for this CATC system are listed in Table II. In this system, we want to select the specified questions through the type of question (TYPE), the topic in the textbook (TOPIC or CHAPTER), the question difficulty level (LEVEL), or the specified question ID number (NO). All of these fields are assigned the KEY attribute and have index entries in TABLE C and TABLE D; questions with specified field values can be accessed

directly by using these index entries. The CHAPTER field is a subfield of TOPIC; it is assigned the INVISIBLE attribute. An INVISIBLE field only has entries in the index structure and does not occupy any storage in TABLE B.

TABLE II

FIELD ATTRIBUTES

FIELD	KEY	INVISIBLE
TYPE	X	
TOPIC	X	
CHAPTER	X	X
LEVEL	X	
NO	X	
TIME		
FLAG		
CONTENT		
SQ		
SUB		
ANS		

File Organization

After choosing the fields to be included in a file and their attributes, the file parameters and file size must be determined before building the question bank. However, before setting the file parameters and file size, the file structure of Model 204 must be fully understood to accurately calculate the space and its parameters. A Model 204 file consists of a large number of fixed-length storage units. Each of these fixed-length units is called a page. The page size depends on the installation of the computer hardware system. In the current CATC system, the page size is 6184 bytes. Each page includes a 40 byte control

information section which is not available for data storage.

In order to minimize disk storage space and to optimize record retrieval techniques, the records in TABLE B are divided into internal file 'segments'. According to the Model 204 reference manual the maximum number of records stored in one file segment is 8*page size. For a file with a page size of 6184, there are slightly fewer than 50,000 records per segment. The size estimation of TABLE C and TABLE D depends on the file size multipler N, which represents the number of internal file segments. N can be calculated as:

$$N = \frac{\text{\# of records in the file}}{8 * \text{page size}} \quad \text{(rounded to integer)}$$

There are many file parameters to calculate and set. A detailed description is given in chapter 3 of the File Manager's Technical Reference Manual (19). The following description emphasizes the file organization and a few of the parameters that are very important in building the Model 204 file. At the end of this section, there is a very simple example illustrating the major aspects of the Model 204 file structure. A Model 204 file is logically divided into five tables or sections:

File Control Table: This table keeps track of the file parameter settings, ddnames of all data sets in the file, status of the file, and other file control information. The File Control Table is a fixed

size(8 pages); usually it is small by comparison with the rest of the file.

TABLE A: This table is subdivided into three sections:

(1) A dictionary of the field names and their attributes.

(2) The FEW-VALUED section contains the field values of all fields with the FEW-VALUED field attribute and either the CODED or FOR EACH VALUE attribute.

(3) The MANY-VALUED section contains the field values of all fields with the MANY-VALUED field attribute and either the CODED or FOR EACH VALUE attribute.

A one page field name dictionary provides much better performance than a multiple page section. This can reduce the amount of I/O required to access the dictionary. TABLE A is created using hash coding. It can not be expanded unless reorganization of the file is made, therefore extreme care should be exercised when allocating the space for SIZEA.

TABLE B: This table contains the retrievable data of all of the actual data records in the file. TABLE B is usually the largest section of the file and can be expanded after the file has been loaded. Each record starts with 5 bytes of record number

overhead which are used by the system to access the record directly.

The basic structure depends on the field representation attributes as shown in Table III. Each field name code occupies two bytes of storage. The field values are stored in one of following three formats depending upon the selection of field attributes.

(1) The field is CODED, its field value is stored in TABLE A and a four-byte value code pointing to that field value is stored along with field name code.

(2) The field is BINARY, it stores the field name code along with four bytes of binary value.

(3) The field is the default STRING, it stores the field name code and character strings of field value with an additional byte to indicate its length.

The most important parameter to be set in this table is BRECPPG, which determines the maximum number of records stored on one page. It affects the assignment of internal record numbers. Numbers are sequential, with 0 belonging to the first record on the first page of TABLE B. Each page has BRECPPG numbers of record allocated to it. An accurate calculation of this parameter is important

because the setting of this parameter can affect storage utilization in TABLE B, TABLE C, TABLE D, and operating efficiency.

TABLE III

BASIC STRUCTURE IN TABLE B

CODED :

Field Name Code	Coded Field Value

NON-CODED:

Field Name Code	Binary Value	
Field Name Code	Length	Field Value

TABLE C: This table makes up the indexing structure which contains a 'field name=value' pair for each value of every field with the KEY or FRV or NUMERIC RANGE attribute. The basic structure in TABLE C is shown in Table IV. TABLE C is divided into slots of six bytes each. One slot is required for each distinct value of a field with a KEY or FRV or NUMERIC RANGE attribute. Another slot is required for each segment of the file containing records with that value. If that field value is unique, it requires two slots, one for the value and one for the segment which contains it. If that field value is not unique, it requires one slot for the value and, if the file size multipler (N) is 2, two additional slots for the segments which contain the value and pointer point to TABLE D. TABLE C is also created

in hash which can not be expanded, unless reorganization of the file is made, therefore extreme care should be exercised when allocating the space for SIZEC.

TABLE IV

BASIC STRUCTURE IN TABLE C

Unique :	Field Name=Value	Pointer	(to TABLE B)
Non-Unique :	Field Name=Value	Pointer	Pointer
		(to TABLE D)	

TABLE D: This table is divided into a number of sections as follows.

(1) Preallocated field record descriptions: If any preallocated fields are defined in a file, one page is needed to store the record description. This record description describes the arrangement of the fields in the storage preallocated in each record.

(2) Procedure dictionary: The procedure dictionary associates a procedure name or alias with information about the location of the procedure's text and a class if the procedure is secured.

(3) Procedure text: In most cases, the text of each procedure requires one page and the larger

text procedure may need more than one page.

(4) Access Control Table (ACT): The Access Control Table (ACT) contains the entries which map user classes and procedure classes into privileges. The ACT is allocated from TABLE D, one page at a time as needed. The allocation is made by the SECURE command.

(5) Index entries: This section contains a list of records which have a given value for a field. The amount of space used by the index depends upon how many records contain a particular 'field name=value' pair and how many of these records are in each file segment. For each segment in TABLE B, if those 'field name=value' pairs that occur more than 3 percent of the time in the segment, it allocates one page for each of these values pairs. If those 'field name=value' pairs occur fewer than 3 percent of the time, it needs 6 bytes for the field value and 2 bytes for each record which has these value pairs.

TABLE D is usually larger than TABLE C, depending on the number of KEY , NUMBERIC RANGE fields and user language procedures. TABLE D can be expanded very easily from the free space.

Example: This example describes the basic aspects of a Model 204 file. The major features of the question are listed in Table V. The fields in this table,

except the TIME field, have the KEY attribute. The File Control Table is not listed here since the user does not have work directly with it. TABLE A, TABLE B, TABLE C, and TABLE D for these questions are listed in Tables VI, VII, VIII, IX.

When the system open this file, the system reads in the File Control Table to get the file parameters and the control information of TABLE A, TABLE B, TABLE C, TABLE D. Suppose, for example, it is desired to find the questions with TYPE=1 (true-false) and TOPIC=1 1. The system would go to TABLE A to find whether or not TYPE and TOPIC have the KEY attribute. If both have the KEY attribute, it would go to the index table (TABLE C and TABLE D) and find that TYPE=1 appears in records 1 and 2, and that TOPIC=1 1 appears in records 1 and 3. The system would compare the two lists of record numbers and determine that record 1 was the only record which satisfied both selection criteria. Once record 1 has been selected, we can access the data record in TABLE B for additional information.

The NON-KEY field attribute may also be specified in retrieval conditions. For example, suppose it is desired to find all the questions with TYPE=1 (true-false) and TIME=4. The system goes to TABLE A to find out that the TYPE field has

the KEY attribute, but the TIME field does not. Then the system goes to the index table (TABLE C and TABLE D) and finds that TYPE=1 appears in records 1 and 2. Each record on the found list is then sequentially searched through by the system in TABLE B to find out that the record 2 satisfied both selection criteria. A sequential search through the records can have a significant adverse effect on performance if a large number of records are to be searched.

TABLE V

MAJOR QUESTION EXAMPLE FEATURE

number	TYPE	TOPIC	LEVEL	NO	TIME
1	1(TRUE/FALSE)	1 1	1	1	3
2	1(TRUE/FALSE)	1 2	1	2	4
3	3(COMPLETION)	1 1	4	3	6

TABLE VI

PHYSICAL STRUCTURE IN TABLE A

Control Info.	TYPE	TOPIC	LEVEL	NO	TIME	. . . . (field name)
Control Info.	. . . . . (few-valued)					
Control Info.	. . . . . (many-valued)					

TABLE VII

PHYSICAL STRUCTURE IN TABLE B

Record1	TYPE|1|1	TOPIC|3|1 1	LEVEL|1|1	NO|1|1	CONTENT|37|.
Record2	TYPE|1|1	TOPIC|3|1 2	LEVEL|1|1	NO|1|2	CONTENT|45|.
Record3	TYPE|1|3	TOPIC|3|1 1	LEVEL|1|4	NO|1|3	CONTENT|233|
. .					

TABLE VIII

PHYSICAL STRUCTURE IN TABLE C

NO=1	B	REC NO. 1
NO=2	B	REC NO. 2
NO=3	B	REC NO. 3
TYPE=TRUE/FALSE	D	PAGE 2
TYPE=COMPLETION	B	REC NO. 3
TOPIC=1 1	D	PAGE 3
TOPIC=1 2	B	REC NO. 2
LEVEL=1	D	PAGE 4
LEVEL=4	B	REC NO. 3
. .		

TABLE IX

PHYSICAL STRUCTURE IN TABLE D

PAGE 0	EXISTENCE MAP
PAGE 1	PRELOCATED FIELD RECORD DESCRIPTION
PAGE 2	TYPE=TRUE/FALSE,1,2
PAGE 3	TOPIC=1 1,1,3
PAGE 4	LEVEL=1,1,2
PAGE .	.
PAGE .	.

Initialization of Question Bank

The process of creating a file in Model 204 is complex, and failure to observe good practices can be both expensive and time-consuming. After the structure of Model 204 files is fully understood, the method used to calculate the file parameters and file sizes is not difficult. A detailed description is given in chapter 3 of the File Manager's Technical Reference Manual (19). Some parameters and table sizes depend heavily on the actual data records; hence, the actual calculating procedures are skipped here. For further loading calculations refer to the above manual. The file parameters are set in the File Control Table by the CREATE command. Next, the file must be initialized by the INITIALIZE command to erase all information stored in the file except for the setting of file parameters.

The following steps are necessary to verify that the input data is properly formatted and that the load program is functioning properly: (1) Check the format of the input data, if possible, using sample data with the P statement in Fast Load Utility (FLOD) to print the sample data. (2) Run FLOD to load the sample data and create the records in the Model 204 file. (3) After loading the records, issue the DISPLAY FIELD command and use the user query language to print all information. This helps to check that the field names are properly defined and that the data is actually loaded into the Model 204 file. (4) Finally, before the

actual loading process takes place, the load time and working storage needs to be estimated. There is no exact formula for calculating how long it will take to load a file. The most reliable method to estimate it is to use the sample load time to extrapolate the value. The working storage can be estimated from the number of deferred update indices produced by the sample load.

The FLOD language provides limited programming capability. If complicated edits or large amounts of data manipulation are required, a host language program can be used. There are two ways to run a Fast Load Utility. The first method involves running five separate job procedures. The second method requires only one job step, which automatically invokes the five procedures. The first FLOD procedure formats and loads the data into TABLE B. This procedure also creates a deferred update index record to pass to the second procedure(the SORT utility) which sorts the deferred index records. The third procedure accepts each deferred update record and applies the index entries to TABLE C and TABLE D. When the third procedure encounters an index entry for a For-Each-Value(FRV) field it puts these index records in a temporary file. The fourth procedure invokes the SORT utility again to sort these FRV records which are in a temporary file. The fifth and last procedures which adds the deferred FRV entries to TABLE C and TABLE D. A sample loading program is shown in Appendix B.

Question bank maintenance

To maintain this question bank, the Model 204 provides some utility procedures. When a file is created, the exact space requirements are seldom known. Once it has been created, the sizes of TABLE A and C may not be changed without recreating the file and reloading all the data. The sizes of TABLE B and TABLE D can be changed quite easily from the free space with the INCREASE and DECREASE commands. The amount of free space can be increased by the INCREASE DATASETS command.

The DUMP command makes a backup copy of a Model 204 file on a sequential data set at a particular time. This data set may either be stored on a magnetic tape or direct access device. The RESTORE command takes the backup sequential data set produced by the DUMP command and turns it into a Model 204 file again. These commands give the user a means of recovering from disk crashes, operating system crashes, and accidentally scratched data sets. In addition, it also allows the user to move the file from one device to another and to rename the file. The DUMP and RESTORE command can be used in batch program as shown in Appendix C. Further details can be found in the File Manager's Technical Reference Manual (19). The query procedures which are punched on cards can be loaded into a file with the batch program as shown in Appendix D. A copy of query procedures can also be made for backup by batch program as shown in

Appendix D.

Each Model 204 file has associated with it a number of parameters which determine its structure and keep track of its status and usage. There are also other parameters which control or describe the Model 204 system. The current values of any these parameters may be examined by the VIEW command. The values of these parameters may be changed by the RESET or UTABLE command. The DISPLAY command may be used to display sets of file related parameters. For further details see the Command Reference Manual (20).

To manipulate this question bank, the system needs some procedures to carry out the necessary operations. The main procedure provide all procedure options to let a user select the desired operation to execute as shown in Figure 7. The features of most of these operations are described in this section. Those operations having to do with test construction are described in the next section. Each box represents a whole screen on the terminal. The user responds to the message on the screen by entering values from the keyboard. The user may either use the TAB key to advance automatically from one input field to another, or use the terminal's cursor character to move around the screen. After all data has been entered on the screen, Model 204 checks the input data and redisplays the screen for correction if any errors are found. An error indicator, an asterisk (*), is set in column 80 of any screen line that contains an error.

```
 (After Logon to TSO)
>M204FS
```

```
 82.350  DEC 16  21.26.41          PAGE 2
>LOGIN U11338C
 ***  M204.0347: PASSWORD
>????
 ***  M204.0353: U11338C    LOGIN  82 DEC 16  21.28
>OPEN CSFORT
 ***  M204.0620: FILE CSFORT   OPENED
>I MAIN  (or INCLUDE MAIN)
 ***  M204.0620: FILE CSFORT   OPENED
```

```
  C  A  T  C    M A N A G E M E N T   S Y S T E M




 COMPUTER ASSISTED TEST CONSTRUCTION SYSTEM ( C A T C )

              FOR FORTRAN COURSE



C O M P U T I N G & I N F O R M A T I O N   S C I E N C E


    O K L A H O M A   S T A T E   U N I V E R S I T Y
```

Figure 7. (Continue Next Page)

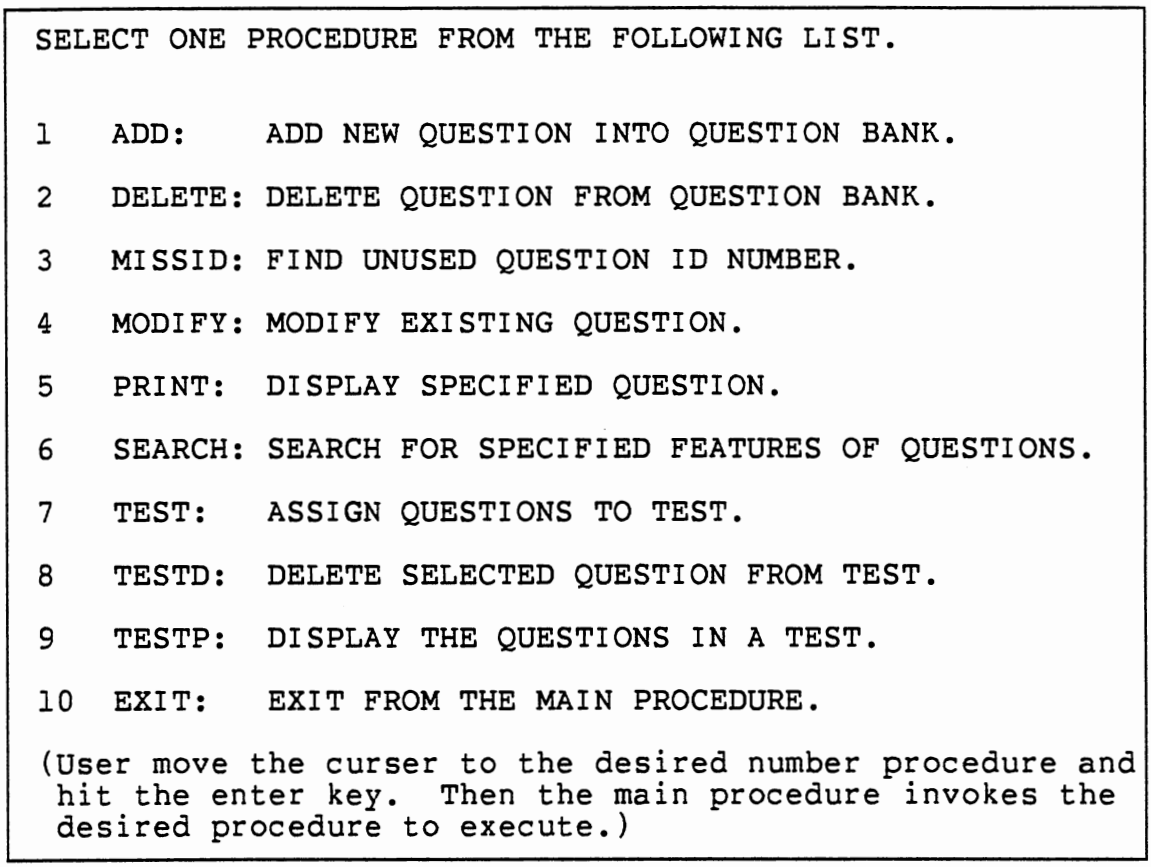


Figure 7. MAIN Procedure Screens.

