EXAM GENERATION SYSTEM

by

KOKA RAVINDRA

B. E., (Electronics and Communication Engineering)

University of Madras, India, 1971

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1973

Approved by:

Major Professor

TABLE OF CONTENTS

CHAPTER PAGE

APPENDICES

SAMPLE OUTPUTS

SOURCE LISTINGS AND FLOW CHARTS

CHAPTER 1

INTRODUCTION AND PROBLEM DEFINITION

Information processing is generally viewed as the manipulation and organization of information in a purposeful way. The advent of computers and their unique capability to store and manipulate vast quantities of information has enabled great advancements in the development of information processing systems. Most organizations, whether scientific or business, are faced with the problem of an information explosion and the computer is a very effective tool to help overcome this problem so that more time may be spent on meaningful and constructive work rather than on the tedious and irritating job of retrieving necessary information each time it is needed. Numerous information processing systems covering a wide range of areas such as the military, business, industry, medicine and education have been successfully developed and implemented. The basic objective underlying all these systems is to access the relevant information in as short a period as possible and to present it to the user in the form he desires.

It was with this objective in mind that the Exam Generation System was developed, and it is hoped that it would assist faculty or anyone associated with the teaching profession in constructing exams for their courses. This system will enable a user to create a database of questions and answers for a particular course, and to obtain a formatted printout of the questions by merely specifying the numbers of the questions he wishes to appear in an exam. The formatted exam will have answer space alloted for each question and this allocation is controlled by the user. An output with the solutions for the questions is also provided for the user's reference.

There might be criticism from certain quarters about such a system as it could standardize the contents of the exams being generated. To counter this, updating of the database has been made very convenient and the user can make changes whenever he wishes to. This should hopefully lead to a dynamic and healthy system.

The factors governing the design of the system are listed below:

1) It is general purpose in the type of data to be accomodated with a restriction on the record size (which could be modified by the user).

2) It enables the user to create a database on an auxiliary unit of his choice by supplying the data in the prescribed format.

3) It enables the user to add or delete data entries to the database.

4) It enables the user to obtain three types of printout.

    (i) A printout of the masterfile.

    (ii) A printout of the exam consisting of the questions retrieved on his request.

    (iii) A printout of the exam along with the solutions to the questions.

5) It operates in a batch environment.

SYSTEM ENVIRONMENT

The system uses the IBM 360/50 at the Computing Center, Kansas State University, with peripheral units such as disk, tape, card reader and line printer. The system is written in PL/1, and the database is a direct access, regional (3) file with undefined length records.

CHAPTER 2

SYSTEM DESCRIPTION

2.1 Introduction

The Exam Generation System is split up into three programs:

(i) Program QACREAT - To create the database

(ii) Program QAUPDAT - To update the database

(iii) Program QAPRINT - To retrieve and print the questions specified for the exam.

These programs are invoked by the cataloged procedures QUIZGENC, QUIZENU and QUIZGENP.

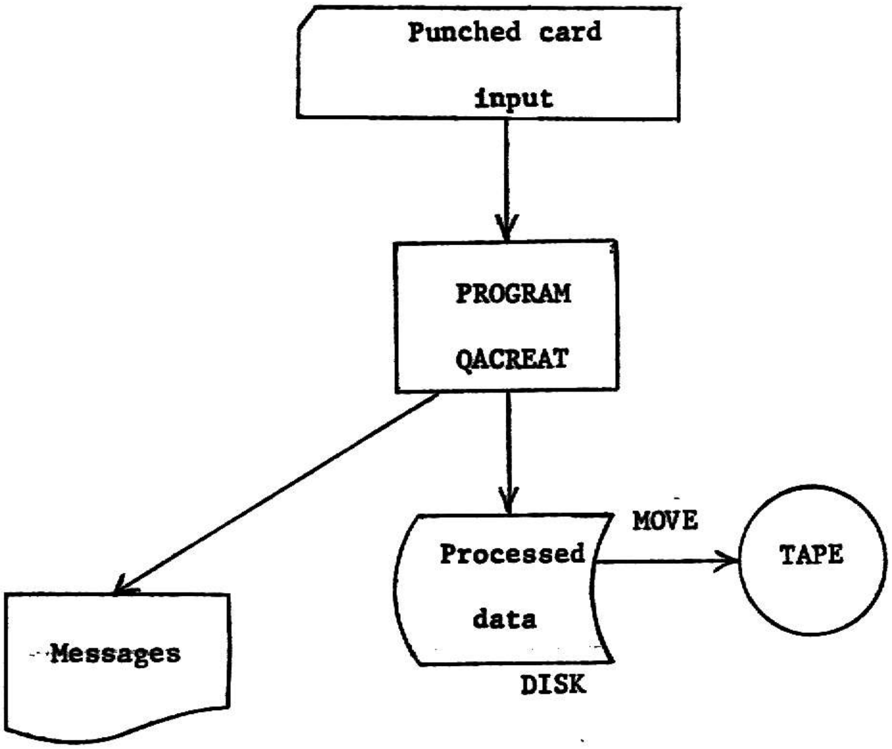
Program QACREAT accepts character data from cards and processes it in accordance with the format code before storing it in the database. The database is initially created on disk and then moved to tape.

Program QAUPDAT updates the database, i.e. adds or deletes data entries. While adding, the data is to be processed in the same way as while creating the database. The question numbers identifying the records to be deleted are read in from cards and the source keys generated. The database is moved to disk updated and then moved back to tape.

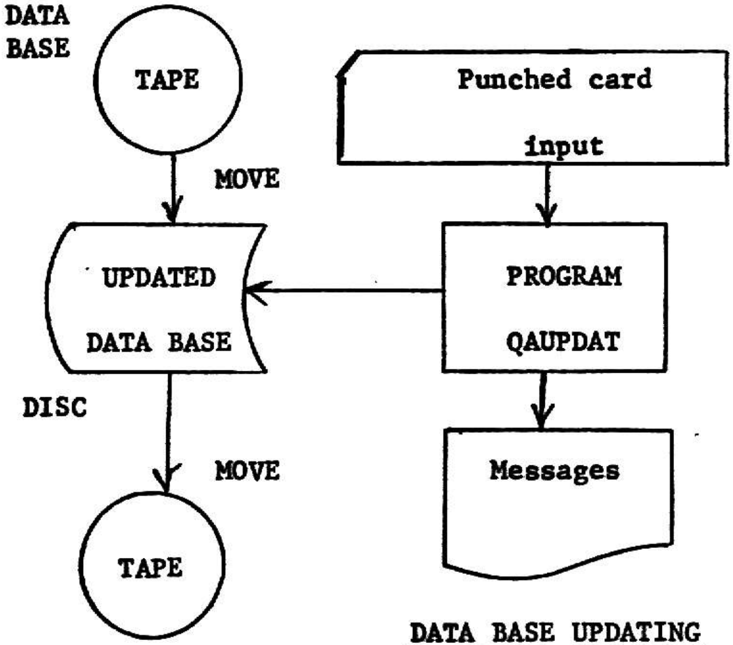
Program QAPRINT retrieves questions identified by the question numbers which the user specifies. It then formats them and prints out the examination and the examination with solutions. A printout of the masterfile is also provided if the MASTFIL option is turned on.

The outline of the system is shown in the next section.

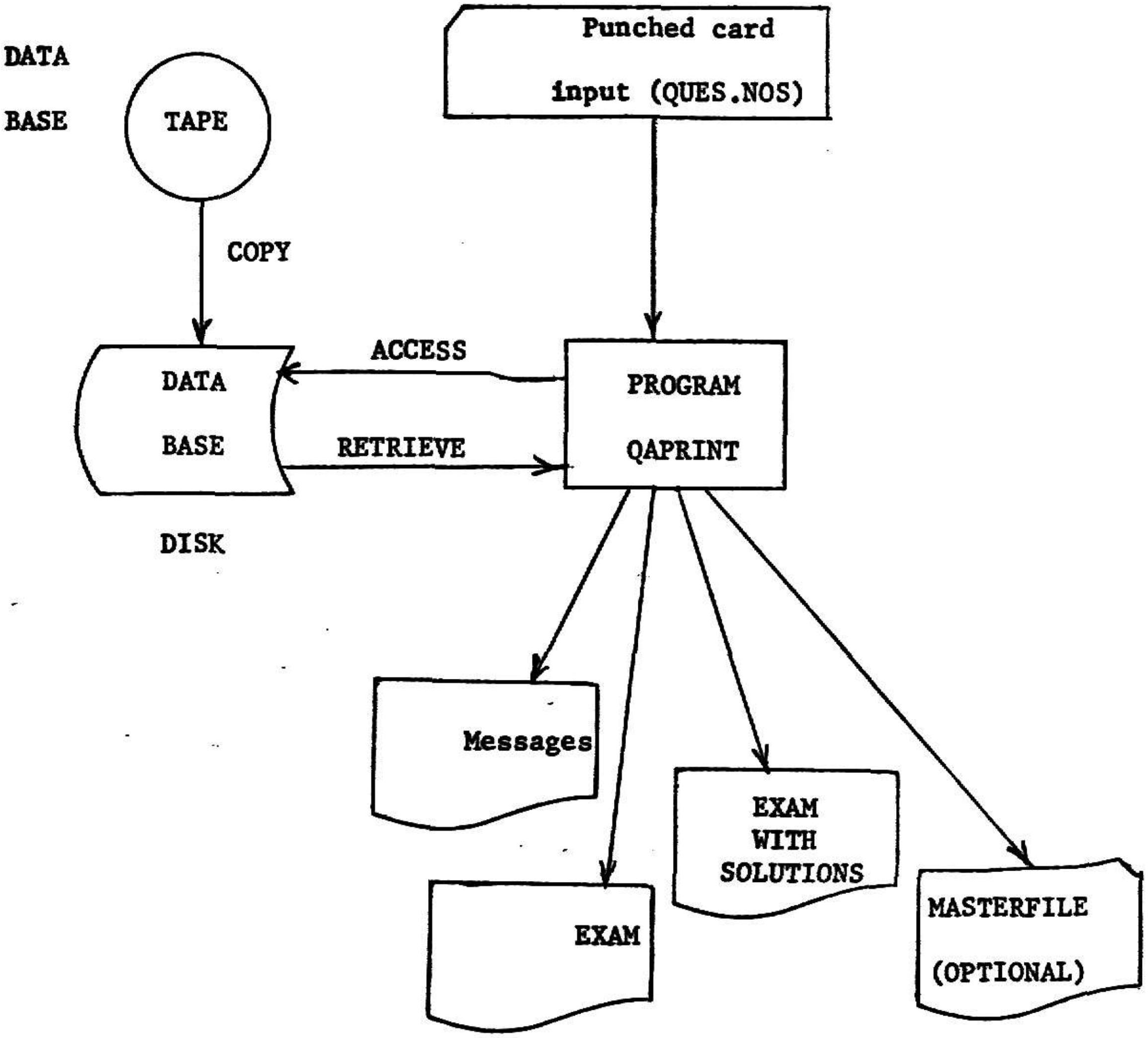
2.2 OUTLINE OF THE SYSTEM



DATA BASE CREATION



DATA BASE UPDATING



EXAM GENERATION

2.3 DATA BASE CREATION

The database is created by invoking the cataloged procedure QUIZGENC which accepts character information punched on cards and operates on it according to the format code before storing the information in the database. The database is a direct access, keyed, regional (3) file with record length undefined. The reasons for choosing regional (3) organization are discussed in a later part of this chapter.

The procedure for creating the database can be thought of as consisting of the following main steps:

1. Reading in data from cards.
2. Operating on the data according to the format code for convenience and efficiency while storing.
3. Generating a source key which will enable direct retrieval of a record when needed.
4. Storing the processed information in the database.
5. Moving the database to a tape unit.