ADD: This procedure provides the system with the ability to add new questions to the question bank. The procedure first responds with a whole screen of question features for the user to enter. Then it responds with a whole screen to let the user enter the question text. Finally, the procedure responds with a message to make sure the input data is correct and to check whether or not the user wants to enter another question. This procedure operates as shown in Figure 8.

```
ADD: ADD NEW QUESTION INTO QUESTION BANK.
IF YOU DON'T WANT TO ADD, PLEASE TYPE BREAK KEY.

PLEASE INPUT THE QUESTION FEATURES FIRST.

ENTER QUESTION ID NUMBER(FROM 1 TO 9999): ____

1. TRUE-FALSE     2. MULTIPLE CHOICE    3. COMPLETION
4. SYNTAX ERROR   5. ESSAY QUESTION
ENTER TYPE OF QUESTION(FROM 1 TO 5): _

ENTER CHAPTER(FROM 1 TO 99): __
ENTER SECTION(FROM 1 TO 99): __

ENTER DIFFICULTY LEVEL(FROM 1 TO 5): _

ENTER ESTIMATED FINISH TIME(FROM 1 TO 99): __

ENTER ANSWER KEY:_____( to the end of line)

DOES QUESTION HAVE SUBSTITUTE VARIABLES?
ENTER(IF YES ):_____( to the end of line)
```

Figure 8. (Continue Next Page)

```
ENTER THE WHOLE QUESTION TEXT.
       12345678901234567890123456789012345678901234567890
CON 1: _____________________________________________________
CON 2: _____________________________________________________
CON 3: _____________________________________________________
CON 4: _____________________________________________________
CON 5: _____________________________________________________
CON 6: _____________________________________________________
CON 7: _____________________________________________________
CON 8: _____________________________________________________
CON 9: _____________________________________________________
CON10: _____________________________________________________
CON11: _____________________________________________________
CON12: _____________________________________________________
CON13: _____________________________________________________
CON14: _____________________________________________________
CON15: _____________________________________________________
CON16: _____________________________________________________
CON17: _____________________________________________________
CON18: _____________________________________________________
CON19: _____________________________________________________
CON20: _____________________________________________________
ENTER( DO YOU NEED MORE SPACE? )( YES:Y, NO:<CR>): _
```

```
FILL IN DATA, THEN PRESS ENTER.
DO YOU REALLY WANT TO ADD THIS QUESTION TO THE BANK?
ENTER (YES:Y, NO:<CR>): _


IS THERE ANOTHER QUESTION TO ADD?
ENTER (YES:Y, NO:<CR>): _
```

```
  (Display of the new question.)
```

Figure 8. ADD Procedure Screens.

DELETE: This procedure provides the system with the ability to delete any specified question in the question bank. The procedure first lets the user enters the question ID number. To make sure it is the specified question to be deleted, the procedure displays this question. An error message is displayed if the question is not in the question bank. Next, the user decides whether or not to delete this question. Finally, the procedure checks whether or not to delete another question. This procedure operates as shown in Figure 9.

```
 $$PLEASE TYPE IN QUESTION ID NUMBER?
> 100

 ( Display the question. )

 $$DO YOU WANT TO DELETE THIS QUESTION?
>
 $$IS THERE ANOTHER QUESTION TO DELETE?
>
```

Figure 9. DELETE Procedure Screens.

MISSID: This procedure provides the system with the ability to find unused question ID numbers in a desired range. Unused question ID numbers can be used to add new questions to the question bank. An error message is displayed if the START VALUE is larger than the LAST VALUE, otherwise the search result is displayed and checks whether or not the user wants to search another range value. This procedure operates as shown in Figure 10.

```
SEARCH THE EXISTING FILE TO FIND THE UNUSED
QUESTION ID NUMBER IN THE DESIRED RANGE.

ENTER START VALUE(FROM 1 TO 9999): ____

ENTER LAST  VALUE(FROM 1 TO 9999): ____


DO YOU WANT TO LOOK AT ANOTHER RANGE OF VALUES?
ENTER (YES:Y, NO:<CR>): _
```

```
( Display of unused question ID numbers. )
```

Figure 10. MISSID Procedure Screens.

MODIFY: This procedure provides the system with the ability to modify the field value of the specified question in the bank. Once the user selects the name of the field to be modified, the procedure responds with the field name and lets the user enter the field value. Finally, the procedure responds with a message to make sure that the modification is correct, and checks whether or not the user wants to modify another question. This procedure operates as shown in Figure 11.

```
PLEASE INPUT THE QUESTION ID NUMBER.

ENTER QUESTION ID NUMBER(FROM 1 TO 9999): ____


DO YOU WANT TO DISPLAY THIS QUESTION?
ENTER (YES:Y, NO:<CR>): _
```

```
SELECT THE FIELD NAME FROM THE FOLLOWING LIST.

1. TYPE (QUESTION TYPE)

2. TOPIC (QUESTION TOPIC)

3. LEVEL (QUESTION DIFFICULTY LEVEL)

4. NO (QUESTION ID NUMBER)

5. FLAG (QUESTION HAS SUBSTITUTE ARRAY OR NOT)

6. TIME (QUESTION ESTIMATED TIME TO FINISH)

7. CONTENT (QUESTION TEXT)

8. SQ (QUESTION RECORD SEQUENCE NUMBER)

9. ANS (QUESTION ANSWER)

0. SUB (QUESTION SUBSTITUTE VARIABLES)
ENTER THE SELECTED FIELD NAME (FROM 0 TO 9): _
```

Figure 11. (Continue Next Page)

```
( The response screen will be different depending on
  the modify field name. )
```

```
 $$DO YOU WANT TO DISPLAY THIS QUESTION?
>
 $$IS THERE ANOTHER QUESTION TO MODIFY?
>
```

Figure 11. MODIFY Procedure Screens.

PRINT: This procedure provides the system with the ability to display the specified question in the bank. Before selecting this procedure, the user should know the question ID number. The procedure first prompts the user for the question ID number, and then displays the question. An error message is displayed if the specified question ID number is not in the question bank. Next, the procedure responds with a message to check whether or not the user wants to display another question. This procedure operates as shown in Figure 12.

```
 $$PLEASE TYPE IN THE QUESTION ID NUMBER?
>1
  TYPE       TOPIC LEVEL TIME FLAG NO
  TRUE/FALSE  1 1   1     3    1  1
     CONTENT
     _____A compiler is an example of hardware.
  ANS: F
  $$IS THERE ANOTHER QUESTION TO DISPLAY?
>
```

Figure 12. PRINT Procedure Screens

Automatic Test Construction

It is possible to construct and display tests using the procedures SEARCH, TEST, TESTD, and TESTP which are invoked by the MAIN procedure (see Fig. 7). These procedures are described in this section. In order to construct a test, the system adds set of 'test records' to the file. These records have a format which is different from the question records described in section 1 of this chapter. For each test there are several test records —— one for each question to be used on the test. The fields in a test record are described here.

The QNO field stores the ID number of a question which is to be included in the test.

The TEST field indicates the type of test; if its value is equal to 2 then it represents a quiz, otherwise it represents a test.

The TESTNO filed stores a number which uniquely identifies the test which this question belongs to.

The TSQ field stores the sequence order of a question in the test.

To search the questions which belong to the specified test, we need to specify the type of test (TEST) and the number of the test (TESTNO), so these two fields must be assigned with the KEY field attribute to allow direct access to the record. The automatic test construction procedures are described now.

SEARCH: This procedure provides the system with the ability to search for specified features of questions. This procedure issues prompts on the screen to let the user enter the specified question features. Then it displays the number of questions which satisfy the specified features, and it checks whether or not the user wants to display all of these questions or a random number of selected questions. Next, the procedure prompts to check whether or not the user wants to assign these questions to a test; if yes, then the assigning screen, which is similiar to the TEST procedure, is displayed. This procedure operates as shown in Figure 13.

```
SEARCH FOR SPECIFIED FEATURES OF QUESTIONS.
YOU CAN RELAX ANY FIELD VALUE BY NOT PROVIDING IT.

1. TRUE-FALSE
2. MULTIPLE CHOICE
3. COMPLETION
4. SYNTAX ERROR
5. ESSAY QUESTION
ENTER TYPE OF QUESTION(FROM 1 TO 5): _


SELECT CHAPTER AND SECTION, OR JUST THE CHAPTER FILED.
ENTER CHAPTER(FROM 1 TO 99): __
ENTER SECTION(FROM 1 TO 99): __

ENTER DIFFICULTY LEVEL(FROM 1 TO 5): _
```

```
??? QUESTIONS SATISFY THE CONDITIONS.
$$DO YOU WANT TO DISPLAY ALL QUESTIONS?
>
$$DO YOU WANT TO DISPLAY A SELECTED QUESTION AT RANDOM?
>
$$DO YOU WANT TO ASSIGN QUESTION TO TEST?
>
```

Figure 13. SEARCH Procedure Screens

TEST: This procedure provides the system with the ability to enter a specified question into a test. When executing this procedure, the user is prompted to enter the type and times of the test. Next, the user is prompted for the information required to enter a question into the test. A message is displayed if a specified question has been included in the test, or the specified sequence order has already been selected. It continues the process if the user has another question to enter. This procedure operates as shown in Figure 14.

```
PLEASE SELECT EITHER TEST OR QUIZ,
AND THE NUMBER OF TEST.

1. TEST
2. QUIZ
ENTER THE CHOICE(TEST:1, QUIZ:2 ): _


ENTER THE NUMBER OF TEST(FROM 1 TO 30) : __
```

```
PLEASE INPUT QUESTION ID NUMBER AND ORDER
SEQUENCE NUMBER IN THE TEST.

INPUT THE QUESTION ID NUMBER.
ENTER(FROM 1 TO 9999): ____

INPUT THE SEQUENCE ORDER IN THE TEST.
ENTER(FROM 1 TO 50): __


DO YOU WANT TO DISPLAY THIS QUESTION?
ENTER(YES:Y, NO:<CR>): _

IS THERE ANOTHER QUESTION TO SELECT?
ENTER(YES:Y, NO:<CR>): _
```

Figure 14. TEST Procedure Screens

TESTD: This procedure provides the system with the ability to delete a specified question from a test. The procedure requests the user to enter the type and times of the test for which a question is to be deleted. Next, the user is prompted for the information required to delete a question from the test. The user is asked whether or not he really want to delete the question. It continues the process if the user has another question to delete. This procedure operates as shown in Figure 15.

```
DELETE THE QUESTION FROM THE TEST.

PLEASE SELECT EITHER TEST OR QUIZ,
AND THE NUMBER OF TEST.

1. TEST
2. QUIZ
ENTER THE CHOICE(TEST:1, QUIZ:2 ): _


ENTER THE NUMBER OF TEST(FROM 1 TO 30) : __

PLEASE TYPE IN THE SEQUENCE NUMBER OF THE QUESTION
YOU WISH TO DELETE.
ENTER(FROM 1 TO 50): __

DO YOU WANT TO DISPLAY THIS QUESTION FIRST?
ENTER(YES:Y, NO:<CR>): _

IS THERE ANOTHER QUESTION TO DELETE?
ENTER(YES:Y, NO:<CR>): _
```

```
 $$DO YOU WANT TO DELETE THIS QUESTION FROM TEST?
>
```

Figure 15. TESTD Procedure Screens

TESTP: This procedure provides the system with the ability to display a specified test. The procedure automatically prompts the user to enter the type and times of the test to be displayed. Then the procedure checks whether or not the user wants to display all questions or a specific question. The user can also use the batch process in Appendix E to copy a specified test into a sequential data set, and use an editor to modify, delete, or add the question in the test. It continues the process if the user has another test to display. This procedure operates as shown in Figure 16.

```
PRINT OUT THE SPECIFIED TEST.

PLEASE SELECT EITHER TEST OR QUIZ,
AND THE NUMBER OF TEST.

1. TEST
2. QUIZ
ENTER THE CHOICE(TEST:1, QUIZ:2 ): _


ENTER THE NUMBER OF TEST(FROM 1 TO 30) : __
```

```
 THERE ARE ??? QUESTIONS IN THIS TEST.
 $$DO YOU WANT TO DISPLAY ALL THESE QUESTIONS?
>
 $$DO YOU WANT TO DISPLAY A SPECIFIC QUESTION?
>
 $$IS THERE ANOTHER TEST TO DISPLAY?
>
```

Figure 16. TESTP Procedure Screens

CHAPTER IV

CONCLUSION AND SUGGESTIONS

One of the painful realities of a CATC system is that it costs considerably more to build and update a high quality test question bank than it does to write computer programs that will use this question bank. In this system, we have standardized the input format which is used to record the questions produced; further modification capabilities to improve the quality of a question bank have also been provided. Hence, the overall quality of a question bank can be properly modified as a result of experience. Question classification still lacks standards today in existing CATC systems. It is essential to remember that the user does not want to be bogged down by confusing and unfamiliar classification indices. In this system, all questions were classified by chapter and subsection number to match the textbook used in the course. They were also evaluated by their difficulty level. The advantage of this classification system is that it allows for straightforward test specification procedures using those attributes with which the average teacher is familiar.

The operations described in this report permit the interactive construction of tests and the manipulation of

test questions at a computer terminal. The system also permits a test constructor to search through questions rapidly and without the clerical errors frequently involved in manual searches.

Major features of this CATC system design have been verified, but the full system is not yet implemented. It is expected that the complete system will be implemented in the near future. No data is available on the total computer system space requirements nor on the execution time characteristics of this system. Nonetheless, some characteristics of the system are sufficiently established so that it does not seem premature to offer some evaluation of them. This system will share principal characteristics of all CATC systems in freeing the instructor from major time demands in examination preparation. Although this system requires more computer storage than it would be if stored sequentially, it provides very flexible data organization and very rapid retrieval response for online queries.

In the future, a CATC system without the ability to collect question statistics will not be a sound system, as it will lack the feedback loop necessary for maintaining the question bank. An online automated test scoring system can be developed based on this system. Since the answer key goes along with each question in the question bank, statistics can be accumulated during scoring to improve the

quality of the question bank, to analyze student performance, and to report test statistics.

Traditionally, the instructor provides all students with the same instruction for the same length of time. Computer Assisted Instruction(CAI) makes it feasible to vary the length of time needed for learning for different students according to the individual's pace. The current CATC system can be combined with this CAI system to provide a means, by developing another online procedure to give the desired search criteria, for students to demonstrate proficiency in special topics.

There is no doubt about the final success of CATC systems. However, their usefulness will continue to be hampered until the computer becomes as reliable as the department secretary or the office typewriter, until teachers place the same trust in the computer to produce tests as they do in a textbook to produce ideas for test questions.

BIBLIOGRAPHY

(1) Lippey, G. Computer-Assisted Test Construction. Educational Publications, Englewood Cliffs, New Jersey, 1974.

(2) Heines, J. M. An Examination of the Literature on Criterion-Referenced and Computer-Assisted Testing. Proceedings of 14th Annual Convention of the Association for Data Base Systems (1976), 15-44.

(3) Salisnjak, J. Computer Aided Test Preparation: Six Years of Experience. Educational Technology, vol 13, 3(March 1973), 37-38.

(4) Bailey, T. E. Unpublished Description of COMSC 2113 Test Generator, Department of Computing and Information Sciences at Oklahoma State Unversity, 1978.

(5) Prosser, F. Repeatable Tests. Educational Technology, vol 13, 3(March 1973), 34-35.

(6) Libaw, F. B. Constructing Tests With the MENTREX Tutorial Testing System. Educational Technology, vol 13, 3(March 1973), 30-31.

(7) Toggenburger, F. Classroom Teacher Support System. Educational Technology, vol 13, 3(March 1973), 42-43.

(8) Lippey, G. Classroom Teacher Support System. Proceedings of 14th Annual Convention of the Association for Data Base Systems (1976), 363-364.

(9) Baker, F. B. An Interactive Approach to Test Construction. Educational Technology, vol 13, 3(March 1973), 13-15.

(10) Epstein, M. G. Computer Assisted Assembly of Tests at Educational Testing Service. Educational Technology, vol 13, 3(March 1973), 23-24.

(11) Geisler, R. G. SOCRATES: An Interactive Student Oriented Computer Aided Test Generator. Proceedings of the EDUCOM Fall Conference (1974), 167-172.

(12) Seely, O. Jr. and W. V. Willis. SOCRATES Test Retrieval at the California State University and Colleges. 1975 Conference on Computers in the Undergraduate Curricula, 135-138.

(13) Gray-Shellberg, L., W. V. Willis, and O. Seely, Jr. The Consolidatation of SOCRATES: Growing Pains in a Large Computerized Test Generation Network. Proceedings of 14th Annual Convention of the Association for Data Base Systems (1976), 338-342.

(14) Willis, W. V. Computer Assisted Test Construction: Two Years of User Experience with the SCORATES System. Growing Pains in a Large Computerized Test Generation Network. Proceedings of 16th Annual Convention of the Association for Data Base Systems (1978), 315-317.

(15) Johnson, K. J., W. V. Willis, J. W. Moore, and O. Seely, Jr. Computer-Assisted Test Construction in Chemistry. Journal of Chemical Education Vol. 58, 2(February 1981), 177-181.

(16) Collins, R. W. and S. J. Duff. PROBGEN: An Interactive Computer Program for Faculty Use in Creating and Generating Individualized, Non-Indentical Problem Sets and Tests. Ninth Conference on Computers in the Undergraduate Curricula (1978), 100-116.

(17) Collins, R. W. and S. J. Duff. Computer-Assisted Test Construction Via Automatic Program Generation: Using PROBGEN II to create Individualized Exams and Problem Sets. Proceedings of 1979 National Educational Computing Conference, 114-129.

(18) Collins, R. W. and S. J. Duff. Automatic Programming: An Educator's Aid for Computer-Assisted Test Construction (CATC). Proceedings of 17th Annual Convention of the Association for Data Base Systems (1979), 51-57

(19) Model 204 Database Management System, File Manager's Technical Reference Manual (1981). Computer Corporation of America, Database Products Division, 675 Massachusetts Avenue, Cambridge, MA 02139

(20) Model 204 Database Management System, Command Reference Manual (1981). Computer Corporation of America, Database Products Division, 675 Massachusetts Avenue, Cambridge, MA 02139

APPENDIX A

MEANINGS OF ATTRIBUTES

(A) Functional Attributes

These field attributes determine how a field can be used in the query language and how they affect retrieval speed in accessing the records.

KEY / NONKEY

An index entry is made for a field assigned with KEY attribute, but no index entry is made for NON-KEY.

VISIBLE / INVISIBLE

An index entry is made for a field assigned with the INVISIBLE attribute. Such fields must also have either the KEY or NUMERIC RANGE attribute and are used only to retrieve records. The INVISIBLE field takes up no storage space in TABLE B, since it is a part of the VISIBLE field which is stored in TABLE B.

NUMERIC RANGE / NON-RANGE

An index entry is made for a field assigned with the NUMERIC RANGE attribut e, but no index entry is made for NON-RANGE.

FOR EACH VALUE(FRV) / NON-FRV

When a field is to be used as a for-each-value loop in retrieving, it must be assigned with FRV attribute. This can be used to avoid sorting a large numbers of records online. A field assigned with FRV attribute must have the KEY attribute as well. A field assigned with a NON-FRV attribute can not be used in a value loop.

DEFERABLE / NON-DEFERABLE

When storing or updating records in the file, the field assigned with the DEFERABLE attribute can be deferred to update the index entry. The NON-DEFERABLE field can not be deferred to update the index entry. The DEFERABLE and NON-DEFERABLE attributes are invalid for a field that is both NON-KEY and NON-RANGE.

LEVEL

A field may be secured against unauthorized access by including a LEVEL clause in the field's description.

(B) Representation Attributes

These field attributes determine how a field is physically stored in the Model 204 file. Each field value stored in TABLE B is in one of three formats depending upon the selection of STRING/BINARY or CODED/NON-CODED field attribiutes. The choice affects space requirements and the time required for updates.

CODED / NON-CODED

When a field is defined with a CODED attribute, its field value is stored in TABLE A and a four-byte value code pointing to that character string is stored in the logical record in TABLE B. For a NON-CODED field attribute its actual field value is stored in TABLE B.

BINARY / STRING

When a field is assigned with the STRING attribute, the field value is stored as a character string with an additional byte to indicate its length. The BINARY field attribute stores decimal integers of one to nine digits as fourbyte binary numbers.

MANY-VALUED / FEW-VALUED

When a field is assigned with the CODED or FRV attribute,the MANY-VALUED or FEW-VALUED option affects only where the value or string is stored in TABLE A. It is invalid for fields which are neither CODED nor FRV.

UPDATE IN PLACE / UPDATE AT END

When a field is changed, the UPDATE IN PLACE attribute will store the new value in the same position; the UPDATE AT END attribute will delete the existing field value and add a new one as the last occurrence in a record.

OCCURS

The OCCURS attribute specifies the number of occurrences of the field that will be prelocated in each TABLE B record.

LENGTH

The LENGTH attribute indicates the maximum length of the field and may be specified only for a field that includes the OCCURS attribute.

PAD

The PAD attribute is used to select the character that will be used to pad field values that are shorter than the length specified in the LENGTH attribute.

APPENDIX B

LOAD PROGRAM

```
//  EXEC M204FLOD
//CSFORT DD DSN=M204.ACT11338.FORTBASE,DISP=SHR,
//    DCB=(RECFM=U,LRECL=0,BLKSIZE=6184),UNIT=3350,
//     VOL=SER=SYSTSO,SPACE=(CYL,(1,1))
//TAPEI DD DSN=U11338C.DATABASE.DATA,DISP=SHR
//CCAIN DD *
*DEFINE PAGE SIZE, WORKING AREA, AND NONDEFAULT PARAMETER
PAGESZ=6184,SPCORE=100000,INMRL=255
*CREATE THE MODEL 204 FILENAME
CREATE FILE CSFORT
*DEFINE THE FIEL PARAMETERS AND FILE SIZES
PARAMETER ASTRPPG=614,ATRPG=1,FVFPG=1,MVFPG=1
PARAMETER BRESERVE=491,BRECPPG=14,BSIZE=20
PARAMETER CSIZE=5
PARAMETER PDSTRPPG=204,PDSIZE=1,DSIZE=35
END
OPEN CSFORT
*ERASE ALL INFORMATION STORED IN THE FILE,
*             EXCEPT THE ABOVE FILE PARAMETERS.
INITIALIZE
*DEFINE THE FIELD NAMES AND ITS ATTRIBUTES
DEFINE TYPE          ( KEY )
DEFINE LEVEL         ( KEY )
DEFINE TOPIC         ( KEY )
DEFINE CHAPTER       ( KEY INVISIBLE )
DEFINE NO            ( KEY )
DEFINE TIME
DEFINE FLAG
DEFINE ANS
DEFINE SUB
DEFINE CONTENT
DEFINE SQ
DEFINE TEST          ( KEY )
DEFINE TESTNO        ( KEY )
DEFINE QNO
DEFINE TSQ
*READ THE INPUT DATA WHILE IT EXIST.
FILELOAD -1,-1,0
*GET THE INPUT DATA
G
*ECHO PRINT THE INPUT DATA
P 1,255
```

```
*STORE THE EXTENT QUESTION TEXT BRANCH TO #10
=10,5,
*STORE THE FIRST QUESTION TEXT BRANCH TO #20
=20,5,#
*STORE THE QUESTION FEATURES
 TYPE=6,2,x'8000'
 TOPIC=8,4
 CHAPTER=8,2
 LEVEL=12,2
 TIME=14,2
 FLAG=17,1
 ANS=54,250
*STORE THE QUESTION ID IN THE BUFFER STRING 0
S 0,1,4
*IF QUESTION DOES NOT HAVE SUBSTITUTE
*                          VARIABLES BRANCH TO 40
=40,17,1
 SUB=18,53
*GET THE NEXT RECORD
=40
#10
*ASSIGN THE NO FIELD START THE NEW RECORD
 NO=1|0S,0|0S,X'8100'
=30
#20
*ASSIGN THE NO FIELD ALONG WITH THE TOPIC, TYPE..
 NO=1|0S,0|0S,X'0100'
#30
 CONTENT=6,250
 SQ=1,4,X'0100'
#40
END
EOJ
//
```

APPENDIX C

DUMP AND RESTORE FILE

```
//   EXEC M204FLOD
//CSFORT DD DSN=M204.ACT11338.FORTBASE,DISP=SHR
//DUMPCSF DD DSN=U11338C.DUMP.CSFORT,DISP=(NEW,CATLG),
//           VOL=SER=DASD80,SPACE=(TRK,(15,2)),UNIT=3350
//CCAIN DD *
PAGESZ=6184
OPEN CSFORT
DUMP TO DUMPCSF
EOJ
//
```

```
//   EXEC M204FLOD
//CSFORT DD DSN=M204.ACT11338.FORTBASE,DISP=SHR
//DUMPCSF DD DSN=U11338C.DUMP.CSFORT,DISP=SHR
//CCAIN DD *
PAGESZ=6184
OPEN CSFORT
RESTORE FROM DUMPCSF
EOJ
//
```

APPENDIX D

PROCEDURE LOAD AND BACKUP

```
//    EXEC M204FLOD
//CSFORT DD DSN=M204.ACT11338.FORTBASE,DISP=SHR
//OUTPROC DD DSN=U11338C.OUTPROC.data,DISP=SHR
//CCAIN DD *
SPCORE=10000,PAGESZ=6184
OPEN CSFORT
RESET UDDLPP=0,UDDCCC=80
USE OUTPROC
DISPLAY (ALIAS,LABEL) ALL
EOJ
//
```

```
//    EXEC M204FLOD,PARM='INCCC=80,SYSOPT=192'
//CSFORT DD DSN=M204.ACT11338.FORTBASE,DISP=SHR
//CCAIN DD *
PAGESZ=6184
OPEN CSFORT
/*
//       DD DSN=U11338C.OUTPROC.DATA,DISP=SHR
//       DD *
EOJ
//
```

APPENDIX E

DISPLAY A TEST TO A FILE

```
//   EXEC M204FLOD
//CSFORT DD DSN=M204.ACT11338.FORTBASE,DISP=SHR
//OUTTEST DD DSN=U11338C.TEST1.DATA,DISP=SHR
//CCAIN DD *
SPCORE=10000,PAGESZ=6184
OPEN CSFORT
RESET UDDLPP=0,UDDCCC=80
USE OUTTEST
**
** CHANGE THE VALUE IN STATEMENT 1, THE PROGRAM WILL
**  DISPLAY DIFFERENT TEST.
**
**      TEST   :  1 --  TEST.
**                2 --  QUIZ.
**
**      TESTNO:   THE SERIES NUMBER OF A TEST.
**
BEGIN
%PFLAG = 'Y'
1. FIND ALL RECORDS FOR WHICH TEST = 1 AND TESTNO = 1
2. SORT RECORDS IN 1 BY TSQ VALUE RIGHT-ADJUSTED
3. FOR EACH RECORDS IN 2
 3.1 %NO = QNO
 3.2 FIND ALL RECORDS FOR WHICH NO = %NO
 3.3 CLEAR LIST PRINTLIST
 3.4 PLACE RECORDS IN 3.2 ON LIST PRINTLIST
 3.5 PRINT 'SEQUENCE=' TSQ
 3.6 CALL 99
99. SUBROUTINE
  INCLUDE PRINT1
END
//
```

APPENDIX F

MAIN PROCEDURE

```
*************************************************************
*                                                           *
*  MAIN: THE MAIN PROCEDURE DRIVE THE DESIRED OPERATION.    *
*                                                           *
*    ADD    : ADD NEW QUESTION INTO BANK.                   *
*    DELETE: DELETE QUESTION FROM BANK.                     *
*    MISSID: FIND UNUSED QUESTION ID NUMBER.                *
*    MODIFY: MODIFY QUESTION IN THE BANK.                   *
*    PRINT : DISPLAY SPECIFIED QUESTION.                    *
*    SEARCH: SEARCH FOR SPECIFIED FEATURES OF QUESTION.     *
*    TEST   : ASSIGN QUESTION TO TEST.                      *
*    TESTD : DELETE SELECTED QUESTION FROM THE TEST.        *
*    TESTP : DISPLAY THE QUESTION IN A TEST.                *
*                                                           *
*************************************************************
```

```
OPEN CSFORT
INCLUDE START
*************************************************************
*      EXECUTE THE INITIALIZE PROCEDURE                     *
*          TO SET USER PARAMETERS.                          *
*************************************************************
BEGIN
*************************************************************
*  DEFINE MENU.                                             *
*************************************************************
MENU SELECT
TITLE 'SELECT ONE PROCEDURE FROM THE FOLLOWING LIST.'
SKIP 2 LINES
PROMPT 'ADD:    ADD NEW QUESTION INTO QUESTION BANK.'
SKIP 1 LINES
PROMPT 'DELETE: DELETE QUESTION FROM QUESTION BANK.'
SKIP 1 LINES
PROMPT 'MISSID: FIND UNUSED QUESTION ID NUMBER.'
SKIP 1 LINES
PROMPT 'MODIFY: MODIFY EXISTING QUESTION.'
SKIP 1 LINES
PROMPT 'PRINT:  DISPLAY SPECIFIED QUESTION.'
SKIP 1 LINES
PROMPT 'SEARCH: SEARCH FOR SPECIFIED FEATURES ' -
PROMPT 'OF QUESTIONS.'
SKIP 1 LINES
```

```
PROMPT 'TEST:   ASSIGN QUESTIONS TO TEST.'
SKIP 1 LINES
PROMPT 'TESTD:  DELETE SELECTEC QUESTION FROM TEST.'
SKIP 1 LINES
PROMPT 'TESTP:  DISPLAY QUESTIONS IN A TEST.'
SKIP 1 LINES
PROMPT 'EXIT:   EXIT FROM THE MAIN PROCEDURE.'
END MENU
****** END MENU. ********
*************************************************************
*  READ MENU                                                *
*************************************************************
READ MENU SELECT
IF $SETG('SELECTION',%SELECT:SELECTION) THEN STOP
END
*************************************************************
*  ACCORDING TO DIFFERENT SELECTION,                        *
*  EXECUTE THE DESIRED PROCEDURE.                           *
*************************************************************
IF SELECTION=1,ADD
IF SELECTION=2,DELETE
IF SELECTION=3,MISSID
IF SELECTION=4,MODIFY
IF SELECTION=5,PRINT
IF SELECTION=6,Search
IF SELECTION=7,TEST
IF SELECTION=8,TESTD
IF SELECTION=9,TESTP
```

APPENDIX G

ADD PROCEDURE

```
*************************************************************
*                                                           *
*  ADD: ADD THE NEW QUESTION INTO QUESTION BANK.            *
*                                                           *
*= P = = D = = L = = = = D = = E = = S = = I = = G = = N = *
*   DEFINE SCREEN.                                          *
*   DEFINE VARIABLES.                                       *
*   SET PRINT FLAG TO POSTIVE.                              *
*    1.CLEAR LIST PRINTLIST.                                *
*    2.READ THE SCREEN OF QUESTION FEATURES.                *
*    3.-9.CHECK THAT THERE IS NO NULL STRING IN INPUT       *
*       DATA.  IF THERE IS, REREAD SCREEN.                  *
*   10.IF THERE IS SUBSTITUTE VARIABLES SET THE FLAG.      *
*   11.READ THE QUESTION TEXT.                             *
*   12.-16.CHECK, THE INPUT ID QUESTION EXIST OR NOT.       *
*   17.INITIALIZE THE QUESTION TEXT VARIABLES.              *
*   18.CONCATENATE THE TOPIC FROM CHAPTER AND SECTION.      *
*   19.CONCATENATE THE QUESTION TEXT.                       *
*   20.IF NEED MORE SPACE TO STORE THE QUESTION TEXT,       *
*      THEN READ SCREEN ADDE AGAIN AND ASSIGN TO TEXT       *
*      VARIABLES.                                           *
*   21.READ SCREEN CHECK, CHECK THAT INPUT DATA IS CORRECT *
*      AND IS THERE ANOTHER QUESTION TO ADD.                *
*   22.IF INPUT QUESTION IS NOT CORRECT THEN JUMP TO 33.   *
*   23.STORE THE FIRST QUESTION TEXT.                       *
*   24.INCREMENT THE SEQUENCE INDEX.                        *
*   25.IF NO MORE QUESTION TEXT THEN JUMP TO 27.            *
*   26.STORE THE EXTENDED QUESTION TEXT.                    *
*   27.IF SEQUENCE INDEX IS LESS THAN 8 THEN JUMP TO 24.   *
*   28. PRINT THE QUESTION TEXT TOO LONG MESSAGE.           *
*   29.--32.DISPLAY THE NEW QUESTION.                       *
*   33.IS THERE MORE QUESTION TO ADD THEN JUMP TO 1.        *
*   34.PRINT THE FINISH MESSAGE.                            *
*                                                           *
*************************************************************
```

```
BEGIN
******** DEFINE THE SCREEN ********
SCREEN ADD
TITLE 'ADD: ADD NEW QUESTION INTO QUESTION BANK.'
PROMPT 'IF YOU DON"T WANT TO ADD, PLEASE TYPE BREAK KEY.'
SKIP 1 LINE
```

```
PROMPT 'PLEASE INPUT THE QUESTION FEATURES FIRST.'
SKIP 1 LINE
PROMPT 'ENTER QUESTION ID NUMBER(FROM 1 TO 9999):' -
              INPUT NO LEN  4  NUMERIC RANGE 1 TO 9999
SKIP 1 LINE
PROMPT '1. TRUE-FALSE              2. MULTIPLE CHOICE' -
PROMPT '          3. COMPLETION'
PROMPT '4. SYNTAX ERROR            5. ESSAY QUESTION'
PROMPT 'ENTER TYPE OF QUESTION' -
PROMPT '(FROM 1 TO 5):' INPUT LFN LEN 1 ONEOF 1,2,3,4,5
SKIP 1 LINE
PROMPT 'ENTER CHAPTER(FROM 1 TO 99):' INPUT CHAPTER -
                 LEN 2 VERIFY '1234567890 '
PROMPT 'ENTER SECTION(FROM 1 TO 99):' INPUT SECTION -
                 LEN 2 VERIFY '1234567890 '
SKIP 1 LINE
PROMPT 'ENTER DIFFICULTY LEVEL(FROM 1 TO 5):' INPUT -
               LEVEL LEN  1 NUMERIC RANGE 1 TO 5
SKIP 1 LINE
PROMPT 'ENTER ESTIMATED FINISH TIME(FROM 1 TO 99):'  -
          INPUT TIME  LEN  2 NUMERIC RANGE 1 TO 99
SKIP 1 LINE
PROMPT 'ENTER ANSWER KEY:'   INPUT ANS   LEN 74
SKIP 1 LINE
PROMPT 'DOES QUESTION HAVE SUBSTITUTE VARIABLES?'
PROMPT 'ENTER(IF YES ):' INPUT SUB LEN 60
END SCREEN
******** DEFINE THE SCREEN ********
SCREEN ADDE
TITLE  'ENTER THE WHOLE QUESTION.'
PROMPT -
 '12345678901234567890123456789012345678901234567890' -
 AT COLUMN 9
PROMPT 'CON 1:' INPUT CONT1  LEN 50
PROMPT 'CON 2:' INPUT CONT2  LEN 50
PROMPT 'CON 3:' INPUT CONT3  LEN 50
PROMPT 'CON 4:' INPUT CONT4  LEN 50
PROMPT 'CON 5:' INPUT CONT5  LEN 50
PROMPT 'CON 6:' INPUT CONT6  LEN 50
PROMPT 'CON 7:' INPUT CONT7  LEN 50
PROMPT 'CON 8:' INPUT CONT8  LEN 50
PROMPT 'CON 9:' INPUT CONT9  LEN 50
PROMPT 'CON10:' INPUT CONT10 LEN 50
PROMPT 'CON11:' INPUT CONT11 LEN 50
PROMPT 'CON12:' INPUT CONT12 LEN 50
PROMPT 'CON13:' INPUT CONT13 LEN 50
PROMPT 'CON14:' INPUT CONT14 LEN 50
PROMPT 'CON15:' INPUT CONT15 LEN 50
PROMPT 'CON16:' INPUT CONT16 LEN 50
PROMPT 'CON17:' INPUT CONT17 LEN 50
PROMPT 'CON18:' INPUT CONT18 LEN 50
PROMPT 'CON19:' INPUT CONT19 LEN 50
PROMPT 'CON20:' INPUT CONT20 LEN 50
PROMPT 'ENTER(DO YOU NEED MORE SPACE TO STORE?)' -
```

```
PROMPT '( YES:Y, NO:<CR>):' INPUT SPACE LEN 1
END SCREEN
******** DEFINE THE SCREEN ********
SCREEN CHECK
PROMPT 'DO YOU REALLY WANT TO ADD THEIS QUESTION TO ' -
PROMPT 'THE BANK?'
PROMPT 'ENTER (YES:Y, NO:<CR>):' INPUT PRINT LEN 1
SKIP 2 LINES
PROMPT 'IS THERE ANOTHER QUESTION TO ADD?'
PROMPT 'ENTER (YES:Y, NO:<CR>):' INPUT YES LEN 1
END SCREEN
******** END DEFINE SCREEN ********
%CONTENT IS LEN 250
%SUB     IS LEN 250
%IN      IS FIXED
%TEXT   IS LEN 250 ARRAY(8)
%PFLAG = 'Y'
%SFLAG = '1'
1. CLEAR LIST PRINTLIST
2. READ SCREEN ADD
3.IF %ADD:NO EQ '' THEN
  3.1 TAG %ADD:NO
  3.2 REREAD SCREEN ADD
4. IF %ADD:LFN EQ '' THEN
  4.1 TAG %ADD:LFN
  4.2 REREAD SCREEN ADD
5. IF %ADD:SECTION EQ '' THEN
  5.1 TAG %ADD:SECTION
  5.2 REREAD SCREEN ADD
6. IF %ADD:CHAPTER EQ '' THEN
  6.1 TAG %ADD:CHAPTER
  6.2 REREAD SCREEN ADD
7. IF %ADD:LEVEL EQ '' THEN
  7.1 TAG %ADD:LEVEL
  7.2 REREAD SCREEN ADD
8. IF %ADD:TIME EQ '' THEN
  8.1 TAG %ADD:TIME
  8.2 REREAD SCREEN ADD
9. IF %ADD:ANS EQ '' THEN
  9.1 TAG %ADD:ANS
  9.2 REREAD SCREEN ADD
10. IF %ADD:SUB NE '' THEN %SFLAG = '0'
11. READ SCREEN ADDE
12. %NO = %ADD:NO
    %IN = 1
13. FIND ALL RECORDS FOR WHICH NO=%NO
14. COUNT RECORDS IN 13
15. %COUNTA = COUNT IN 14
16. IF %COUNTA NE 0 THEN
 16.1 PRINT 'THERE HAS QUESTION EXIST WITH' AND -
              'THIS ID NUMBER' AND %NO
 16.2 %X=$READ('DO YOU REALLY WANT TO OVERRIDE IT?')
 16.3 IF %X NE 'Y' THEN JUMP TO 1
 16.4 FIND ALL RECORDS FOR WHICH NO = %NO
```

```
 16.5 FOR EACH RECORD IN 16.4
   16.5.1 DELETE RECORD
17. %TEXT(1) = ''
    %TEXT(2) = ''
    %TEXT(3) = ''
    %TEXT(4) = ''
    %TEXT(5) = ''
    %TEXT(6) = ''
    %TEXT(7) = ''
    %TEXT(8) = ''
18. %TOP=%ADD:CHAPTER WITH %ADD:SECTION
    IF $LEN(%ADD:SECTION) EQ 1 THEN -
       %TOP = %ADD:CHAPTER WITH ' ' WITH %ADD:SECTION
19. %TEXT(1) = %ADDE:CONT1 WITH %ADDE:CONT2 WITH -
      %ADDE:CONT3 WITH %ADDE:CONT4 WITH %ADDE:CONT5
    %TEXT(2) = %ADDE:CONT6 WITH %ADDE:CONT7 WITH -
      %ADDE:CONT8 WITH %ADDE:CONT9 WITH %ADDE:CONT10
    %TEXT(3) = %ADDE:CONT11 WITH %ADDE:CONT12 WITH -
      %ADDE:CONT13 WITH %ADDE:CONT14 WITH %ADDE:CONT15
    %TEXT(4) = %ADDE:CONT16 WITH %ADDE:CONT17 WITH -
      %ADDE:CONT18 WITH %ADDE:CONT19 WITH %ADDE:CONT20
20. IF %ADDE:SPACE EQ 'Y' THEN
 20.1 READ SCREEN ADDE
 20.2 %TEXT(5) = %ADDE:CONT1 WITH %ADDE:CONT2 WITH -
        %ADDE:CONT3 WITH %ADDE:CONT4 WITH %ADDE:CONT5
 20.3 %TEXT(6) = %ADDE:CONT6 WITH %ADDE:CONT7 WITH -
        %ADDE:CONT8 WITH %ADDE:CONT9 WITH %ADDE:CONT10
 20.4 %TEXT(7) = %ADDE:CONT11 WITH %ADDE:CONT12 WITH -
        %ADDE:CONT13 WITH %ADDE:CONT14 WITH %ADDE:CONT15
 20.5 %TEXT(8) = %ADDE:CONT16 WITH %ADDE:CONT17 WITH -
        %ADDE:CONT18 WITH %ADDE:CONT19 WITH %ADDE:CONT20
21. READ SCREEN CHECK
22. IF %CHECK:PRINT NE 'Y' THEN JUMP TO 33
23. STORE RECORD
     TYPE    = %ADD:LFN
     TOPIC   = %TOP
     LEVEL   = %ADD:LEVEL
     FLAG    = %SFLAG
     TIME    = %ADD:TIME
     NO      = %ADD:NO
     ANS     = %ADD:ANS
     SQ      = 1
     CONTENT = %TEXT(1)
     SUB     = %ADD:SUB
24. %IN = %IN+1
25. IF $LEN(%TEXT(%IN)) EQ 0 THEN JUMP TO 27
26. STORE RECORD
     NO = %NO
     SQ = %IN
     CONTENT = %TEXT(%IN)
27. IF %IN LT 8 THEN JUMP TO 29
28. PRINT 'QUESTION TEXT IS TOO LONG.'
29. FIND ALL RECORDS FOR WHICH NO = %NO
30. PLACE RECORDS IN 29 ON LIST PRINTLIST
```

```
 31. call 99
 32. CLEAR LIST PRINTLIST
 33. IF %CHECK:YES EQ 'Y' THEN JUMP TO 1
 34. PRINT '***** FINISH THE ADD QUESTION PROCEDURE'
*************************************************************
*   SUBROUTINE TO PRINT THE DESIRED QUESTION.               *
*************************************************************
 99. SUBROUTINE
   INCLUDE PRINT1
END
```