Step 1: The data is read in from cards, 80 characters at a time. End of each data entry is indicated by the symbol $END.

Step 2. Two format codes 'U' and 'D' which determine the form in which the question is to be printed are available. A default code value of 'D' is assumed in the event of the user not specifying a format code. Format code 'U' indicates that the user wants his data on each card to be reproduced on a fresh line without suppressing the blanks preceeding the information on a card. Format code 'D' indicates that all blanks preceeding and following the information on a card can be suppressed and that the information need not begin in a fresh line. The maximum length of the line is fixed at 50 and hence when reproduced it will have as many 50 character lines as possible and the remaining information placed, left justified in the line.

It is convenient to store and manipulate the information pertaining to a question in a single logical record. To enable this the data read in, is first operated upon. It is also desirable to conserve storage space by suppressing blanks preceeding and following the relevant information on a card. In the case of format 'D', the relevant information from a card is isolated and concatenated with information on successive cards of the data entry since data on a card need not begin in a fresh line. However in the case of format 'U' it is necessary that a special character '@' be inserted on the disk and tape records at the right of the information from each card indicating the beginning of a fresh line. Also, in this case the blanks preceeding the information on a card are left intact since this is useful for data in the form of a program where the logical structure is to be preserved.

Step 3: The database as mentioned earlier is a regional (3) organization. The major advantage of this over other types of database organizations is that it allows control over the relative placement of records. The source key which identifies each record can be thought of as having two logical parts, the region number and a comparison key. The rightmost eight characters of the source key make up the region number. The question number is assigned to the data entries in the ascending order and serves as the comparison key. An algorithm which derives a region number from the question number is designed in such a way as to optimize the use of space within the dataset. Duplicate region numbers will occur, but their only effect may be to lengthen the search time for records with duplicate region numbers.

The algorithm used is a hashing technique which translates the question number to a corresponding unique memory location, i.e.,

Region number = MOD(Question number/10).

The question number concatenated with the region number forms the source key of each record. This algorithm generates ten records with identical region numbers. Since each region number corresponds to a track on the disc, the 2314 unit with a track capacity of 7244 bytes will accomodate ten records, with an average length of 720 bytes. If overflow occurs, the system searches the next track until it finds empty space. It is advisable to limit track overflows for efficiency in execution.

Step 4: The processed information along with question number, space code, and the format code is stored in the database in a location determined by the source key.

Step 5: The database created on disk is then moved to a tape unit specified by the user. IBM utility program IEHMOVE performs this operation, and since the devices are not compatible, it unloads the database maintaining the regional organization and then moves the database.

Additional features:

1. If any of the codes (space or format) are omitted, then a message indicating the default value assumed for a data entry is printed
2. A listing of the masterfile of questions and answers is provided to the user in the form in which they will be reproduced when retrieved for generating an examination.

2.4 DATA BASE UPDATING

The database can be updated by invoking the cataloged procedure QUIZGENU. Two update operations, ADD and DELETE, are available. Also the number of data entries to be added or deleted has to be specified. All update operations are performed on disk, and the dataset has to be moved to disk from tape and after updating moved back to tape.

The procedure for adding a data entry is the same as that described in Section 2.3, except that steps 3 and 4 are performed by a subroutine, ADD. The question numbers assigned to the data entries are in sequence with the numbers already existing in the dataset.

The procedure for deleting a data entry can be thought of as consisting of the following steps:

1. Reading in the question numbers identifying the records to be deleted from cards.
2. Generating source keys using the hashing technique described in Section 2.3.
3. Deleting the records identified by the source keys.

Subroutine DELETE performs steps 2 and 3 and is passed the question numbers read in through an array arguement.

The update operations are to be specified in the PARM parameter passed to the PL/1 main procedure. The data specified in the PARM parameter field can be up to 100 characters long and must be enclosed in quotation marks. Section 3.4 of the user's manual contains a detailed description of the parameter specification. The PL/1 program scans the PARM parameter and isolates the operations specified for that particular run.

Additional Features:

1. A message indicating the question numbers added or deleted is printed.
2. A listing of the updated masterfile of questions and answers is provided to the user in the form in which they will be reproduced when retrieved for generating an exam.
3. An ON KEY condition is enabled to take care of errors in specification of question numbers by the user while deleting data entries. A diagnostic message is generated informing the user that the data entry referred to does not exist in the dataset.

2.5 EXAM GENERATION

An exam can be generated by invoking the cataloged procedure QUIZGENP. The name of the dataset from which the questions are to be retrieved and the tape unit on which it resides are to be specified by the user. Also, the print options which are explained in detail in Section 3.4 of the user's manual should be specified by the user. However, a default value of QUESFIL is assumed in the event of the user not specifying an option.

The procedure for generating an exam can be though of as consisting of the following steps:

Step 1. Reading in the question numbers identifying the records to be printed out for the exam.

Step 2. Generating source keys using the hash technique described in the earlier Section 2.3.

Step 3. Retrieving records corresponding to these source keys and formatting the information in a record according to the format code.

Step 4. Printing the exam providing answer space for each question in accordance with the space code.

Steps 1 and 2 have been described in earlier sections.

Step 3: The record retrieved contains the information which has been pre-processed while creating the database. Hence the formatting operation differs depending on the type of format specified for this record. In the case of format 'D', the record is broken into as many 50 character blocks as possible and the remaining information padded with blanks to the right to make up a 50 character block. These blocks are printed out successively in Step 4,and answer space is provided by skipping a number of lines, which is determined by the space code before printing the next question.

In the case of format code 'U', the record is searched for the special character '@' and the information preceeding each '@' moved to a block of 80 characters. Blanks are padded at the right if the information is less than 80 characters. These blocks are printed out in the same way as in the previous case in Step 4.

The questions, when printed out, are numbered serially starting with 1. The title information is printed out if provided and also an 'EXAM #' and 'DATE'.