APPENDIX H

DELETE PROCEDURE

```
*************************************************************
*                                                           *
*  DELETE:  DELETE THE QUESTION FROM THE BANK.              *
*                                                           *
*= P = = D = = L = = = = D = = E = = S = = I = = G = = N = *
*     DEFINE VARIABLES.                                     *
*     SET THE PRINT FLAG TO POSITIVE.                       *
*     1.INPUT THE QUESTION ID NUMBER.                       *
*     2.FIND ALL RECORDS WITH DESIRED ID NUMBER.            *
*     3.PLACE ALL FIND RECORDS ON LIST PRINTLIST.           *
*     4.DISPLAY THE QUESTION WHICH WANT TO DELETE.          *
*     7.CLEAR LIST PRINTLIST.                               *
*     8.--10.CHECK, THE DELETE QUESTION HAS INCLUDED        *
*       IN TEST OR NOT.                                     *
*    11.IF TRUE, PRINT THE QUESTION HAS INCLUDE IN TEST.    *
*    12.CHECK, DO YOU WANT TO DELETE THIS QUESTION.         *
*    13.IF TRUE, THEN DELETE THIS QUESTION.                 *
*    14.CHECK, IS THERE ANOTHER QUESTION TO DELETE.         *
*    15.IF TRUE, THEN JUMP TO 1.                            *
*    16.PRINT THE FINISH MESSAGE.                           *
*                                                           *
*************************************************************
```

```
BEGIN
%COUNT1 IS FIXED
%PFLAG = 'N'
 1. %NO = $READ('PLEASE TYPE IN THE QUESTION ID NUMBER?')
 2. FIND ALL RECORDS FOR WHICH NO = %NO
 3. PLACE RECORDS IN 2 ON LIST PRINTLIST
 4. CALL 99
    IF %ERROR EQ 'Y' THEN JUMP TO 14
 5. %X = $READ('DO YOU WANT TO DISPLAY THIS QUESTION?')
 6. IF %X EQ 'Y' THEN
   6.1 %PFLAG = 'Y'
   6.2 CALL 99
   6.3 %PFLAG = 'N'
 7. CLEAR LIST PRINTLIST
 8. FIND ALL RECORDS FOR WHICH QNO = %NO
 9. COUNT RECORDS IN 8
10. %COUNT1 = COUNT IN 9
11. IF %COUNT1 GT 0 THEN
 11.1 PRINT 'SOME TEST HAS INCLUDE THIS QUESTION,' -
```

```
            AND 'PLEASE MODIFY THE TEST FIRST.'
 11.2 FOR EACH RECORD IN 8
   11.2.1 PRINT 'TIMES=' AND TNO AND -
                'SEQUENCE=' AND TSQ
 11.3 JUMP TO 14
12. %X = $READ('DO YOU WANT TO DELETE THIS QUESTION?')
13. IF %X EQ 'Y' THEN
 13.1 FOR EACH RECORD IN 2
   13.1.1 IF $LEN(TOPIC) EQ 4 THEN
                %CHAP = $SUBSTR(TOPIC,1,2)
                DELETE CHAPTER = %CHAP
   13.1.2 IF $LEN(TOPIC) EQ 3 THEN
                %CHAP = $SUBSTR(TOPIC,1,1)
                DELETE CHAPTER = %CHAP
   13.1.3 DELETE RECORD
 13.2 DELETE RECORDS IN 2
14. %X = $READ('IS THERE ANOTHER QUESTION TO DELETE?')
15. IF %X EQ 'Y' THEN JUMP TO 1
16. PRINT '***** FINISH THE DELETE QUESTION PROCEDURE'
*************************************************************
*   SUBROUTINE TO PRINT THE DESIRED QUESTION.               *
*************************************************************
99. SUBROUTINE
  INCLUDE PRINT1
END
```

APPENDIX I

MISSID PROCEDURE

```
*************************************************************
*                                                           *
*  MISSID: FIND THE UNUSED QUESTION ID NUMBER IN THE        *
*          PROVIDED RANGE.  THESE ID NUMBER CAN BE USED     *
*          TO ADD NEW QUESTION TO THE BANK.                 *
*                                                           *
*= P = = D = = L = = = = D = = E = = S = = I = = G = = N = *
*  DEFINE SCREEN.                                           *
*  DEFINE VARIABLES.                                        *
*  1.READ INPUT SCREEN.                                     *
*  2.GET THE RANGE OF VALUE.                                *
*    SET THE NEGATIVE FLAG.                                 *
*  3.CHECK, THE INPUT VALUE IS VALID OR NOT.                *
*    IF INVALID, PRINT ERROR MESSAGE.                       *
*                JUMP TO CHECK CONTINUE OR EXIT.            *
*  4.FOR EACH QUESTION ID NUMBER IN THE RANGE.              *
*     FIND THAT THE QUESTION EXIST IN THE BANK OR NOT.      *
*     IF NOT EXIST, PRINT QUESTION ID NUMBER.               *
*                   SET POSITIVE FLAG.                      *
*  5.IF THERE IS NO UNUSING QUESTION ID NUMBER IN           *
*     THE DESIRED RANGE, PRINT THE MESSAGE.                 *
*  6.CHECK, CONTINUE TO FIND ANOTHER RANGE OR EXIT.         *
*  7.IF CONTINUE, JUMP TO READ SCREEN AGAIN.                *
*  8.PRINT THE FINISH MESSAGE.                              *
*                                                           *
*************************************************************
```

```
BEGIN
******** DEFINE THE SCREEN ********
SCREEN INPUT
TITLE 'SEARCH THE EXISTING FILE TO FIND THE UNUSED'
PROMPT 'QUESTION ID NUMBER IN THE DESIRED RANGE.'
SKIP 1 LINE
PROMPT 'ENTER START VALUE(FROM 1 TO 9999):' INPUT START -
                      LEN 4 NUMERIC RANGE 1 TO 9999
SKIP 1 LINE
PROMPT 'ENTER LAST  VALUE(FROM 1 TO 9999):' INPUT LAST  -
                      LEN 4 NUMERIC RANGE 1 TO 9999
SKIP 2 LINE
PROMPT 'DO YOU WANT TO LOOK AT ANOTHER RANGE OF VALUES?'
PROMPT 'ENTER (YES:Y, NO=<CR>):' INPUT YES LEN 1
END SCREEN
```

```
******** END SCREEN DEFINE *********
%START IS FIXED
%LAST  IS FIXED
%VALUE IS FIXED
%REVA  IS FIXED
%V1    IS FIXED
******** END VARIABLES DEFINE ********
1. READ SCREEN INPUT
2. %START= %INPUT:START
   %LAST = %INPUT:LAST
   %FLAG = 'N'
3. IF  %START GT %LAST THEN
       PRINT 'ERROR: FIRST VALUE IS LARGER' AND -
                          'THAN THE LAST VALUE.'
       JUMP TO  6
4. FOR %VALUE FROM %START TO %LAST
  4.1 %NO   = ''
      %REVA = %VALUE
  4.2 %V1   = $MOD(%REVA,10)
  4.3 IF %V1 EQ 0 THEN %V1 = 10
  4.4 %NO=$SUBSTR('1234567890',%V1,1) WITH %NO
  4.5 %REVA = %REVA/10
  4.6 IF %REVA GE 1 THEN JUMP TO 4.2
  4.7 FIND ALL RECORDS FOR WHICH NO = %NO
  4.8 COUNT RECORDS IN 4.7
  4.9 %COUNT1 = COUNT IN 4.8
  4.10 IF %COUNT1 EQ 0 THEN
    4.10.1 PRINT %NO TO COLUMN 10
    4.10.2 %FLAG = 'Y'
5. IF %FLAG EQ 'N' THEN
      PRINT 'THERE IS NO UNUSED QUESTION ID NUMBER' AND -
        'IN THIS RANGE.' AND %START AND 'TO' AND %LAST
6. IF %INPUT:YES EQ 'Y' THEN JUMP TO 1
7. PRINT '***** FINISH THE MISSID PROCEDURE'
END
```

APPENDIX J

MODIFY PROCEDURE

```
**************************************************************
*                                                            *
*  MODIFY: MODIFY THE QUESTION IN THE BANK.                  *
*                                                            *
*= P = = D = = L = = = = D = = E = = S = = I = = G = = N = =  *
*   DEFINE SCREEN.                                           *
*   DEFINE VARIABLES.                                        *
*   SET THE PRINT FLAG TO POSITIVE.                          *
*    0.READ SCREEN TO ENTER THE QUESTION ID NUMBER.          *
*      CLEAR LIST PRINTLIST.                                 *
*    1.QUESTION ID NUMBER SHOULD NOT BE THE NULL STRING.     *
*    2.--5.CHECK THAT WANT TO DISPLAY THE QUESTION OR NOT.   *
*    6.READ THE INPUT FIELD NAME SCREEN.                     *
*    7.ACCORDING TO THE INPUT FIELD NAME SET FLAG AND        *
*      JUMP TO THE RIGHT STATEMENT.                          *
*    8.READ QUESTION TYPE SCREEN.                            *
*    9.CHANGE THE QUESTION TYPE.                             *
*   10.READ QUESTION TOPIC SCREEN.                           *
*   11.CHANGE THE QUESTION TOPIC.                            *
*   12.READ QUESTION LEVEL SCREEN.                           *
*   13.CHANGE THE QUESTION LEVEL.                            *
*   14.READ QUESTION TIME SCREEN.                            *
*   15.CHANGE THE QUESTION TIME.                             *
*   16.READ QUESTION ANSWER SCREEN.                          *
*   17.CHANGE THE QUESTION ANSWER.                           *
*   18.READ QUESTION FLAG SCREEN.                            *
*   19.CHANGE THE QUESTION FLAG.                             *
*   20.READ QUESTION SUB SCREEN.                             *
*   21.CHANGE THE QUESTION SUBSTITUTE VARIABLES.             *
*   22.READ QUESTION NO SCREEN.                              *
*   23.CHANGE THE QUESTION ID NUMBER.                        *
*   24.PRINT THE SEQUENCE AND QUESTION TEXT.                 *
*   25.CHECK THAT QUESTION TEXT RECORD IS THE ONE WHICH      *
*      YOU WANT TO CHANGE.                                   *
*   26.CHECK, AFTER MODIFY THE QUESTION, DO YOU WANT         *
*      TO DISPLAY IT OR NOT.                                 *
*   27.IF TRUE, THEN DISPLAY IT.                             *
*   28.IS THERE ANOTHER QUESTION TO MODIFY?                  *
*   29.IF TRUE, THEN JUMP TO FIRST STATEMENT.                *
*   30.PRINT THE FINISH MESSAGE.                             *
*                                                            *
**************************************************************
```

```
BEGIN
******** DEFINE THE SCREEN ********
SCREEN INO
TITLE  'PLEASE INPUT THE QUESTION ID NUMBER.'
SKIP 2 LINE
PROMPT 'ENTER QUESTION ID NUMBER(FROM 1 TO 9999):' -
              INPUT NO LEN 4 NUMERIC RANGE 1 TO 9999
SKIP 2 LINE
PROMPT 'DO YOU WANT TO DISPLAY THIS QUESTION?'
PROMPT 'ENTER (YES:Y, NO:<CR>):' INPUT YES LEN 1
END SCREEN
******** DEFINE THE SCREEN ********
SCREEN IFIELD
TITLE  'SELECT THE FIELD NAME FROM THE FOLLOWING LIST.'

PROMPT '1. TYPE (QUESTION TYPE)'

PROMPT '2. TOPIC (QUESTION TOPIC)'

PROMPT '3. LEVEL (QUESTION DIFFICULTY LEVEL)'

PROMPT '4. NO (QUESTION ID NUMBER)'

PROMPT '5. FLAG (QUESTION HAS SUBSTITUTE ARRAY OR NOT)'

PROMPT '6. TIME (QUESTION ESTIMATED TIME TO FNISH)'

PROMPT '7. CONTENT (QUESTION TEXT)'

PROMPT '8. SQ (QUESTION RECORD SEQUENCE NUMBER)'

PROMPT '9. ANS (QUESTION ANSWER)'

PROMPT '0. SUB (QUESTION SUBSTITUTE VARIABLES)'

PROMPT 'ENTER THE SELECTED FIELD NAME(FROM 0 TO 9):' -
           INPUT OPTION LEN 1 NUMERIC RANGE 0 TO 9
END SCREEN
******** DEFINE THE SCREEN ********
SCREEN ICONT
PROMPT '*** INPUT THE CONTENT'
PROMPT -
 '12345678901234567890123456789012345678901234567890' -
   AT COLUMN 9
PROMPT 'CON1:' INPUT CONT1 LEN 50
PROMPT 'CON2:' INPUT CONT2 LEN 50
PROMPT 'CON3:' INPUT CONT3 LEN 50
PROMPT 'CON4:' INPUT CONT4 LEN 50
PROMPT 'CON5:' INPUT CONT5 LEN 50
END SCREEN
******** DEFINE THE SCREEN ********
SCREEN ILFN
TITLE  'MODIFY THE QUESTION TYPE.'
```

```
SKIP 2 LINE
PROMPT '1. TRUE-FALSE'
PROMPT '2. MULTIPLE'
PROMPT '3. COMPLETION'
PROMPT '4. SYNTAX'
PROMPT '5. QUESTION'
PROMPT 'SELECT ONE TYPE OF QUESTION:' INPUT OPTION -
                    ONEOF 1,2,3,4,5
END SCREEN
******** DEFINE THE SCREEN ********
SCREEN ITOPIC
TITLE  'MODIFY THE QUESTION TOPIC.'
SKIP 2 LINE
PROMPT 'INPUT THE TOPIC VALUE:' INPUT TOPIC LEN  4
END SCREEN
******** DEFINE THE SCREEN ********
SCREEN ILEVEL
TITLE  'MODIFY THE QUESTION DIFFICULT LEVEL.'
SKIP 2 LINE
PROMPT 'INPUT THE LEVEL VALUE:' INPUT LEVEL LEN  1 -
                                  NUMERIC RANGE 1 TO 5
END SCREEN
******** DEFINE THE SCREEN ********
SCREEN IFLAG
TITLE  'MODIFY THE QUESTION SUBSTITUTE FLAG.'
SKIP 2 LINE
PROMPT 'INPUT THE FLAG VALUE:' INPUT FLAG LEN  1 -
                                  ONEOF 1,0
END SCREEN
******** DEFINE THE SCREEN ********
SCREEN ITIME
TITLE  'MODIFY THE QUESTION ESTIMATE FINISH TIME.'
SKIP 2 LINE
PROMPT 'INPUT THE TIME VALUE:' INPUT TIME LEN  2 -
                                 NUMERIC RANGE 1 TO 99
END SCREEN
******** DEFINE THE SCREEN ********
SCREEN IANS
TITLE  'MODIFY THE QUESTION ANSWER KEY.'
SKIP 2 LINE
PROMPT 'INPUT THE ANS VALUE:' INPUT ANS LEN 77
END SCREEN
******** DEFINE THE SCREEN ********
SCREEN ISUB
TITLE  'MODIFY THE QUESTION SUBSTITUTE VARIABLES.'
SKIP 2 LINE
PROMPT 'INPUT THE SUB VALUE:' INPUT SUB LEN 77
END SCREEN
******** DEFINE THE VARIABLES ********
%CONTENT IS LEN 250
%SUB     IS LEN 250
%ANS     IS LEN 250
%COUNT1  IS FIXED
%PFLAG='Y'
```

```
 0. READ SCREEN INO
    %NO = %INO:NO
    %PFLAG = 'N'
    CLEAR LIST PRINTLIST
 1. IF %INO:NO EQ '' THEN
  1.1 PRINT 'ERROR:PLEASE ENTER THE ID NUMBER FIRST'
  1.2 JUMP TO 28
 2. FIND ALL RECORDS FOR WHICH NO=%NO
 3. PLACE RECORDS IN 2 ON LIST PRINTLIST
 4. IF %INO:YES EQ 'Y' THEN
  4.1 %PFLAG = 'Y'
  4.2 CALL 99
  4.3 %PFLAG = 'N'
 5. IF %ERROR EQ 'Y' THEN JUMP TO 28
 6. READ SCREEN IFIELD
 7.   IF %IFIELD:OPTION EQ  '1'  THEN JUMP TO  8
  71. IF %IFIELD:OPTION EQ  '2'  THEN JUMP TO 10
  72. IF %IFIELD:OPTION EQ  '3'  THEN JUMP TO 12
  73. IF %IFIELD:OPTION EQ  '4'  THEN JUMP TO 22
  74. IF %IFIELD:OPTION EQ  '5'  THEN JUMP TO 18
  75. IF %IFIELD:OPTION EQ  '6'  THEN JUMP TO 14
  76. IF %IFIELD:OPTION EQ  '7'  THEN JUMP TO 24
  77. IF %IFIELD:OPTION EQ  '8'  THEN JUMP TO 24
  78. IF %IFIELD:OPTION EQ  '9'  THEN JUMP TO 16
  79. IF %IFIELD:OPTION EQ  '0'  THEN JUMP TO 20
  80. PRINT 'ERROR: SOMETHING WRONG'
  81. JUMP TO 23
 8. READ SCREEN ILFN
 9. FOR EACH RECORD ON LIST PRINTLIST
   9.1 IF SQ EQ '1' THEN
              CHANGE TYPE TO %ILFN:OPTION
   9.2 JUMP TO 26
10. READ SCREEN ITOPIC
11. FOR EACH RECORD ON LIST PRINTLIST
  11.1 IF SQ EQ '1' THEN CHANGE TOPIC TO %ITOPIC:TOPIC
  11.2 JUMP TO 26
12. READ SCREEN ILEVEL
13. FOR EACH RECORD ON LIST PRINTLIST
  13.1 IF SQ EQ '1' THEN CHANGE LEVEL TO %ILEVEL:LEVEL
  13.2 JUMP TO 26
14. READ SCREEN ITIME
15. FOR EACH RECORD ON LIST PRINTLIST
  15.1 IF SQ EQ '1' THEN CHANGE TIME TO %ITIME:TIME
  15.2 JUMP TO 26
16. READ SCREEN IANS
17. FOR EACH RECORD ON LIST PRINTLIST
  17.1 IF SQ EQ '1' THEN CHANGE ANS TO %IANS:ANS
  17.2 JUMP TO 26
18. READ SCREEN IFLAG
19. FOR EACH RECORD ON LIST PRINTLIST
  19.1 IF SQ EQ '1' THEN CHANGE FLAG TO %IFLAG:FLAG
  19.2 IF %IFLAG:FLAG EQ '0' THEN
         %SUB = $READ('PLEASE TYPE IN SUBSTITUTE ARRAY.')
  19.3 IF %IFLAG:FLAG EQ '1' THEN
```

```
                DELETE SUB
  19.4 JUMP TO 26
20. READ SCREEN ISUB
21. FOR EACH RECORD ON LIST PRINTLIST
  21.1 IF FLAG EQ '1' THEN
          PRINT '***ERROR: THE ORIGINAL QUESTION DOES '...
          PRINT 'NOT HAVE SUBSTITUTE VARIABLE ARRAY'
          JUMP TO 26
  21.2 IF SQ EQ '1' THEN CHANGE SUB TO %ISUB:SUB
  21.3 JUMP TO 26
22. READ SCREEN INO
23. FOR EACH RECORD ON LIST PRINTLIST
  23.1 CHANGE NO TO %INO:NO
  23.2 JUMP TO 26
24. PRINT 'SQ' AT COLUMN 2 AND 'CONTENT' AT COLUMN 5
25. FOR EACH RECORD ON LIST PRINTLIST
  25.1 PRINT SQ AT COLUMN 2 AND -
                  $SUBSTR(CONTENT,1,70) AT COLUMN 5
       IF $LEN($SUBSTR(CONTENT,71,70)) THEN
       PRINT $SUBSTR(CONTENT,71,70) AT COLUMN 5
  25.2 IF $LEN($SUBSTR(CONTENT,141,70)) THEN
       PRINT $SUBSTR(CONTENT,141,70) AT COLUMN 5
  25.3 IF $LEN($SUBSTR(CONTENT,211)) THEN
       PRINT $SUBSTR(CONTENT,211) AT COLUMN 5
  25.4 %X = $READ('DO YOU WANT TO CHANGE THIS' WITH -
                       ' RECORD SEQUENCE?')
  25.5 IF %X EQ 'Y' AND %IFIELD:OPTION EQ '8' THEN
    25.5.1 %SQ = $READ('PLEASE TYPE IN SEQUENCE NUMBER.')
    25.5.2 CHANGE SQ TO %SQ
  25.6 IF %X EQ 'Y' AND %IFIELD:OPTION EQ '7' THEN
    25.6.1 READ SCREEN ICONT
    25.6.2 %CONTENT=%ICONT:CONT1 WITH %ICONT:CONT2 WITH -
                    %ICONT:CONT3 WITH %ICONT:CONT4 WITH -
                    %ICONT:CONT5
    25.6.3 CHANGE CONTENT TO %CONTENT
26. %X = $READ('DO YOU WANT TO DISPLAY THIS QUESTION?')
27. IF %X EQ 'Y' THEN
  27.1 %PFLAG = 'Y'
  27.2 CALL 99
  27.3 %PFLAG = 'N'
28. %X = $READ('IS THERE ANOTHER QUESTION TO MODIFY?')
29. IF %X EQ 'Y' THEN JUMP TO 0
30. PRINT '***** FINISH THE MODIFY PROCEDURE'
99. SUBROUTINE
 INCLUDE PRINT1
END
END PROCEDURE
```

APPENDIX K

PRINT PROCEDURE

```
*************************************************************
*                                                           *
*  PRINT: PRINT THE DESIRED QUESTION ID NUMBER RECORD.      *
*                                                           *
*= P = = D = = L = = = = D = = E = = S = = I = = G = = N = *
*    SET THE PRINT FLAG TO POSITIVE.                        *
*    1. ENTER THE QUESTION ID NUMBER.                       *
*    2. FIND ALL RECORDS WITH DESIRED ID NUMBER.            *
*    3. PLACE THE FIND RECORDS ON LIST PRINTLIST.           *
*    4. INCLUDE THE PRINT1 TO PRINT THE QUESTION AND CHECK.*
*    5. CLEAR LIST PRINTLIST.                               *
*    6. IS THERE ANOTHER QUESTION TO DISPLAY?               *
*    7. IF TRUE, JUMP TO 1.                                 *
*    8. PRINT THE FINISH MESSAGE.                           *
*                                                           *
*************************************************************
```

```
BEGIN
%PFLAG = 'Y'
1. %NO = $READ('PLEASE TYPE IN THE QUESTION ID NUMBER?')
2. FIND ALL RECORDS FOR WHICH NO = %NO
3. PLACE  RECORDS IN 2 ON LIST PRINTLIST
4. CALL 99
5. CLEAR LIST PRINTLIST
6. %X = $READ('IS THERE ANOTHER QUESTION TO DISPLAY?')
7. IF %X EQ 'Y' THEN JUMP TO 1
8. PRINT '***** FINISH THE PRINT PROCEDURE'
*************************************************************
*  SUBROUTINE TO PRINT THE DESIRED QUESTION.                *
*************************************************************
99. SUBROUTINE
   INCLUDE PRINT1
END
```

APPENDIX L

SEARCH PROCEDURE

```
*************************************************************
*                                                           *
*  SEARCH: SEARCH THE QUESTION BANK TO FIND THE  DESIRED    *
*          FEATURES OF THE QUESTIONS.  USER CAN RELAX SOME  *
*          FEATURES BY NOT PROVIDING IT.  THE FOLLOWING     *
*          LISTS ARE THE SPECIFIED FEATURES.                *
*                                                           *
*                                                           *
*   QUESTION TYPE    : ONE OF TRUE/FALSE, MULTIPLE, SYNTAX, *
*                      COMPLETION, QUESTION.   TO FIND ALL  *
*                      QUESTIONS BELONG TO SPECIFIED TYPE.  *
*                                                           *
*   QUESTION CHAPTER: PROVIDE TWO CHARACTERS TO INDICATE    *
*                      THE CHAPTER NUMBER.  TO FIND ALL     *
*                      QUESTIONS BELONGING TO SPECIFEID     *
*                      CHAPTER.                             *
*                                                           *
*   QUESTION SECTION: PROVIDE TWO CHARACTERS TO INDICATE    *
*                      THE SECTION NUMBER.  TO FIND ALL     *
*                      QUESTIONS BELONGING TO SPECIFEID     *
*                      SECTION. (THIS FIELD MUST ENTER      *
*                      TOGETHER WITH CHAPTER.)              *
*                                                           *
*   QUESTION LEVEL   : PROVIDE  THE  QUESTION  DIFFICULTY   *
*                      LEVEL  TO  FIND  ALL  QUESTIONS      *
*                      BELONGING TO THIS LEVEL.             *
*                                                           *
*                                                           *
*                                                           *
*= P = = D = = L = = = = D = = E = = S = = I = = G = = N = *
*  DEFINE SCREEN.                                           *
*  DEFINE VARIABLES.                                        *
*  READ SEARCH SCREEN TO INPUT THE QUESTION FEATURES.       *
*  1.--5. ACCORDING TO THE INPUT FEATURES SET THE FLAG.     *
*  6.IF THERE IS TOPIC FLAG THEN                            *
*      FIND ALL QUESTIONS WHICH HAVE THE DESIRED TOPIC AND  *
*            PLACE IT ON THE LIST SEELIST.                  *
*      JUMP TO 12.                                          *
*  7.IF THERE IS CHAPTER FLAG THEN                          *
*      FIND ALL QUESTIONS WHICH HAVE THE DESIRED CHAPTER    *
*           AND PLACE THEM ON THE LIST SEELIST.             *
*      JUMP TO 12.                                          *
*  8.IF THERE IS QUESTION TYPE FLAG AND LEVEL FLAG THEN     *
```

```
*      FIND ALL QUESTIONS WITH THE DESIRED TYPE AND LEVEL, *
*           AND PLACE THEM ON THE LIST SEELIST.            *
*      JUMP TO 15.                                         *
*  9.IF THERE IS QUESTION LEVEL FLAG THEN                  *
*      FIND ALL QUESTIONS WITH THE DESIRED LEVEL,          *
*           AND PLACE THEM ON THE LIST SEELIST.            *
*      JUMP TO 15.                                         *
*  10.IF THERE IS QUESTION TYPE FLAG THEN                  *
*      FIND ALL QUESTIONS WITH THE DESIRED TYPE ,          *
*           AND PLACE THEM ON THE LIST SEELIST.            *
*      JUMP TO 15.                                         *
*  11.FIND ALL QUESTION.                                   *
*      PLACE ALL QUESTION ON THE LIST SEELIST.             *
*  12.IF THERE IS QUESTION TYPE FLAG THEN                  *
*      FIND ALL QUESTIONS ON LIST SEELIST WHICH DO NOT     *
*           HAVE DESIRED QUESTION TYPE.                    *
*      REMOVE ALL QUESTIONS FOUND IN ABOVE STATEMENT FROM  *
*           LIST SEELIST.                                  *
*  13.IF THERE IS QUESTION LEVEL FLAG THEN                 *
*      FIND ALL QUESTIONS ON LIST SEELIST WHICH DO NOT     *
*           HAVE DESIRED QUESTION LEVEL.                   *
*      REMOVE ALL QUESTIONS FOUND IN ABOVE STATEMENT FROM  *
*           LIST SEELIST.                                  *
*  15.--17.COUNT THE QUESTIONS ON THE LIST SEELIST WHICH   *
*           SATISFY THE DESIRED FEATURE AND PRINT THEM.    *
*  18.--19.PRINT THE MESSAGE, TO CHECK DISPLAY ENTIRE      *
*           QUESTION OR NOT.                               *
*  20.--21.PRINT THE MESSAGE, TO CHECK DISPLAY RANDOM      *
*           QUESTION OR NOT.                               *
*  22.CHECK THAT WHETHER OR NOT TO ASSIGN THE FOUND        *
*     QUESTIONS TO TEST.                                   *
*  23.READ THE SELECT SCREEN.                              *
*     CHECK THAT THE INPUT DATA IS VALID OR NOT.           *
*     IF VALID THEN ASSIGN THE QUESTION TO TEST.           *
*     OTHERWISE REREAD SELECT SCREEN.                      *
*  25.PRINT THE FINISH MESSAGE.                            *
*                                                          *
************************************************************
```

```
BEGIN
******** DEFINE THE SCREEN *******
SCREEN SEARCH
TITLE 'SEARCH FOR SPECIFIED FEATURES OF QUESTIONS.'
PROMPT 'YOU CAN RELAX ANY FIELD VALUE ' -
PROMPT 'BY NOT PROVIDING IT.'
SKIP 2 LINE
PROMPT '1. TRUE-FALSE'
PROMPT '2. MULTIPLE CHOICE'
PROMPT '3. COMPLETION'
PROMPT '4. SYNTAX ERROR'
PROMPT '5. ESSAY QUESTION'
PROMPT 'ENTER TYPE OF QUESTION' -
PROMPT  '(FROM 1 TO 5):' INPUT LFN LEN 1 -
```

```
                    NUMERIC RANGE 1 TO 5
SKIP 2 LINE
PROMPT 'SELECT CHAPTER AND SECTION, ' -
PROMPT 'OR JUST THE CHAPTER FIELD'
PROMPT 'ENTER CHAPTER(FROM 1 TO 99):' INPUT CHAPTER -
                          LEN 2 NUMERIC RANGE 1 TO 99
PROMPT 'ENTER SECTION(FROM 1 TO 99):' INPUT SECTION -
                          LEN 2 NUMERIC RANGE 1 TO 99
SKIP 2 LINE
PROMPT 'ENTER DIFFICULTY LEVEL(FROM 1 TO 5):' -
          INPUT LEVEL LEN 1 NUMERIC RANGE 1 TO 5
END SCREEN
******** DEFINE THE SCREEN ********
SCREEN KIND
TITLE 'PLEASE ENTER TEST OR QUIZ, AND THE TIMES'
SKIP 1 LINE
PROMPT '1. TEST'
PROMPT '2. QUIZ'
PROMPT 'ENTER THE CHOICE(EITHER 1 OR 2):'  -
               INPUT OPTION LEN 1 ONEOF 1,2
SKIP 2 LINES
PROMPT 'ENTER THE NUMBER OF TEST(FROM 1 TO 30):' -
             INPUT TIME  LEN 2 NUMERIC RANGE 1 TO 30
END SCREEN
******** DEFINE THE SCREEN ********
SCREEN STEST
TITLE 'PLEASE ENTER QUESTION ID NUMBER AND ORDER'
PROMPT 'SEQUENCE NUMBER IN THE TEST'
SKIP 1 LINE
PROMPT 'INPUT THE RANDOM NUMBER WHICH IS LESS THAN' -
PROMPT 'THE FOUND NUMBER.'
PROMPT 'ENTER:' INPUT RNO LEN 4 NUMERIC RANGE 1 TO 9999
SKIP 1 LINE
PROMPT 'ENTER THE SEQUENCE ORDER IN THE TEST'
PROMPT 'ENTER(FROM 1 TO 50):' INPUT SQ LEN 2 -
                          NUMERIC RANGE 1 TO 50
SKIP 2 LINE
PROMPT 'DO YOU WANT TO DISPLAY THIS QUESTION?'
PROMPT 'ENTER(YES:Y, NO:<CR>):' INPUT YES LEN 1
SKIP 2 LINE
PROMPT 'IS THERE ANOTHER QUESTION TO SELECT?'
PROMPT 'ENTER(YES:Y, NO:<CR>):' INPUT YES1 LEN 1
END SCREEN
******** END DEFINE SCREEN ********
%COUNT1 IS FIXED
%K1     IS FIXED
%INDEX  IS FIXED
%PFLAG='Y'
READ SCREEN SEARCH
1. IF %SEARCH:LFN     NE '' THEN %FLAGL = 'Y'
2. IF $LEN(%SEARCH:SECTION) EQ 1 THEN
 2.1 %TOPIC=%SEARCH:CHAPTER WITH ' ' WITH %SEARCH:SECTION
 2.2 %FLAGT = 'Y'
3. IF $LEN(%SEARCH:SECTION) EQ 2 THEN
```

```
   3.1 %TOPIC = %SEARCH:CHAPTER WITH %SEARCH:SECTION
   3.2 %FLAGT = 'Y'
 4. IF $LEN(%SEARCH:SECTION) EQ 0 THEN
   4.1 IF %SEARCH:CHAPTER  NE '' THEN  %FLAGC = 'Y'
 5. IF %SEARCH:LEVEL   NE '' THEN %FLAGV = 'Y'
 6. IF (%FLAGT EQ 'Y') THEN
   6.1 FIND ALL RECORDS FOR WHICH TOPIC = %TOPIC
   6.2 PLACE RECORDS IN 6.1 ON LIST SEELIST
   6.3 JUMP TO 12
 7. IF (%FLAGC EQ 'Y') THEN
  7.1 FIND ALL RECORDS FOR WHICH CHAPTER = %SEARCH:CHAPTER
  7.2 PLACE RECORDS IN 7.1 ON LIST SEELIST
  7.3 JUMP TO 12
 8.IF ( %FLAGL EQ 'Y' ) AND ( %FLAGV EQ 'Y' ) THEN
   8.1 FIND ALL RECORDS FOR WHICH
          LEVEL = %SEARCH:LEVEL AND TYPE = %SEARCH:LFN
   8.2 PLACE RECORDS IN 8.1 ON LIST SEELIST
   8.3 JUMP TO 15
 9.IF ( %FLAGL EQ 'Y' ) AND ( %FLAGV NE 'Y' ) THEN
   9.1 FIND ALL RECORDS FOR WHICH
           TYPE = %SEARCH:LFN
   9.2 PLACE RECORDS IN 9.1 ON LIST SEELIST
   9.3 JUMP TO 15
10.IF ( %FLAGL NE 'Y' ) AND ( %FLAGV EQ 'Y' ) THEN
  10.1 FIND ALL RECORDS FOR WHICH
           LEVEL = %SEARCH:LEVEL
  10.2 PLACE RECORDS IN 10.1 ON LIST SEELIST
  10.3 JUMP TO 15
11.IF ( %FLAGL NE 'Y' ) AND ( %FLAGV NE 'Y' ) THEN
  11.1 FIND ALL RECORDS TOPIC IS PRESENT
  11.2 PLACE RECORDS IN 11.1 ON LIST SEELIST
  11.3 JUMP TO 15
12.IF  %FLAGL EQ 'Y'  THEN
  12.1 FIND ALL RECORDS ON LIST SEELIST FOR WHICH
           TYPE = NOT %SEARCH:LFN
  12.2 REMOVE RECORDS IN 12.1 FROM LIST SEELIST
13.IF  %FLAGV EQ 'Y'  THEN
  13.1 FIND ALL RECORDS ON LIST SEELIST FOR WHICH
           LEVEL = NOT %SEARCH:LEVEL
  13.2 REMOVE RECORDS IN 13.1 FROM LIST SEELIST
15. COUNT RECORDS ON LIST SEELIST
16. %COUNT1 = COUNT IN 15
17. PRINT %COUNT1 AND 'QUESTIONS SATISFY THE CONDITIONS.'
18. %X = $READ('DO YOU WANT TO DISPLAY ALL' WITH -
            ' QUESTIONS?')
19. IF %X EQ 'Y' THEN
  19.1 FOR EACH RECORD ON LIST SEELIST
    19.1.1  %NO = NO
    19.1.2  FIND ALL RECORDS FOR WHICH NO = %NO
    19.1.3  PLACE  RECORDS IN 19.1.2 ON LIST PRINTLIST
    19.1.4  CALL 99
    19.1.5  CLEAR LIST PRINTLIST
  19.2  JUMP TO 22
20. %X = $READ('DO YOU WANT TO DISPLAY A SELECTED' WITH -
```

```
              ' QUESTION AT RANDOM?')
21. IF %X EQ 'Y' THEN
  21.1 %K1 = $READ('PLEASE TYPE IN THE RANDOM NUMBER.')
  21.2 IF %K1 GT %COUNT1 THEN
    21.2.1 PRINT 'ERROR: INPUT NUMBER TOO LARGE'
    21.2.2 JUMP TO 21.1
  21.3 FOR EACH RECORD ON LIST SEELIST
    21.3.1 %INDEX = %INDEX+1
    21.3.2 IF %INDEX EQ %K1 THEN
      21.3.2.1 %NO = NO
      21.3.2.2 FIND ALL RECORDS FOR WHICH NO = %NO
      21.3.2.3 PLACE RECORDS IN 21.3.2.2 ON LIST PRINTLIST
      21.3.2.4 CALL 99
      21.3.2.5 CLEAR LIST PRINTLIST
  21.4. %X = $READ-
            ('DO YOU WANT TO DISPLAY ANOTHER QUESTION?')
  21.5. IF %X EQ 'Y' THEN
          %INDEX = 0
          JUMP TO 21.1
22. %X = $READ('DO YOU WANT TO ASSIGN QUESTION TO TEST?')
23. IF %X EQ 'Y' THEN
 23.1 READ SCREEN KIND
 23.2 IF %KIND:OPTION EQ '' THEN
   23.2.1   TAG %KIND:OPTION
   23.2.2   REREAD SCREEN KIND
 23.3 IF %KIND:TIME EQ '' THEN
   23.3.1   TAG %KIND:TIME
   23.3.2   REREAD SCREEN KIND
 23.4 READ SCREEN STEST
 23.5 IF %STEST:SQ EQ '' THEN
   23.5.1  TAG %STEST:SQ
   23.5.2  REREAD SCREEN STEST
 23.6 IF %STEST:RNO EQ '' THEN
   23.6.1  TAG %STEST:RNO
   23.6.2  REREAD SCREEN STEST
 23.7 IF %STEST:RNO GT %COUNT1 THEN
   23.7.1  PRINT 'THE RANDOM NUMBER IN THE SCREEN' AND -
         'SHOULD BE LESS THAN THE FOUND NUMBER. ' %COUNT1
   23.7.2  TAG %STEST:RNO
   23.7.3  REREAD SCREEN STEST
 23.11 FIND ALL RECORDS FOR WHICH TEST = %KIND:OPTION -
            AND TESTNO = %KIND:TIME AND TSQ = %STEST:SQ
 23.12 COUNT RECORDS IN 23.11
 23.13 %COUNTR = COUNT IN 23.12
 23.14 IF %COUNTR GE 1 THEN
   23.14.1 PRINT 'HAS SELECTED THE QUESTION IN' AND -
                    'THIS SEQUENCE' AND %STEST:SQ
   23.14.2 TAG %STEST:SQ
   23.14.3 REREAD SCREEN STEST
   23.14.4 JUMP TO 23.5
 23.8  %INDEX=0
 23.9  FOR EACH RECORD ON LIST SEELIST
  23.9.1 %INDEX = %INDEX + 1
  23.9.2 IF %INDEX EQ %STEST:RNO THEN
```

```
    23.9.2.1 %NO = NO
    23.9.2.2 FIND ALL RECORDS FOR WHICH QNO = %NO AND -
             TESTNO = %KIND:TIME AND TEST = %KIND:OPTION
    23.9.2.3 COUNT RECORDS IN 23.9.2.2
    23.9.2.4 %COUNTR = COUNT IN 23.9.2.3
    23.9.2.5 IF %COUNTR GE 1 THEN
      23.9.2.5.1 PRINT 'HAS SELECTED THIS' AND -
                       'QUESTION ' %NO ' IN THIS TEST.'
      23.9.2.5.2 TAG %STEST:RNO
      23.9.2.5.3 REREAD SCREEN STEST
      23.9.2.5.4 JUMP TO 23.5
    23.9.2.6 STORE RECORD
             TEST   = %KIND:OPTION
             TESTNO = %KIND:TIME
             TSQ    = %STEST:SQ
             QNO    = %NO
    23.9.2.7 IF %STEST:YES EQ 'Y' THEN
      23.9.2.7.1 FIND ALL RECORDS FOR WHICH NO = %NO
      23.9.2.7.2 PLACE RECORDS IN 23.9.2.7.1 -
                                  ON LIST PRINTLIST
      23.9.2.7.3 CALL 99
      23.9.2.7.4 REMOVE RECORDS IN 23.9.2.7.1 -
                                   FROM LIST SEELIST
      23.9.2.7.5 %COUNT1 = %COUNT1 - 1
      23.9.2.7.6 CLEAR LIST PRINTLIST
    23.9.2.8 IF %STEST:YES1 EQ 'Y' THEN
      23.9.2.8.1 JUMP TO 23.4
25. PRINT '***** FINISH THE SEARCH PROCEDURE'
*************************************************************
*   SUBROUTINE TO PRINT THE DESIRED QUESTION.               *
*************************************************************
99. SUBROUTINE
  INCLUDE PRINT1
  END
```