Additional features:

1. A printout of the questions along with their answers is also provided to the user each time an exam is generated. This option is turned on by default.
2. A printout of the masterfile can be obtained by turning the MASTFIL option on Section 3.4 of the user's manual contains more details.
3. If the user specifies a question number refering to a record which does not exist in the dataset, a diagnostic message is generated.

KANSAS STATE UNIVERSITY

USER'S MANUAL

FOR

QUIZGEN

Programmed by: KOKA RAVINDRA

ABSTRACT

QUIZGEN is a set of PL/1 programs which is designed to enable:

1) Creation of a database of questions and answers on an auxiliary unit.
2) Updating of the database created.
3) Generation of a formatted exam.

TABLE OF CONTENTS

INTRODUCTION

1. This manual is designed to provide the user the rudimentary information required to utilize the programs QACREAT, QAUPDAT, and QAPRINT which are invoked by the cataloged procedure QUIZGEN. These are a set of programs in PL/1 which enables the user to create a database of questions and answers on tape by providing the data on cards, and then to retrieve questions from this database to generate a formatted exam with answer space provided for each question. It also enables the user to update the database by providing the necessary control and data.

Program QACREAT reads in the character data punched on cards and, after operating on it, stores it in the database. The name of the database created and the tape unit on which it is to reside should be specified by the user.

Program QAPRINT will access the database created by the previous program and will retrieve those records which the user identifies by specifying question numbers. The exam is generated by this program and will consist of the retrieved questions. It will also provide the user with a printout of the exam with solutions for his reference. Print options are discussed in detail in Section (3.4).

Program QAUPDAT will add or delete data entries. The update operations and the data should be specified by the user. The program provides the user with a listing of the updated masterfile.

JOB CONTROL LANGUAGE (JCL) REQUIREMENTS

2. The Job Control Language (JCL) necessary to utilize the programs discussed in Section 1 is listed below. The various parameters on the EXEC card, the MOVE and COPY control cards and the data formats for the three programs are described in Sections 3 and 4.

Program QACREAT:

```
//Jobname      JOB (see KSU Computing Center User's Guide)
//             EXEC QUIZGENC,NAME=xxxx,[S=yyyy],
//             TAPE=uuuu,[PARM.CREATE='wwww']
//CREATE.SYSIN DD *
               data (questions and answers)
/*
//MOVE.SYSIN DD *
    b̸MOVE DSNAME=xxxx,TO=TAPE9=uuuu,FROM=2314=111111
/*
```

The parameters in square brackets are optional and if omitted, default values which are explained in Section 3, are assumed.

Program QAUPDAT:

```
//Jobname    JOB (see KSU Computing Center User's Guide)
//           EXEC QUIZGENU,NAME=xxxx,TAPE=uuuu,
//       PARM.UPDAT='wwww'
//MOVE.SYSIN DD *
     ƀMOVE DSNAME=xxxx,TO=2314=111111,FROM=TAPE9=uuuu
/*
//UPDAT.SYSIN DD *
             data
/*
//MOVE2.SYSIN DD *
     ƀMOVE DSNAME=xxxx,TO=TAPE9=uuuu,FROM=2314=111111
/*
```

<u>Program QAPRINT</u>:

```
//Jobname    JOB (see KSU Computing Center User's Guide)
//           EXEC QUIZGENP,NAME=xxxx,TAPE=uuuu,
//       [PARM.PRINT='wwww']
//MOVE.SYSIN DD *
     ƀCOPY DSNAME=xxxx,TO=2314=111111,FROM=TAPE9=uuuu
/*
//PRINT.SYSIN DD *
           data (question No's of the questions to be printed out)
/*
```

PARAMETER DESCRIPTION

3.1 NAME=xxxx Where xxxx is the name of the dataset which is created by invoking the cataloged procedure QUIZGENC consisting of the data provided by the user. This dataset will be moved to tape and saved for future use. The name of the dataset should be specified while invoking any one of the cataloged procedures QUIZGENU, QUIZGENP and QUIZGENC.

<u>NOTE</u>: The name should be enclosed in apostrophes if it contains any special characters.

<u>EXAMPLES</u>:

1) NAME = QUES600
2) NAME = 'CODP07.BREWER'

3.2 S=yyyy Where yyyy is the space parameter and specifies the space to be allocated for the dataset being created by QUIZGENC. It is optional and if omitted a default value will be assumed.

<u>NOTE</u>: The space value has to be enclosed in apostrophes. The default value is (1000, (15, 10)) and will accommodate 15 data entries with an average record length of 1000.

<u>EXAMPLES</u>:

1) S = '(TRK,(1,1))'
2) S = '(CYL,(1,1))'

3.3 TAPE=uuuu Where uuuu is the volume serial number of the tape unit on which the dataset is to reside. The dataset is initially created on disc and then moved to a user specified tape unit. The tape unit number should be specified while invoking any one of the cataloged procedures QUIZGENC, QUIZGENU or QUIZGENP so that the dataset can be accessed.

NOTE: The number should be enclosed in apostrophes, if it contains special characters.

EXAMPLES:

1) TAPE = 000506

2) TAPE = '2400-6'

3.4 PARM OPTIONS

The PARM parameter can be used to pass information to the program steps in the cataloged procedure. The various options provided in the three programs are described below.

Program QACREAT (QUIZGENC)

NO TITLE This option is provided so that the user need not provide title information for his database, if he chooses to. This option is to be passed to the CREATE step of the cataloged procedure QUIZGENC and has to be enclosed in apostrophes. If omitted, it is assumed that title information is being provided by the user.

EXAMPLE: PARM.CREATE = 'NO TITLE'

Program QAUPDAT (QUIZGENU)

ADD This option enables the user to add data entries to his database. The number of data entries to be added is to be specified and the format for that is 'ADD= no' where no is an integer number. If no is omitted then 1 is assumed. This option should be passed to the UPDAT step of QUIZGENU.

DELETE This option enables the user to delete existing records from his database. The number of records to be deleted is to be specified, just like in ADD and the format is 'DELETE=no' where no has the same meaning as before. This option is also passed to the UPDAT step of QUIZGENU. If no is omitted, then 1 is assumed.