APPENDIX M

TEST PROCEDURE

```
*************************************************************
*                                                           *
*   TEST: SELECT THE SPECIFIED  QUESTION INTO THE TEST.     *
*                                                           *
*= P = = D = = L = = = = D = = E = = S = = I = = G = = N = *
*   DEFINE SCREEN.                                          *
*   DEFINE VARIABLES.                                       *
*    0. READ INPUT SCREEN.                                  *
*    1.-2. CHECK THAT THE INPUT STRING IS NOT NULL.         *
*       IF NULL, THEN TAG IT AND REREAD SCREEN.             *
*    3.-4. READ SCREEN TO SELECT DESIRED QUESTION.          *
*       CHECK THAT THE INPUT STRING IS NOT NULL.            *
*       IF NULL, THEN TAG IT AND REREAD SCREEN.             *
*    5. CLEAR THE PRINT LIST FIRST.                         *
*    6.--9. CHECK THAT THE INPUT QUESTION NUMBER EXISTS.    *
*   11. IF TRUE, THEN DISPLAY IT.                           *
*   12.--15. CHECK THAT THE SPECIFIED QUESTION HAS BEEN     *
*       SELECTED IN THIS TEST.                              *
*   16.--19. CHECK THAT THE SPECIFIED SEQUENCE HAS          *
*       SELECTED THE QUESTION IN IT.                        *
*   20. SELECT THE SPECIFIED QUESTION TO THE TEST.          *
*   21. IS THERE ANOTHER QUESTION TO SELECT?                *
*       IF TRUE, JUMP TO 4.                                 *
*   22. PRINT THE FINISH MESSAGE.                           *
*                                                           *
*************************************************************
```

```
BEGIN
******** DEFINE THE SCREEN ********
SCREEN KIND
TITLE 'PLEASE SELECT EITHER TEST OR QUIZ, '
PROMPT 'AND THE NUMBER OF TEST.'
SKIP 1 LINE
PROMPT '1. TEST'
PROMPT '2. QUIZ'
PROMPT 'ENTER THE CHOICE(TEST:1, QUIZ:2):'  -
                  INPUT OPTION LEN 1 ONEOF 1,2
SKIP 2 LINES
PROMPT 'ENTER THE NUMBER OF TEST(FROM 1 TO 30):' -
               INPUT TIME  LEN 2 NUMERIC RANGE 1 TO 30
END SCREEN
******** DEFINE THE SCREEN ********
```

```
SCREEN MAKE
TITLE 'PLEASE ENTER QUESTION ID NUMBER AND ORDER'
PROMPT 'SEQUENCE NUMBER IN THE TEST'
SKIP 1 LINE
PROMPT 'INPUT THE QUESTION ID NUMBER.'
PROMPT 'ENTER(FROM 1 TO 9999):' -
            INPUT NO LEN 4 NUMERIC RANGE 1 TO 9999
SKIP 1 LINE
PROMPT 'INPUT THE SEQUENCE ORDER IN THE TEST'
PROMPT 'ENTER(FROM 1 TO 50):' INPUT SEQ LEN 2 -
                            NUMERIC RANGE 1 TO 50
SKIP 2 LINE
PROMPT 'DO YOU WANT TO DISPLAY THIS QUESTION?'
PROMPT 'ENETER(YES:Y, NO:<CR>):' INPUT YES LEN 1
SKIP 2 LINE
PROMPT 'IS THERE ANOTHER QUESTION TO ENTER?'
PROMPT 'ENTER(YES:Y, NO:<CR>):' INPUT YES1 LEN 1
END SCREEN
******** END DEFINE SCREEN ********
%COUNTT IS FIXED
%PFLAG='Y'
 0. READ SCREEN KIND
 1. IF %KIND:OPTION EQ '' THEN
   1.1 TAG %KIND:OPTION
   1.2 REREAD SCREEN KIND
 2. IF %KIND:TIME EQ '' THEN
   2.1 TAG %KIND:TIME
   2.2 REREAD SCREEN KIND
 3. READ SCREEN MAKE
    IF %MAKE:NO EQ '' THEN
   3.1   TAG %MAKE:NO
   3.2   REREAD SCREEN MAKE
 4. IF %MAKE:SEQ EQ '' THEN
   4.1   TAG %MAKE:SEQ
   4.2   REREAD SCREEN MAKE
 5  CLEAR LIST PRINTLIST
 6. FIND ALL RECORDS FOR WHICH NO = %MAKE:NO
 7. COUNT RECORDS IN 6
 8. %COUNTT = COUNT IN 7
 9. IF %COUNTT EQ 0 THEN
   9.1 PRINT 'THERE IS NO QUESTION WITH THIS' AND -
          'QUESTION ID NUMBER' AND %MAKE:NO
   9.2 JUMP TO 3
11. IF %MAKE:YES EQ 'Y' THEN
  11.1 %NO = %MAKE:NO
  11.2 PLACE RECORDS IN 6 ON LIST PRINTLIST
  11.3 CALL 99
12. FIND ALL RECORDS FOR WHICH
        TEST = %KIND:OPTION AND QNO = %MAKE:NO AND -
        TESTNO = %KIND:TIME
13. COUNT RECORDS IN 12
14. %COUNTT = COUNT IN 13
15. IF %COUNTT GE 1 THEN
  15.1 PRINT 'ALREADY HAS SELECT THIS QUESTION' AND -
```

```
              %MAKE:NO  AND ' IN THE TEST'
  15.2 JUMP TO 21
 16. FIND ALL RECORDS FOR WHICH
         TEST = %KIND:OPTION AND TSQ = %MAKE:SEQ AND -
         TESTNO = %KIND:TIME
 17. COUNT RECORDS IN 16
 18. %COUNTT = COUNT IN 17
 19. IF %COUNTT GE 1 THEN
   19.1 PRINT 'THERE ALREADY HAS THE QUESTION EXIST' AND -
            'IN THIS SEQUENCE' AND %MAKE:SEQ
   19.2 %X = $READ('DO YOU REALLY WANT TO OVERRIDE IT?')
   19.3 IF %X NE 'Y' THEN JUMP TO 21
   19.4 DELETE ALL RECORDS IN 16
 20. STORE RECORD
     TEST   = %KIND:OPTION
     TESTNO = %KIND:TIME
     QNO    = %MAKE:NO
     TSQ    = %MAKE:SEQ
 21. IF %MAKE:YES1 EQ 'Y' THEN JUMP TO 3
 22. PRINT '***** FINISH THE TEST PROCEDURE'
*************************************************************
*   SUBROUTINE TO PRINT THE DESIRED QUESTION.               *
*************************************************************
 99. SUBROUTINE
   INCLUDE PRINT1
 END
```

APPENDIX N

TESTD PROCEDURE

```
*************************************************************
*                                                           *
*  TESTD: DELETE THE SPECIFIED QUESTION FROM THE TEST.      *
*                                                           *
*= P = = D = = L = = = = D = = E = = S = = I = = G = = N = *
*   1. READ INPUT SCREEN.                                   *
*   2.--3. ASSIGN THE KIND OF TEST.                         *
*   4. FIND ALL QUESTION IN THE TEST.                       *
*   5. COUNT THE QUESTION IN THE TEST.                      *
*   6. ASSIGN THE COUNT TO THE VARIABLE.                    *
*   7. CHECK THAT THE COUNT OF QUESTION IS ZERO OR NOT.     *
*      IF COUNT IS ZERO, DISPLAY THE ZERO COUNT MESSAGE.    *
*   8.--11. FIND RECORDS ON TESTLIST WITH GIVEN             *
*      SEQUENCE NUMBER AND CHECK THE COUNT.                 *
*  12. CHECK THAT WANT TO DISPLAY THIS QUESTION OR NOT.     *
*      IF TRUE, THEN DISPLAY IT.                            *
*  13. CHECK REALLY WANT TO DELETE IT OR NOT.               *
*  14. IF FALSE, THEN JUMP TO 16.                           *
*  15. DELETE THE QUESTION IN THE TEST.                     *
*  16. IS THERE ANOTHER QUESTION TO DELETE.                 *
*      IF TRUE, THEN JUMP TO 1.                             *
*  17. PRINT THE FINISH MESSAGE.                            *
*                                                           *
*************************************************************
```

```
BEGIN
******** DEFINE THE SCREEN ********
SCREEN SELECT
TITLE 'DELETE THE QUESTION FROM THE TEST.'
SKIP 1 LINE
PROMPT 'PLEASE SELECT EITHER TEST OR QUIZ, '
PROMPT 'AND THE NUMBER OF TEST.'
SKIP 1 LINE
PROMPT '1. TEST'
PROMPT '2. QUIZ'
PROMPT 'ENTER THE CHOICE(TEST:1, QUIZ:2 ):' INPUT -
               OPTION LEN 1 ONEOF 1,2
SKIP 1 LINE
PROMPT 'ENTER THE NUMBER OF TEST(FROM 1 TO 30):' -
           INPUT EQ LEN 2 NUMERIC RANGE 1 TO 30
SKIP 2 LINE
PROMPT 'PLEASE TYPE IN THE SEQUENCE NUMBER' -
```

```
 PROMPT ' OF QUESTION YOU WISH TO DELETE.'
 PROMPT 'ENTER NUMBER(FROM 1 TO 50):' INPUT NO -
                        LEN 2 NUMERIC RANGE 1 TO 50
 SKIP 2 LINE
 PROMPT 'DO YOU WANT TO DISPLAY THIS QUESTION FIRST?'
 PROMPT 'ENTER(YES:Y, NO:<CR>):' INPUT DIS LEN 1
 SKIP 2 LINE
 PROMPT 'IS THERE ANOTHER QUESTION TO DELETE?'
 PROMPT 'ENTER(YES:Y, NO:<CR>):' INPUT YES LEN 1
 END SCREEN
 ******** END DEFINE SCREEN ********
 %COUNTT IS FIXED
 %PFLAG = 'Y'
 1. READ SCREEN SELECT
 2. IF %SELECT:OPTION EQ '1' THEN %LFN = 'TEST'
 3. IF %SELECT:OPTION EQ '2' THEN %LFN = 'QUIZ'
 4. FIND ALL RECORDS FOR WHICH -
       TEST = %SELECT:OPTION AND TESTNO = %SELECT:SEQ
 5. COUNT RECORDS IN 4
 6. %COUNTT = COUNT IN 5
 7. IF %COUNTT EQ 0 THEN
  7.1 PRINT 'THERE IS NO QUESTION IN THIS' AND -
            %LFN AND %SELECT:SEQ
  7.2 JUMP TO 16
 8. FIND ALL RECORDS IN 4 -
                FOR WHICH TSQ = %SELECT:NO
 9. COUNT RECORDS IN 8
10. %COUNTT = COUNT IN 9
11. IF %COUNTT EQ 0 THEN
   11.1 PRINT 'THERE IS NO QUESTION WITH SEQUENCE' AND -
        %SELECT:NO AND 'IN THIS' AND %LFN AND %SELECT:SEQ
   11.2 JUMP TO 16
12. IF %SELECT:DIS EQ 'Y' THEN
  12.1 FOR EACH RECORD IN 8
   12.1.1 NOTE QNO
   12.1.2 FIND ALL RECORDS FOR WHICH NO = VALUE IN 12.1.1
   12.1.3 PLACE RECORDS IN 12.1.2 ON LIST PRINTLIST
   12.1.4 CALL 99
   12.1.5 CLEAR LIST PRINTLIST
13. %X = $READ('DO YOU WANT TO DELETE THIS ' WITH -
                                'QUESTION FROM TEST?')
14. IF %X NE 'Y' THEN JUMP TO 16
15. FOR EACH RECORD IN 8
  15.1 DELETE RECORD
16. IF %SELECT:YES EQ 'Y' THEN JUMP TO 1
17. PRINT '***** FINISH THE TEST DELETE PROCEDURE'
*************************************************************
*   SUBROUTINE TO PRINT DESIRED QUESTION.                   *
*************************************************************
99. SUBROUTINE
  INCLUDE PRINT1
END
```