NOTE: If both options ADD and DELETE are specified, then they should be separated by commas. There is no restriction on the order in which they are to be specified. At least one option has to be specified.

EXAMPLES:

(i) PARM.UPDAT = 'ADD,DELETE'

This indicates that one record is to be added and one to be deleted.

(ii) PARM.UPDAT = 'ADD=3'

This indicates that three records are to be added.

(iii) PARM.UPDAT = 'ADD=2,DELETE=4'

This indicates that two records are to be added and four deleted.

Program QAPRINT (QUIZGENP)

MASTFIL This option enables the user to obtain a formulated listing of the masterfile of questions and answers, in ascending order of the question numbers. This option is to be passed to the PRINT step of the cataloged procedure QUIZGENP and has to be enclosed in apostrophes. This option is turned off by default and has to be enabled.

QUESFIL This option enables the user to obtain a formatted exam and also a listing of the exam with the solutions. This option is turned on by default and has to be specified only if the default option is being overridden by the user. This also is to be passed to the PRINT step in QUIZGENP.

NOTE: Specification of an option overrides the default option.

EXAMPLES:

```
  (i) PARM.PRINT = 'MASTFIL'
 (ii) PARM.PRINT = 'QUESFIL'
(iii) PARM.PRINT = 'MASTFIL,QUESFIL'
```

In (i) the masterfile listing will be provided and in (ii) the formatted exam and the formatted exam with solutions will be printed out. In (iii) both the printouts of (i) and (ii) will be provided.

3.5 MOVE and COPY control cards:

```
ƀMOVE  DSNAME=xxxx,TO=TAPE9=uuuu,FROM=2314=111111
ƀCOPY
```

xxxx   is the name of the dataset to be moved and is the same as that described in Section 3.1.

uuuu   is the volume serial number of the tape unit to which the dataset is to be moved. It is the same as that mentioned in Section 3.3.

NOTE: MOVE deletes the dataset from the device from which it is being moved.

COPY leaves the dataset as it is on the device from which it is copied.

DATA FORMATS

4.1 Program QACREAT (QUIZGENC)

Title Information: This information is optional and if supplied, should precede all other data. The information consists of the following:

1. Course number
2. Course title
3. Name of the Instructor.

It should be in the above order and the information should be enclosed in apostrophes. The course number should not be longer than seven characters. The course title and Instructor's name can be of any length. If the user chooses to omit any of the information, then he should place a blank in that portion. The NO TITLE option should be turned on if he chooses to omit all title information.

EXAMPLES:

```
'286-600' 'DISCRETE STRUCTURES''R.K.BREWER'
'286-600' 'b̸' 'R.K.BREWER'
```

DATA ENTRY

The title information, if provided, should be followed by the data entries which consists of sets of questions and answers. Every question should start in a fresh card but may start in any column. The last card in the set of cards pertaining to a particular question and answer should contain $END. If there is no blank space on the last card, then an extra card with $END in it should be included. The format code, space code and answer are optional and can be omitted if the user chooses. The format for specifying these is given below.

The format code should be a single alphabetic character and the space code should be an integer number. They should be preceded by two ampersands, if specified. The format code should always precede the space code if both are specified. If an answer is being supplied, then it should be preceded by &ANS.

<u>FORMAT CODE</u>: &&U or &&D

U indicates that the user wants his data to appear exactly as he has punched them on cards. Preceding blanks will not be suppressed.

D indicates that the user wants his data to appear in consecutive lines with 50 characters in a line. All preceding and following blanks in a datacard will be suppressed.

If the format code is not specified by the user then it defaults to D.

<u>SPACE CODE</u>: &&<u>number</u>

The number can be any integer number and indicates the number of lines of blank space required for the solution of that particular question.

If the user does not specify a space code, then it defaults to 2.

EXAMPLES:

(i)

```
&&U &&5 WHY IS THE SKY BLUE &ANS BECAUSE
```

```
IT IS NOT RED $END
```

(ii)

```
&&3 WHY IS THE SKY BLUE $END
```

(iii)

```
 WHY IS THE SKY BLUE $END
```

4.2 Program QAUPDAT (QUIZGENU)

There are two data formats corresponding to ADD and DELETE operations.

ADD: The data format is the same as that described in 4.a except that there is no title information.

DELETE: The question numbers of the questions to be deleted are to be specified and they should be seperated by blanks. They should start in a fresh card and can be continued on successive cards.

NOTE: The data for ADD and DELETE should be in the same order as specified in the parm option of QUIZGENU.

EXAMPLES:

(i) For PARM.UPDAT='ADD,DELETE=3' the format would be

```
Card 1      WHY IS THE SKY BLUE $END

Card 2       5  6  8
```

(ii) For PARM.UPDAT='DELETE=4,ADD' the format would be

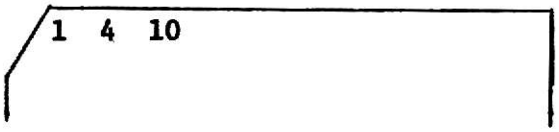
```
Card 1          5  6  8  10
Card 2          WHY IS THE SKY BLUE $END
```

4.3 Program QAPRINT (QUIZGENP)

The data consists of the question numbers which are to be printed out for that particular exam. The question numbers should be seperated by at least one blank and should start in a fresh card. They can be continued on successive cards.

EXAMPLES:

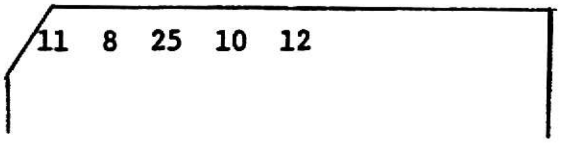
(i)



```
1  4  10
```

Questions 1, 4 and 10 of the masterfile will appear in the exam in this case.

(ii)



```
11  8  25  10  12
```

Questions 11, 8, 25, 10 and 12 will be printed out in the exam.

APPENDIX A.1

This is the output obtained by invoking the cataloged procedure QUIZGENC. It shows the masterfile created.

MESSAGES

```
DATA 1  FORMAT CODE NOT SPECIFIED.DEFAULTS TO D
DATA 2  SPACE CODE NOT SPECIFIED.DEFAULT VALUE ASSUMED
DATA 3  SPACE CODE NOT SPECIFIED.DEFAULT VALUE ASSUMED
DATA 4  SPACE AND FORMAT CODES NOT SPECIFIED.DEFAULT VALUES ASSUMED
DATA 5  SPACE AND FORMAT CODES NOT SPECIFIED.DEFAULT VALUES ASSUMED
DATA 6  SPACE CODE NOT SPECIFIED.DEFAULT VALUE ASSUMED
```

1) WRITE A PL/1 ALGORITHM TO CREATE A DOUBLY LINKED LIST.

```
2)  TT:    PROC OPTIONS (MAIN);
                DCL A CHAR(1);
                CALL START;
                GET LIST(A);
                I=MAK(A);
                PUT LIST(A,I)F .
         AA:    GET LIST(A);
                I=LUK(A);
                PUT LIST(A,I);
                   IF I<0 THEN
                   I=MAK(A);
                PUT LIST(A,I);
                GOTO AA;
         START:    PROC;
                   DCL (I,J,K,LINK(0:100,2)) FIXED BINARY STATIC;
                   DCL A CHAR(*), (WORD,X(0:100)) CHAR(30) STATIC;
                   LINK=0;
                   X=' ';
                   I,J=0;
                   K=1;
                   RETURN;
         LUK:      ENTRY(A)RETURNS(FIXED BIN);
                   I=1;
                   WORD=A;
         RE:          IF WORD=X(I) THEN
                      RETURN(I);
                      IF WORD>X(I) THEN
                      K=2;
                      ELSE
                      K=1;
                      IF LINK(I,K)=0 THEN
                      RETURN(-I);
                      ELSE
                      I=LINK(I,K);
                   GOTO RE;
         MAK:      ENTRY(A)RETURNS(FIXED BIN);
                   WORD=A;
                   J=J+1;
                   X(J)=WORD;
                   LINK(I,K)=J;
                   RETURN(J);
                   END START;
                END TT;

 'A' 'B' 'C' 'A' 'C' 'D' 'E' 'D'

  /* THE ENTRY "LUK" IS ALWAYS USED, BOTH IN SEARCHING THE LIST X AND */
  /*IN ADDING TO IT.  WHEN AN ITEM IS TO BE ADDED TO THE LIST, LUK IS */
  /*INVOKED.  IF A MINUS RETURN IS OBTAINED, THE ITEM IS NOT ON THE   */
  /*LIST AND MAK IS INVOKED, ADDING THE ITEM TO THE LIST.  MUTIPLE    */
  /*COPIES OF A WORD MAY NOT BE PUT ON THE LIST.                      */
```

A) GIVEN THE ABOVE CODE, WHAT SORT OF LIST DOES IT BUILD? (4 PTS)

B) GIVEN THE FOLLOWING INPUT THAT IS ASSUMED ADDED TO THE LIST ONE ITEM AT A TIME, GRAPH THE RESULTANT LIST. (4 PTS)

MJQCPXDRWISKAZ

3) GIVEN STORAGE BLOCKS OF 3000,2000, AND 1000 UNITS, RESPECTIVELY, WOULD THE BEST FIT OR FIRST FIT STRATEGY ALLOW THE MAXIMUM ALLOCATIONS GIVEN REQUESTS OF 1000, 2000, AND 3000 UNITS, RESPECTIVELY? WHICH IS USUALLY THE BEST STRATEGY ACCORDING TO KNUTH?

A)____________________________(1.5 PTS)

B)____________________________(1 PTS)

4) GIVE LOC(A J,K), IF A IS THE MATRIX GIVEN, AND IF EACH NODE OF THE 119JHG IS , WORDS LONG, ASSUMING THE NODES ARE STORED CONSECUTIVELY IN LEXICOGRAPHIC ORDER OF THE INDICES. N=4, M=5, J=2, K=3 (10 PTS)

5) GIVEN THE FOLLOWING LIST, GRAPH IT AND GIVE A MEMORY REPRESENTATION USING HEADED, SINGLY LINKED, NON-CIRCULAR LIST REPRESENTATION. (9 PTS) L=(:L, :N, , :(:L)) N=(:(N), :())

6) GIVEN THE FOLLOWING TREE:
A) THREAD IT IN POST ORDER. (9 PTS)
B) GIVE THE P$ OF THE NODE CONTAINING F. (HINT: *,$,# ARE PRE,POS & END) (2.5 PTS)
C) GIVE THE PRE AND END ORDER TRAVERSALS OF THE TREE. (4 PTS)

7) WHICH OF THE FOLLOWING DO NOT MATCH?

SUN:HEAT
MOON:LUNATIC
GIRL:EMOTION
PLACE:SENTIMENT
MIND:HAYSTACK

APPENDIX A.2

This is the output obtained by invoking the cataloged procedure QUIZGENU. It shows the updated masterfile.

```
                                   MESSAGES


DATA 1  SPACE CODE NOT SPECIFIED.DEFAULT VALUE ASSUMED
          8  HAS BEEN ADDED TO THE DATASET
          9  HAS BEEN ADDED TO THE DATASET

DATA 3  SPACE CODE NOT SPECIFIED.DEFAULT VALUE ASSUMED
         10  HAS BEEN ADDED TO THE DATASET

DATA 4  FORMAT CODE NOT SPECIFIED.DEFAULTS TO D
         11  HAS BEEN ADDED TO THE DATASET

DATA 5  SPACE AND FORMAT CODES NOT SPECIFIED.DEFAULT VALUES ASSUMED
         12  HAS BEEN ADDED TO THE DATASET

DATA 6  SPACE AND FORMAT CODES NOT SPECIFIED.DEFAULT VALUES ASSUMED
         13  HAS BEEN ADDED TO THE DATASET

DATA 7  SPACE AND FORMAT CODES NOT SPECIFIED.DEFAULT VALUES ASSUMED
         14  HAS BEEN ADDED TO THE DATASET
```