APPENDIX O

TESTP PROCEDURE

```
*************************************************************
*                                                           *
*  TESTP: DISPLAY THE QUESTION IN THE TEST.                 *
*                                                           *
*= P = = D = = L = = = = D = = E = = S = = I = = G = = N = *
*   1. READ INPUT SCREEN.                                   *
*   2.--3. ASSIGN THE KIND OF TEST.                         *
*   4. FIND ALL QUESTIONS IN THE TEST.                      *
*   5. COUNT THE QUESTIONS IN THE TEST.                     *
*   6. ASSIGN THE COUNT TO THE VARIABLE.                    *
*   7. CHECK THAT THE COUNT OF QUESTION IS ZERO OR NOT.     *
*   8. PRINT THE COUNT OF QUESTIONS.                        *
*   9. PLACE RECORDS ON TESTLIST.                           *
*  10. SORT THE QUESTION BY SEQUENCE NUMBER.                *
*  11. CHECK THAT WANT TO DISPLAY ALL QUESTION OR NOT.      *
*  12. IF TRUE, THEN DISPLAY ALL QUESTION.                  *
*  13. CHECK THAT WANT TO DISPLAY RANDOM NUMBER OF          *
*      QUESTION OR NOT.                                     *
*  14. IF TRUE, THEN DISPLAY THE DESIRED QUESTION.          *
*  15. IS THERE ANOTHER TEST TO DISPLAY.                    *
*  16. IF TRUE, THEN JUMP TO 1.                             *
*  17. PRINT THE FINISH MESSAGE.                            *
*                                                           *
*************************************************************
```

```
BEGIN
******** DEFINE THE SCREEN ********
SCREEN SELECT
TITLE 'PRINT OUT THE SPECFIED TEST.'
SKIP 1 LINE
PROMPT 'PLEASE SELECT EITHER TEST OR QUIA, ' -
PROMPT 'AND THE NUMBER OF TEST.'
SKIP 1 LINE
PROMPT '1. TEST'
PROMPT '2. QUIZ'
PROMPT 'ENTER THE CHOICE(TEST:1, QUIZ:2 ):'  -
                  INPUT OPTION LEN 1 ONEOF 1,2
SKIP 1 LINE
PROMPT 'ENTER THE NUMBER OF TEST(FROM 1 TO 30):' INPUT -
          SEQ LEN 2 NUMERIC RANGE 1 TO 30
END SCREEN
******** END DEFINE SCREEN ********
```

```
 %COUNTT IS FIXED
 %PFLAG = 'Y'
 1. READ SCREEN SELECT
 2. IF %SELECT:OPTION EQ '1' THEN %LFN = 'TEST'
 3. IF %SELECT:OPTION EQ '2' THEN %LFN = 'QUIZ'
 4. FIND ALL RECORDS FOR WHICH -
      TEST = %SELECT:OPTION AND TESTNO = %SELECT:SEQ
 5. COUNT RECORDS IN 4
 6. %COUNTT = COUNT IN 5
 7. IF %COUNTT EQ 0 THEN
  7.1 PRINT 'THERE IS NO QUESTION IN THIS' AND -
            %LFN AND %SELECT:SEQ
  7.2 JUMP TO 15
 8. PRINT 'FIND' AND %COUNTT AND 'IN THIS TEST.'
 9. PLACE RECORDS IN 4 ON LIST TESTLIST
10. SORT RECORDS ON LIST TESTLIST -
                          BY TSQ VALUE RIGHT-ADJUSTED
11. %X=$READ('DO YOU WANT DISPLAY ALL QUESTIONS?')
12. IF %X EQ 'Y' THEN
  12.1 FOR EACH RECORD IN 10
   12.1.1 NOTE QNO
   12.1.2 FIND ALL RECORDS FOR WHICH NO = VALUE IN 12.1.1
   12.1.3 COUNT RECORDS IN 12.1.2
   12.1.4 %COUNTT = COUNT IN 12.1.3
   12.1.5 IF %COUNTT EQ 0 THEN
      12.1.5.1 PRINT 'THIS QUESTION HAS BEEN DELETE' AND -
                          %NO AND 'SEQUENCE NUMBER' AND TSQ
      12.1.5.2 JUMP TO 12.1.10
   12.1.6 CLEAR LIST PRINTLIST
   12.1.7 PLACE RECORDS IN 12.1.2 ON LIST PRINTLIST
   12.1.8 PRINT 'SEQUENCE:' TSQ
   12.1.9 CALL 99
   12.1.10 *PROCESS NEXT RECORD
  12.2 JUMP TO 15
13. %X = $READ('DO YOU WANT DISPLAY RANDOM NUMBER ' WITH -
             'OF QUESTION?')
    %INDEX = 0
14. IF %X EQ 'Y' THEN
  14.1 %K1 = $READ('PLEASE TYPE IN THE RANDOM NUMBER?')
  14.2 FOR EACH RECORD IN 10
   14.2.1 %INDEX = %INDEX+1
   14.2.2 IF %INDEX NE %K1 THEN JUMP TO 14.2.12
   14.2.3 NOTE QNO
   14.2.4 FIND ALL RECORDS FOR WHICH NO = VALUE IN 14.2.3
   14.2.5 COUNT RECORDS IN 14.2.4
   14.2.6 %COUNTT = COUNT IN 14.2.5
   14.2.7 IF %COUNTT EQ 0 THEN
     14.2.7.1 PRINT 'THIS QUESTION HAS BEEN DELETED' AND -
                          %NO AND 'SEQUENCE NUMBER' AND TSQ
     14.2.7.2 JUMP TO 15
   14.2.8 CLEAR LIST PRINTLIST
   14.2.9 PLACE RECORDS IN 14.2.4 ON LIST PRINTLIST
   14.2.10 CALL 99
   14.2.11 JUMP TO 13
```

```
   14.2.12 *PROCESS NEXT RECORD
  14.3 %INDEX = 0
15. %X = $READ('IS THERE ANOTHER TEST TO DISPLAY?')
16. IF %X EQ 'Y' THEN JUMP TO 1
17. PRINT '***** FINISH THE TEST PRINT PROCEDURE'
************************************************************
*   SUBROUTINE TO PRINT DESIRED QUESTION.                  *
************************************************************
99. SUBROUTINE
  INCLUDE PRINT1
END
```

APPENDIX P

START PROCEDURE

```
*************************************************************
*                                                           *
*   START: TO INITIALIZE THE NECESSARY PARAMETER AND        *
*          DISPLAY THE CATC SYSTEM MESSAGE.                 *
*                                                           *
*    SET THE USER PARAMETERS.                               *
*    DISPLAY THE CATC SYSTEM MESSAGE.                       *
*                                                           *
*************************************************************
```

```
UTABLE LVTBL = 150
UTABLE LNTBL = 400
UTABLE LQTBL = 1000
UTABLE LSTBL = 6000
RESET  HDRCTL= 1
BEGIN
SET HEADER 1 'C   A   T   C       M  A  N  A  G  E  ' -
           AT 4 WITH 'M  E  N  T    S  Y  S  T  E  M'
NEW PAGE
1. SKIP 10 LINES
2. PRINT 'COMPUTER   ASSISTED   TEST   CONSTRUCTION  ' -
         AT 8 WITH 'SYSTEM    ( C A T C )'
3. SKIP 2 LINES
4. PRINT ' FOR  FORTRAN  COURSE'  AT 25
5. SKIP 3 LINES
6. PRINT ' C O M P U T I N G    &   ' AT 6 WITH -
         'I N F O R M A T I O N    S C I E N C E'
7. SKIP 2 LINES
8. PRINT ' O K L A H O M A    S T A T E    ' AT 10 WITH -
           ' U N I V E R S I T Y'
END
```

APPENDIX Q

PRINT1 PROCEDURE

```
*****************************************************************
*                                                               *
*   PRINT1: CALL BY THE OTHER PROCEDURE TO DISPLAY THE          *
*           FORMATED QUESTION.                                  *
*                                                               *
* = P = = D = = L = = = = D = = E = = S = = I = = G = = N =     *
*   DEFINE VARIABLES.                                           *
*   SET SCREEN HEADER.                                          *
*   99.1. SET THE ERROR FLAG TO NEGATIVE.                       *
*   99.2--99.4. IF THE COUNT ON LIST PRINTLIST IS 0 THEN        *
*        PRINT ERROR MESSAGE, SET ERROR FLAG AND RETURN.        *
*   99.5. IF PRINT FLAG IS NOT SET, THEN RETURN.                *
*   99.6. SORT THE RECORDS IN 99.1 BY SEQUENCE NUMBER.          *
*   99.7. PRINT THE TITLE.                                      *
*   99.8. PRINT THE QUESTION.                                   *
*   99.9. IF THERE ARE SUBSTITUTE VARIABLES THEN                *
*           PRINT THEM.                                         *
*   99.10 PRINT THE ANSWER.                                     *
*                                                               *
*****************************************************************
```

```
%TEMPCONT IS LEN 250
%SUBP     IS LEN 250
%ANSP     IS LEN 250
%ARRAY    IS STRING ARRAY(5)
%ARRAY(1) = 'TRUE-FALSE'
%ARRAY(2) = 'MULTIPLE'
%ARRAY(3) = 'COMPLETION'
%ARRAY(4) = 'SYNTAX'
%ARRAY(5) = 'QUESTION'
SET HEADER 1 'C  A  T  C      M  A  N  A  G  E  ' AT 5 -
     WITH 'M  E  N  T' WITH '      S  Y  S  T  E  M'
99.1. %ERROR = 'N'
99.2. COUNT RECORDS ON LIST PRINTLIST
99.3. %COUNT = COUNT IN 99.2
99.4. IF %COUNT EQ 0 THEN
  99.4.1 PRINT 'THERE IS NO SUCH QUESTION ID' AND %NO
  99.4.2 %ERROR='Y'
  99.4.3 JUMP TO 99.11
99.5. IF %PFLAG NE 'Y' THEN JUMP TO 99.12
99.6. SORT RECORDS ON LIST PRINTLIST -
                   BY SQ VALUE RIGHT-ADJUSTED
```

```
99.7. PRINT   'TYPE'     AT COLUMN  2 -
          AND 'TOPIC'    AT COLUMN 12 -
          AND 'LEVEL'    AT COLUMN 18 -
          AND 'TIME'     AT COLUMN 24 -
          AND 'FLAG'     AT COLUMN 29 -
          AND 'NO'       AT COLUMN 36
99.8. FOR EACH RECORD IN 99.6
 99.8.1 IF SQ NE '1' THEN JUMP TO 99.8.4
 99.8.2 PRINT %ARRAY(TYPE) AT COLUMN 2 -
            AND TOPIC      AT COLUMN 14 -
            AND LEVEL      AT COLUMN 20 -
            AND TIME       AT COLUMN 26 -
            AND FLAG       AT COLUMN 31 -
            AND NO         AT COLUMN 36 -
            AND 'CONTENT' AT COLUMN  5
           %ANSP = ANS
 99.8.3 IF FLAG EQ '0' THEN
          %FLAGF = 'Y'
          %SUBP  = SUB
 99.8.4 %TEMPCONT = CONTENT
 99.8.5 %K = $LEN(%TEMPCONT)
 99.8.6 IF %K LE 0 THEN JUMP TO 99.8.10
 99.8.7 PRINT %TEMPCONT AT COLUMN 5 TO COLUMN 54
 99.8.8 %TEMPCONT = $SUBSTR(%TEMPCONT,51)
 99.8.9 JUMP TO 99.8.5
 99.8.10 *PROCESS NEXT RECORD
99.9. IF %FLAGF EQ 'Y' THEN
      PRINT 'SUBSTITUTE VARIABLES ARE:' AND %SUBP
      %FLAGF = 'N'
99.10. PRINT 'ANS:' AND %ANSP
99.11. SKIP 1 LINE
99.12. *PROCESS NEXT RECORD
```

VITA

HUNE-CHI FUH

Candidate for the Degree of

Master of Science

Report: USING MODEL 204 DATABASE MANAGEMENT SYSTEM TO BUILD A COMPUTER ASSISTED TEST CONSTRUCTION SYSTEM FOR A FORTRAN COURSE.

Major Field: Computing and Information Science

Biographical:

Personal Data: Born in Hsinchu, Taiwan, Republic of China, October 28, 1952, the son of Mr. and Mrs. R. F. Fuh.

Education: Graduated from Hsinchu Provincial High School, Hsinchu, Taiwan, Republic of China, in June, 1971; received Bachelor of Science in Engineering Science degree from National Chung Kung university, Tainan, Taiwan, Republic of China, in June, 1976; completed requirements for the Master of Science degree at Oklahoma State University in May, 1983.

Professional Experience: Quality Control Engineer, Tai Chuyan Metal Company, LungTan, Taiwan, Republic of China, June, 1976 - September, 1976; Assistant Engineer,Taiwan Matsushita Electrical Company, October, 1976, - August, 1980; Programmer, Department of Agricultural Economics, Oklahoma State University, Stillwater, Oklahoma, January, 1981 - September, 1981; Programmer, Cooperative Extension Service, Oklahoma State University, Stillwater, Oklahoma, October, 1981 - December, 1981;

Name: Hune-Chi Fuh Date of Degree: May, 1983

Institution: Oklahoma State University

Location: Stillwater, Oklahoma

Title of Study: USING MODEL 204 DATABASE MANAGEMENT SYSTEM TO BUILD A COMPUTER ASSISTED TEST CONSTRUCTION SYSTEM FOR A FORTRAN COURSE.

Pages in Study: 92 Candidate for Degree of Master of Science

Major Field: Computing and Information Science

Scope and Method of Study: This paper describes the design and implementation of a computer assisted test construction (CATC) system for a FORTRAN course. The CATC system uses the Model 204 database management system. This CATC system is designed to be used by persons without knowledge in either computer programming or computer operations. It is composed of a set of computer programs which are invoked by one main procedure. These programs perform the functions of (1) updating, adding and deleting the questions in the bank; (2) searching the desired features of questions from the bank; (3) assigning the questions from the bank to the test; (4) deleting and printing the questions from the test.

Findings and Conclusions: This computer assisted test construction system permits the interactive construction of tests and the manipulation of the bank of test questions at a computer terminal. The system permits a test constructor to search through a large number of question very rapidly and without the clercial errors frequently involved in manual searches.

ADVISER'S APPROVAL_____

USING MODEL 204 DATABASE MANAGEMENT SYSTEM TO BUILD A COMPUTER ASSISTED TEST CONSTRUCTION SYSTEM FOR A FORTRAN COURSE

PRESENTATION

BY

HUNE-CHI FUH

February 24, 1983

CHAP I: Introduction.

A. Motivation of the CATC System.

B. Two Fundamentally Approaches To the CATC System.

1. CATC-R

2. CATC-G

C. General Aim of the CATC System.

D. Purpose of this Project.

CHAP II: Literature Review.

Three primary sources mainly describe methodological approaches and logistical concerns. There are 9 CATC systems which are described briefly in this chapter.

CHAP III: Design Philosophy and System Features.

A. Introduction.

B. Entering Question.

1. Introduction.

2. Field Description.

3. Record Format.

4. Data Compression.

5. Substitute Variables.

6. Input Data Format.

C. Organization of Question Bank.

1. Introduction.

Elementary In Record:

Field Name Code	Field value

Each Record Structure:

Record#	Field1 | Value	Field2 | Value	....

2. Field Attributes.

Functional Attributes.

Default	Nondefault
NONKEY VISIBLE NON-RANGE NON-FRV DEFERABLE	KEY INVISIBLE NUMERIC RANGE FRV NON-DEFERABLE LEVEL

Representational Attributes.

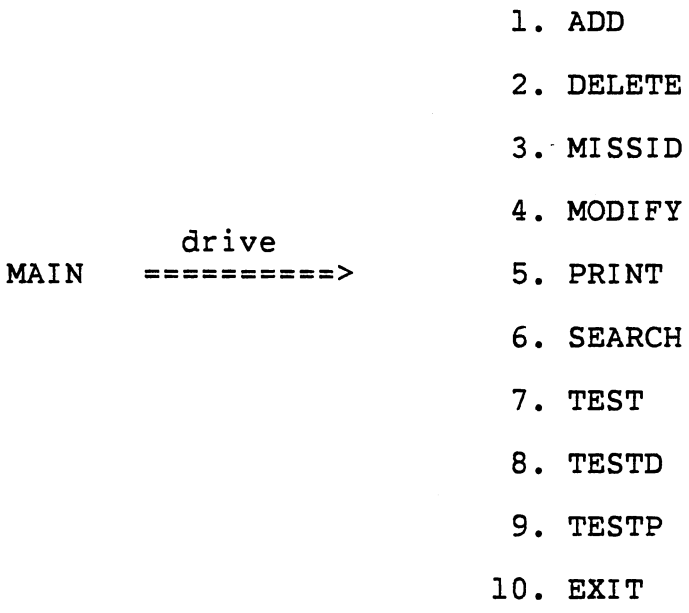
Default	Nondefault
NON-CODED STRING MANY-VALUED UPDATE IN PLACE	CODED BINARY FEW-VALUED UNDATE AT END OCCURS LENGTH PAD

3. File Organization.

D. Initialization of Question Bank.

E. Question Bank Maintenance.

F. Automatic Test Construction.



CHAP IV: Conclusions and Suggestions.

Conclusions:

A. Quality of This CATC System.

B. Classification of This CATC System.

C. Interactive Operation of This CATC System.

D. Verify of This CATC Sytem.

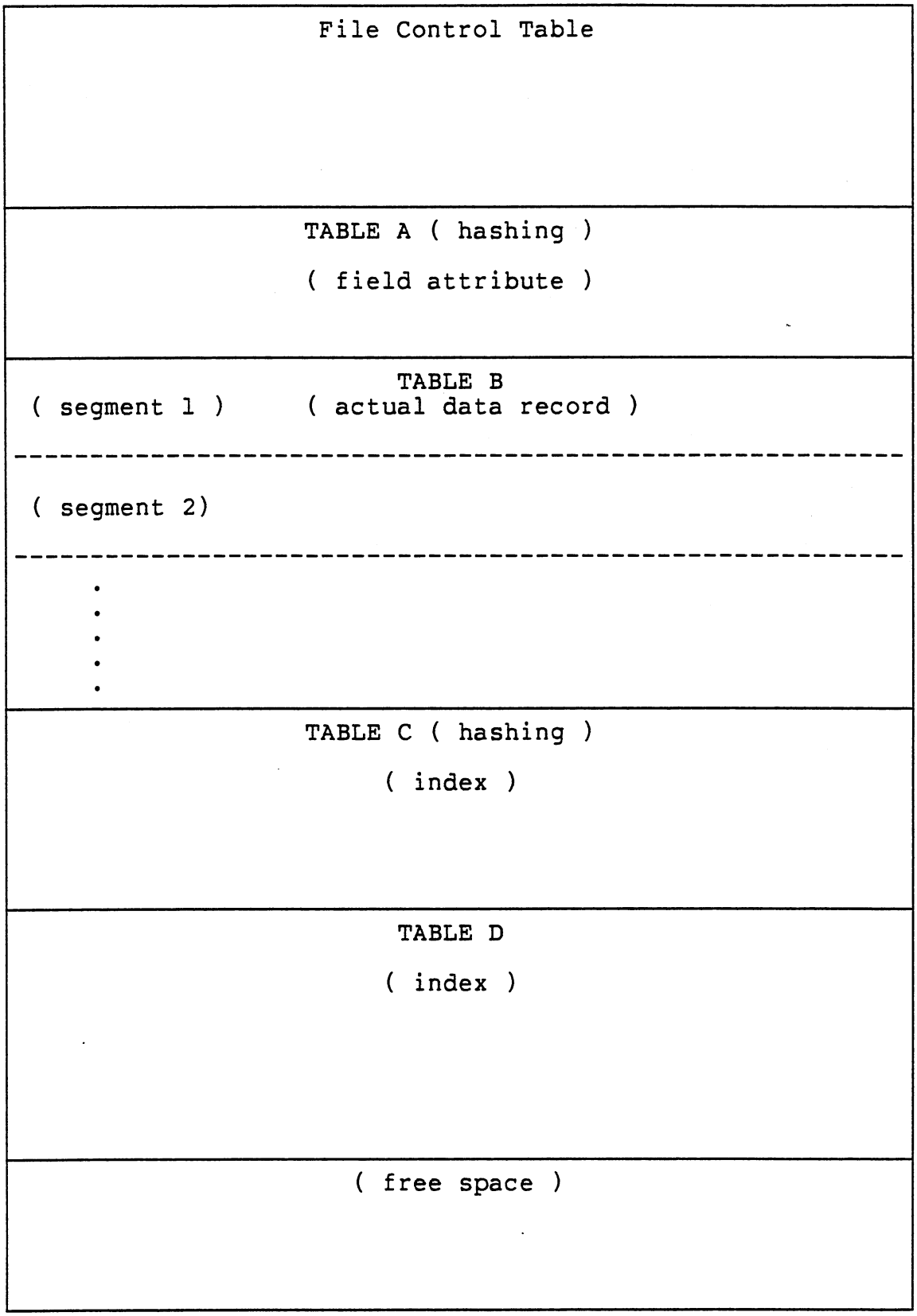
Suggestions:

A. Improve Current System.

B. Develope Automatic Scoring System.

C. Combine with CAI system.

Model 204 File Structure



File Control Table:

Fixed Size 8 Pages.