286_600 DISCRETE STRUCTURES R.K.BREWER

1) WRITE A PL/1 ALGORITHM TO CREATE A DOUBLY LINKED LIST.

```
2)  TT:    PROC OPTIONS (MAIN);
                DCL A CHAR(1);
                CALL START;
                GET LIST(A);
                I=MAK(A);
                PUT LIST(A,I)F
        AA:     GET LIST(A);
                I=LUK(A);
                PUT LIST(A,I);
                   IF I<0 THEN
                   I=MAK(A);
                PUT LIST(A,I);
                GOTO AA;
        START:     PROC;
                   DCL (I,J,K,LINK(0:100,2)) FIXED BINARY STATIC;
                   DCL A CHAR(*), (WORD,X(0:100)) CHAR(30) STATIC;
                   LINK=0;
                   X=' ';
                   I,J=0;
                   K=1;
                   RETURN;
        LUK:       ENTRY(A)RETURNS(FIXED BIN);
                   I=1;
                   WORD=A;
        RE:           IF WORD=X(I) THEN
                      RETURN(I);
                      IF WORD>X(I) THEN
                      K=2;
                      ELSE
                      K=1;
                      IF LINK(I,K)=0 THEN
                      RETURN(-I);
                      ELSE
                      I=LINK(I,K);
                   GOTO RE;
        MAK:       ENTRY(A)RETURNS(FIXED BIN);
                   WORD=A;
                   J=J+1;
                   X(J)=WORD;
                   LINK(I,K)=J;
                   RETURN(J);
                   END START;
                END TT;
```

'A' 'B' 'C' 'A' 'C' 'D' 'E' 'D'

```
/* THE ENTRY "LUK" IS ALWAYS USED, BOTH IN SEARCHING THE LIST X AND */
/*IN ADDING TO IT.  WHEN AN ITEM IS TO BE ADDED TO THE LIST, LUK IS */
/*INVOKED.  IF A MINUS RETURN IS OBTAINED, THE ITEM IS NOT ON THE   */
/*LIST AND MAK IS INVOKED, ADDING THE ITEM TO THE LIST.  MUTIPLE    */
/*COPIES OF A WORD MAY NOT BE PUT ON THE LIST.                      */
```

A) GIVEN THE ABOVE CODE, WHAT SORT OF LIST DOES IT BUILD? (4 PTS)

B) GIVEN THE FOLLOWING INPUT THAT IS ASSUMED ADDED TO THE LIST ONE ITEM AT A TIME, GRAPH THE RESULTANT LIST. (4 PTS)

MJQCPXDBWTSKAZ

3) GIVEN STORAGE BLOCKS OF 3000,2000, AND 1000 UNITS, RESPECTIVELY, WOULD THE BEST FIT OR FIRST FIT STRATEGY ALLOW THE MAXIMUM ALLOCATIONS GIVEN REQUESTS OF 1000, 2000, AND 3000 UNITS, RESPECTIVELY? WHICH IS USUALLY THE BEST STRATEGY ACCORDING TO KNUTH?

A)______________________________(1.5 PTS)

B)______________________________(1 PTS)

4) GIVE LOC(A J,K), IF A IS THE MATRIX GIVEN, AND IF EACH NODE OF THE 1I9JHO IS , WORDS LONG, ASSUMING THE NODES ARE STORED CONSECUTIVELY IN LEXICOGRAPHIC ORDER OF THE INDICES. N=4, M=5, J=2, K=3 (10 PTS)

5) GIVEN THE FOLLOWING LIST, GRAPH IT AND GIVE A MEMORY REPRESENTATION USING HEADED, SINGLY LINKED, NON-CIRCULAR LIST REPRESENTATION. (9 PTS) L=(:L, :N, , :(:L)) N=(:(N), :())

6) GIVEN THE FOLLOWING TREE:
A) THREAD IT IN POST ORDER. (9 PTS)
B) GIVE THE P$ OF THE NODE CONTAINING F. (HINT: *,$,# ARE PRE,POS & END) (2.5 PTS)
C) GIVE THE PRE AND END ORDER TRAVERSALS OF THE TREE. (4 PTS)

7) WHICH OF THE FOLLOWING DO NOT MATCH?
SUN:HEAT
MOON:LUNATIC
GIRL:EMOTION
PLACE:SENTIMENT
MIND:HAYSTACK

8) IF THE FOLLOWING TREE IS ADDED TO THE RIGHT OF THE TREE IN 10,
A) GIVE THE NATURALLY CORRESPONDING BINARY TREE TO THAT FOREST. (4 PTS)
B) GIVE THE POSTORDER SEQUENTIAL REPRESENTATION OF IT.

9) GIVEN THE FOLLOWING CODE AND DATA,

```
DEVICE:PROC OPTIONS (MAIN);
DCL ARRAY (101) CHAR (1);
DCL (N,OP) CHAR (1);
J,K=51;
A:GET EDIT (OP) (A(1));
IF OP=' '.THEN GOTO L99;
IF OP>'B' THEN
   DO;
   PUT LIST (ARRAY(K));
   K=K-1;
   GOTO A;
   END;
GET EDIT (N) (A(1));
IF OP<'B' THEN
   DO;
   K=K+1;
```

```
      ARRAY(K)=N;                                                          17
      END;                                                                 18
   ELSE                                                                    19
      DO;                                                                  20
      ARRAY(J)=N;                                                          21
      J=J-1;                                                               22
      END;                                                                 23
   GOTO A;                                                                 24
   L99:                                                                    25
   END DEVICE;                                                             26
  A1B2CDA3A4B5B6CB7A8CDCCC
```

A) GIVE THE FORMAL NAME OF THE DEVICE IT SIMULATES, (2.5 PTS.)