1. File Parameters.
2. DD Name in the File.
3. File Control Information.

TABLE A:

1. Field Names and Attributes Section.
2. FEW-VALUED Section.
3. MANY-VALUED Section.

TABLE B:

Store the actual data records, devided into internal file segments.

CODED :

Field Name Code	Coded Field Value
<---2 bytes---->	<------4 bytes------->

NON-CODED:

Field Name Code	Binary Value
<---2 bytes---->	<------4 bytes------->

Field Name Code	Length	Field Value
<---2 bytes---->	<1 byte>	<Max 255 bytes>

File Size Multipler: The number of internal file segments.

$$N = \frac{\text{\# of records in the file}}{8 * \text{page size}} \quad \text{(rounded to interger)}$$

TABLE C:

Make up the indexing structure.

Unique : | Field Name=Value | Pointer | (to TABLE B)

<---6 bytes----> <6 bytes>

# of N

Non-Unique : | Field Name=Value | Pointer | ..... | Pointer |

<---6 bytes----> <6 bytes>.....<6 bytes>

(to TABLE D)

TABLE D:

1. Preallocated Field Description.
2. Procedure Directory.
3. Procedures text.
4. Access Control Table.
5. Inverted File Lists.

Field value pairs occur fewer than 3%:

| Field Name=Value | Pointer | ..... | Pointer |

<---6 bytes----> <2 bytes> ..... <2 bytes>

Field value pairs occur more than 3%, allocate one page.

EXAMPLE

Major Question Example Feature

number	TYPE	TOPIC	LEVEL	NO	TIME
1	1(TRUE/FALSE)	1 1	1	1	3
2	1(TRUE/FALSE)	1 2	1	2	4
3	3(COMPLETION)	1 1	4	3	6

Physical Structure in TABLE A

Control Info.	TYPE	TOPIC	LEVEL	NO	TIME	. . . . (field name)
Control Info.	. . . . . (few-valued)					
Control Info.	. . . . . (many-valued)					

Physical Structure in TABLE B

Record1	TYPE|1|1	TOPIC|3|1 1	LEVEL|1|1	NO|1|1	TIME|1|3	...
Record2	TYPE|1|1	TOPIC|3|1 2	LEVEL|1|1	NO|1|2	TIME|1|4	...
Record3	TYPE|1|3	TOPIC|3|1 1	LEVEL|1|4	NO|1|3	TIME|1|6	...
.						

Physical Structure in TABLE C

NO=1	B	REC NO. 1
NO=2	B	REC NO. 2
NO=3	B	REC NO. 3
TYPE=TRUE/FALSE	D	PAGE 2
TYPE=COMPLETION	B	REC NO. 3
TOPIC=1 1	D	PAGE 3
TOPIC=1 2	B	REC NO. 2
LEVEL=1	D	PAGE 4
LEVEL=4	B	REC NO. 3
.		

Physical Structure in TABLE D

PAGE 0	EXISTENCE MAP
PAGE 1	PRELOCATED FIELD RECORD DESCRIPTION
PAGE 2	TYPE=TRUE/FALSE,1,2
PAGE 3	TOPIC=1 1,1,3
PAGE 4	LEVEL=1,1,2
PAGE .	.

```
83.016  JAN 16  15.50.06                                                                                                    PAGE 1

***         ******              MODEL 204 INITIALIZATION.  VERSION = 6.2                                                  **********
***         ******              SMF SYSTEM ID = A168                                                                      **********
***         ******              EXECUTE PARAMETERS: SYSOPT=192                                                            **********
***         ******              READING PARAMETERS
SPCORE=10000,PAGESZ=6184
SPCORE    10000         SPARE CORE RESERVED
PAGESZ    6184          FILE PAGE SIZE
***         ******              MINIMUM SERVSIZE REQUIRED FOR THIS USER IS 17640                                          **********
***         ******              FIXED SERVER SIZE FOR THIS USER IS 3856                                                   **********
***         ******              INITIALIZATION COMPLETED.  BUFFERS = 59                                                   **********
***         ******              TIMELEFT = 3.739
OPEN CSFORT
***         ******              FILE CSFORT   OPENED
RESET UDDLPP=0,UDDCCC=80
UDDLPP    0             U DDNAME - LINES PER PAGE
UDDCCC    80            U DDNAME - CONT CHAR COLUMN
USE OUTPRIN
VIEW ASIZE
ASIZE     3             PAGES IN TABLE A
VIEW ARETRIES
ARETRIES  0             TABLE A PAGE RETRIES
VIEW BSIZE
BSIZE     20            PAGES IN TABLE B
TABLEB LIST RECLEN
PAGE NO.       FREE SPACE      FREE SLOTS
    0              4242             0
    1              2914             0
    2              5219             0
    3              6112             0
    4              6112             0
    5              6112             0
    6              6112             0
    7              6112             0
    8              6112             0
    9              6112             0
   10              6066             0
   11              5798             0
   12              5804             0
   13              6024             0
   14              6032             4
5658          AVG. FREE SPACE PER PAGE
0             AVG. FREE SLOTS PER PAGE
15            NUMBER OF PAGES PROCESSED

14            BRECPPG - TABLE B RECORDS PER PAGE
491           BRESERVE - TABLE B RESERVED SPACE PER PAGE
189           AVG. RECORD LENGTH
VIEW BHIGHPG
BHIGHPG   14            TABLE B HIGHEST ACTIVE PAGE
VIEW CSIZE
CSIZE     5             PAGES IN TABLE C
TABLEC
***         ******              NUMBER OF SLOTS= 5120
***         ******              SLOTS USED = 124
```

```
***      ******          PERCENTAGE OF TABLE C USED = 2
VIEW CRETRIES
CRETRIES  0          TABLE C PAGE RETRIES
VIEW DSIZE
DSIZE     35         PAGES IN TABLE D
VIEW DPGSUSED
DPGSUSED  22         TABLE D PAGES USED
NEW PAGE
```

```
VIEW SYSTEM
VERSION    6.2           RELEASE OF MODEL 204
SYSOPT     X'C0'         SYSTEM OPTIONS
OPSYS      X'40'         OPERATING SYSTEM
SYSID      A168          SMF SYSTEM IDENTIFICATION
LSERVPD    520           LENGTH OF SERVER PDL
NUSERS     1             NUMBER OF USERS
NSERVS     1             NUMBER OF SERVERS
SNAPID     0             NUMBER OF SNAPS SO FAR
LENQTBL    6             # OF RESOURCE ENQ TABLE ENTRIES PER USER
NSUBTKS    4             MAX NUMBER OF PSEUDO-SUBTASKS
NFILES     2             NUMBER OF FILE AREAS
NDCBS      10            NUMBER OF FILE DCBS
PAGESZ     6184          FILE PAGE SIZE
NUMBUF     59            NUMBER OF BUFFERS
EXCPVR          X'0000'  EXCPVR APPENDAGE ID
PAGEFIX    X'00000000'   PAGE-FIX PARAMETER STRING
SPCORE     10992         SPARE CORE RESERVED
TIMELEF    3739          TIME LEFT BEFORE END OF RUN
MAXTIME    5242          MAXIMUM RUN TIME
MINBUF     3             MINIMUM NUMBER OF BUFFERS
MAXBUF     256           MAXIMUM NUMBER OF BUFFERS
LOGADD     0             ROOM FOR PASSWORD ADDITIONS
IFAMBS     2048          IFAM2 BUFFER SIZE
TIMESVC    0             CCA TIMING SVC #
SMFSVC     0             SMF SVC #
SMFLORN    0             SMF LOGOUT REC #
SMFSLRN    0             SMF "SINCE LAST" REC #
LRETBL     200           LENGTH OF RECORD ENQ TABLE
RCVOPT     X'00'         RECOVERY OPTIONS
CPTIME     0             MINUTES BETWEEN CHECKPOINTS
CPTO       0             CHECKPOINT TIMEOUT IN MINUTES
CPSORT     1             IFAM2 CHKP SIGNON RETRY
CPTQ       0             BATCH QUIESCE TIMEOUT
CPMAX      32767         MAX # OF CHECKPOINTS
NDIR       7             NUMBER OF FILE DIRECTORY ENTRIES
LPFORCE    X'00'         FORCE PROCEDURE AUDIT
NGROUP     5             MAX # OF OPEN GROUPS
TERMBUF    1             # OF TERMINAL BUFFERS
ACCTIM     0             TIME BEFORE PARTIAL STAT LINES
SMPLTIM    0             PERFORMANCE SAMPLE TIME
RPTCNT     0             SAMPLES BETWEEN PERFORMANCE STAT LINES
LRUPG      0             LRU PAGES REMEMBERED
LRUTIM     0             TIME TO FLUSH OBSOLETE LRU REFERENCES
NEW PAGE
```

```
VIEW CWAIT
CPUSLICE   200          CPU SLICE - CPU
IOSLICE    75           CPU SLICE - IO
SRVSLICE   500          SERVER SLICE - CPU
SIOSLICE   250          SERVER SLICE - IO
INCSLICE   0            INCLUDE SLICING LIMIT
WAITSCAN   250          WAITSCAN INTVL
AGESCAN    0            AGESCAN INTVL
AGEINTVL   86400000     AGEING INTVL
AGEINCR    0            AGEING PRTY INCR
AGESLICE   0            AGEING SLICE INCR
SLICEMAX   0            MAX AGE SLICE
PRIOMAX    0            MAX AGE PRTY
NEW PAGE
```

```
VIEW USER
IFAMCHNL   IFAMPROD     IFAM2 CRAM CHANNEL NAME
CRIOCHNL   M204PROD     USER LANGUAGE CRAM CHANNEL NAME
VTAMNAME   M204         VTAM APPLICATION NAME
CRFSCHNL   M204FULL     FULL SCREEN CRAM NAME
UPRIV      X'80'        USER PRIVILEGE BITS
CURPRIV    X'BFFF'      PRIVS FOR CURRENT FILE/GROUP
CURCLASS   0            USER PRIV CLASS (PROC) FOR FILE/GROUP
CURSLVL    0            SELECT FLS LEVEL FOR FILE/GROUP
CURRLVL    0            READ FLS LEVEL FOR FILE/GROUP
CURULVL    0            UPDATE FLS LEVEL FOR FILE/GROUP
CURALVL    0            ADD FLS LEVEL FOR FILE/GROUP
CURREC     0
TERMOPT    0            TERMINAL OPTIONS
MCNCT      86400        MAX CONNECT TIME
MCPU       86400000     MAX CPU TIME
MDKRD      1000000      MAX DISK READS
MDKWR      1000000      MAX DISK WRITES
MOUT       1000000      MAX OUTPUT LINES
MUDD       1000000      MAX UDD LINES
MBSCAN     -1           MAX TABLE B RECORDS SCAN
VTYPE      STRING       DEFAULT %VAR TYPE
VLEN       20           DEFAULT %VAR LENGTH
VDP        0            DEFAULT %VAR DECIMAL PLACES
PRIORITY   STANDARD     USER PRIORITY
ERMX       30           MAXIMUM NUMBER OF ERRORS
TABSP      10           TAB SPACING
LECHO      X'01'        LINE ECHO
LEECHO     X'00'        LINE EDIT ECHO
LSECHO     X'00'        LINE SUBSTITUTION ECHO
CECHO      X'00'        CARD ECHO
CEECHO     X'00'        CARD EDIT ECHO
CSECHO     X'00'        CARD SUBSTITUTION ECHO
EDIT       X'00'        EDIT CONTROL
SUB        X'04'        SUBSTITUTION CONTROL
PROMPT     X'09'        PROMPT CONTROL
HDRCTL     X'00'        HEADER CONTROL
OUTPNO     5            OUTPUT PAGE NUMBER
OUTLPP     56           OUTPUT LINES PER PAGE
ERASE      @   X'7C'    ERASE-CHARACTER SYMBOL
FLUSH      #   X'7B'    ERASE-LINE SYMBOL
LINEND     ;   X'5E'    LOGICAL LINE END SYMBOL
PAGE       P   X'D7'    BACKPAGE COMMAND CHARACTER
UDDCCC     80           U DDNAME - CONT CHAR COLUMN
UDDLPP     0            U DDNAME - LINES PER PAGE
UDDRFM     X'12'        U DDNAME - RECORD FORMAT
PGSEP      2            LINES BETWEEN PAGES
ENQRETRY   4            RETRY RECORD ENQUEUING
LAUDIT     X'01'        LINE AUDIT BITS
CAUDIT     X'00'        CARD AUDIT BITS
IODEV      1            TYPE OF I/O DEVICE
LINENO     1            BTAM - RELATIVE LINE NUMBER
OUTMRL     132          MAX OUTPUT RECORD LENGTH
OUTCCC     132          OUTPUT CONT CHAR COLUMN
```

```
INMRL     80          MAX INPUT RECORD LENGTH
INCCC     72          INPUT CONT CHAR COLUMN
NBKPG     0           NUMBER OF BACKPAGES KEPT
POLLNO    0           RELATIVE POLLING NUMBER
NOTERM    0           NUMBER OF TERMINALS ON LINE
DVADD     0           DEVICE ADDRESS
TERMID                TCAM/VTAM TERMINAL ID
NEW PAGE
```

```
VIEW UTABLE
 LIBUFF     255          LENGTH OF INPUT BUFFER
 LOBUFF     256          LENGTH OF OUTPUT BUFFER
 LPDLST     1000         LENGTH OF USER PUSH DOWN LIST
 LQTBL      400          LENGTH OF QTBL
 LNTBL      50           LENGTH OF NTBL
 LSTBL      600          LENGTH OF STBL
 LTTBL      50           LENGTH OF TTBL
 LVTBL      50           LENGTH OF VTBL
 LGTBL      88           LENGTH OF GLOBAL VARIABLE TABLE
 LFTBL      1000         LENGTH OF FTBL
 LXTBL      1000         LENGTH OF XTBL
 LITBL      0            LENGTH OF ITBL
 LOUTPB     0            LENGTH OF OUTPUT PAGE BUFFER
 LFSCB      0            LENGTH OF FULL SCREEN BUFFER
 NORQS      5            NUMBER OF REQUESTS PRESERVED
 HTLEN      132          MAX LENGTH OF EACH HEADER OR TRAILER
 MAXHDR     5            MAX NUMBER OF HEADERS
 MAXTRL     5            MAX NUMBER OF TRAILERS
 LSERVER    17640        SERVER SIZE REQUIREMENT
 FIXSIZE    3856         FIXED SERVER SIZE REQUIREMENT
 NEW PAGE
```

```
VIEW FPARMS
CURFILE   CSFORT       CURRENT FILE
FISTAT    X'00'        (OK) CURRENT STATUS OF FILE
FOPT      X'00'        FILE OPTIONS
FRCVOPT   X'00'        FILE RECOVERY OPTIONS
OPENCTL   X'B0'        (PUBLIC) OPEN CONTROL FLAGS
PRIVDEF   X'BFFF'      DEFAULT FILE PRIVILEGES
PRCLDEF   0            DEFAULT USER CLASS FOR PROCEDURES
SELLVL    0            DEFAULT SELECT FLS LEVEL
READLVL   0            DEFAULT READ FLS LEVEL
UPDTLVL   0            DEFAULT UPDATE FLS LEVEL
ADDLVL    0            DEFAULT ADD FLS LEVEL
SECTY     X'00'        LOW-LEVEL SECURITY IN EFFECT
FILEORG   X'00'        (STANDARD) TABLE B OPTIONS
FIFLAGS   X'01'        FILE STATUS SWITCHES
VERIFY    X'00'        WRITE VERIFY
IVERIFY   X'00'        WRITE VERIFY - INITIALIZATION
NEW PAGE
```

added copy


```
VIEW TABLES
CURFILE    CSFORT        CURRENT FILE
ASTRPPG    614           TABLE A STRINGS PER PAGE
ATRPG      1             TABLE A ATTRIBUTE PAGES
FVFPG      1             TABLE A FEW VALUED FIELD PAGES
MVFPG      1             TABLE A MANY VALUED FIELD PAGES
ASIZE      3             PAGES IN TABLE A
BSIZE      20            PAGES IN TABLE B
CSIZE      5             PAGES IN TABLE C
DSIZE      35            PAGES IN TABLE D
FREESIZE   19            PAGES IN FREE SPACE
ARETRIES   0             TABLE A PAGE RETRIES
BRECPPG    14            TABLE B RECORDS PER PAGE
BRESERVE   491           TABLE B RESERVE SPACE
BPGPMSTR   0             TABLE B PAGES/MASTER AREA
BPGPOVFL   0             TABLE B PAGES/OVERFLOW AREA
BEXTOVFL   0             TABLE B EXTRA OVERFLOW AREAS
HIGHSORT   -1            RECORD WITH HIGHEST SORT KEY
BLOWPG     0             TABLE B LOWEST ACTIVE PAGE
BHIGHPG    14            TABLE B HIGHEST ACTIVE PAGE
MSTRADD    206           RECORDS ADDED TO MASTER AREAS
OVFLADD    0             RECORDS ADDED TO OVERFLOW AREAS
EOVFLADD   0             RECORDS ADDED TO EXTRA OVERFLOW AREAS
EXTNADD    0             EXTENSION RECORDS ADDED
MSTRDEL    136           RECORDS DELETED FROM MASTER AREAS
OVFLDEL    0             RECORDS DELETED FROM OVERFLOW AREAS
EOVFLDEL   0             RECORDS DELETED FROM EXT OVFLOW AREAS
EXTNDEL    0             EXTENSION RECORDS DELETED
BREUSED    0             RECORDS ADDED REUSING RECORD NUMBERS
SPILLADD   0             SPILL INDICATOR - ADDS
SPILLDEL   0             SPILL INDICATOR - DELETES
BSLOTPGS   -1            TABLE B # PGS ON WHICH CAN BEGIN RECORD
CRETRIES   0             TABLE C PAGE RETRIES
DRESERVE   15            TABLE D RESERVE SPACE
DACTIVE    1             TABLE D ACTIVE PAGE
DPGSUSED   22            TABLE D PAGES USED
PDSIZE     1             PAGES PER PROCEDURE DICTIONARY SECTION
PDSTRPPG   204           MAX.NO. NAMES/PROCEDURE DICTIONARY PAGE
EOJ
```