B) GIVE THE OUTPUT RESULTING FROM THE INPUT STRING GIVEN, ASSUMING THAT A MEANS INSERT AT END X, B INSERT AT Y, C DELETE AT X, D DELETE AT Y, (4 PTS)

C) CAN THE INPUT ITEMS, DISREGARDING THE LETTERS BUT RETAINING THE ORDER OF ITEMS, BE PERMUTTED TO 7245683? (5 PTS)

D) IF LINES 14, 19-23 ARE REMOVED, WHAT WOULD THE DEVICE BE CALLED? (2.5 PTS)

E & F) WHEN THEY ARE REMOVED, WHAT ARE THE ANSWERS TO 1B AND 1C ABOVE? (4 PTS AND 5 PTS)

10) GIVEN N=7 AND BASED(I)=TOP(I)=L -C FOR U<=I<=N AND BASE(N+1)=L -C, C=2 AND L -L /C IS 1C0. WHAT WILL HAPPEN GIVEN THE FOLLOWING ACTIONS TAKEN SEQUENTIALLY? INDICATE A NORMAL INSEPTION OR DELETION WITH THE LETTER I OR D, OVERFLOW AND UNDERFLOW WITH O OR U. ASSUME REPACKING WILL TAKE PLACE WHEN CALLED FOR. (4 PTS)

I3__

D3__

I4__

I8__

D8__

D4__

I7__

D1__

I1__

I3__

11) DESCRIBE PARTIAL ORDERING (IN OTHER WORDS, WHAT IS A PARTIALLY ORDERED SET?) (4 PTS)

12) GIVEN TWO DISJOINT CIRCULAR LISTS, L1 AND L2 POINTED TO BY PTR1 AND PTR2 RESPECTIVELY, SHOW THE INSERTION OF L2 AT THE RIGHT OF L1. ISE L1=A,B,C,D AND L2=E,F,G,H. BE SURE TO SHOW BOTH THE BEFORE AND AFTER DIAGRAMS, INCLUDING THE POINTERS. (10 PTS)

13) WHY IS THE SKY BLUE?

ANS: BECAUSE IT IS NOT RED

141 WHO IS THE PRESIDENT OF THE U.S?

ANS: RICHARD NIXON

APPENDIX A.3

This is the output obtained by invoking the cataloged procedure QUIZGENP. It shows the exam printed and the exam with solutions.

```
FILE TO BE PRINTED NOT SPECIFIED.QUESFIL ASSUMED
```

286_600 DISCRETE STRUCTURES R.K.BREWER

EXAM# DATE:

1) GIVEN STORAGE BLOCKS OF 3000,2000, AND 1000 UNITS, RESPECTIVELY, WOULD THE BEST FIT OR FIRST FIT STRATEGY ALLOW THE MAXIMUM ALLOCATIONS GIVEN REQUESTS OF 1000, 2000, AND 3000 UNITS, RESPECTIVELY? WHICH IS USUALLY THE BEST STRATEGY ACCORDING TO KNUTH?

A)____________________________(1.5 PTS)

B)____________________________(1 PTS)

2) WHICH OF THE FOLLOWING DO NOT MATCH?

SUN:HEAT
MOON:LUNATIC
GIRL:EMOTION
PLACE:SENTIMENT
MIND:HAYSTACK

3) GIVEN THE FOLLOWING CODE AND DATA,

```
DEVICE:PROC OPTIONS (MAIN);
DCL ARRAY (101) CHAR (1);
DCL (N,OP) CHAR (1);
J,K=51;
A:GET EDIT (OP) (A(1));
IF OP=' ' THEN GOTO L99;
IF OP>'B' THEN
   DO;
   PUT LIST (ARRAY(K));
   K=K-1;
   GOTO A;
   END;
GET EDIT (N) (A(1));
IF OP<'B' THEN
   DO;
   K=K+1;
   ARRAY(K)=N;
   END;
ELSE
   DO;
   ARRAY(J)=N;
   J=J-1;
   END;
GOTO A;
L99:
END DEVICE;
A1B2CDA3A4B5B6CB7A8CDCCC
```

A) GIVE THE FORMAL NAME OF THE DEVICE IT SIMULATES, (2.5 PTS.)

B) GIVE THE OUTPUT RESULTING FROM THE INPUT STRING GIVEN, ASSUMING THAT A MEANS INSERT AT END X, B INSERT AT Y, C DELETE AT X, D DELETE AT Y, (4 PTS)

C) CAN THE INPUT ITEMS, DISREGARDING THE LETTERS BUT RETAINING THE ORDER OF ITEMS, BE PERMUTTED TO 7245683? (5 PTS)

D) IF LINES 14, 19-23 ARE REMOVED, WHAT WOULD THE DEVICE BE CALLED? (2.5 PTS)

E & F) WHEN THEY ARE REMOVED, WHAT ARE THE ANSWERS TO 1B AND 1C ABOVE? (4 PTS AND 5 PTS)

4) GIVEN TWO DISJOINT CIRCULAR LISTS, L1 AND L2 POINTED TO BY PTR1 AND PTR2 RESPECTIVELY, SHOW THE INSERTION OF L2 AT THE RIGHT OF L1. USE L1=A,B,C,D AND L2=E,F,G,H. BE SURE TO SHOW BOTH THE BEFORE AND AFTER DIAGRAMS, INCLUDING THE POINTERS . (10 PTS)

5) WHY IS THE SKY BLUE?

1) GIVEN STORAGE BLOCKS OF 3000,2000, AND 1000 UNITS, RESPECTIVELY, WOULD THE BEST FIT OR FIRST FIT STRATEGY ALLOW THE MAXIMUM ALLOCATIONS GIVEN REQUESTS OF 1000, 2000, AND 3000 UNITS, RESPECTIVELY? WHICH IS USUALLY THE BEST STRATEGY ACCORDING TO KNUTH?

A)____________________________(1.5 PTS)

B)____________________________(1 PTS)

2) WHICH OF THE FOLLOWING DO NOT MATCH?

SUN:HEAT
MOON:LUNATIC
GIRL:EMOTION
PLACE:SENTIMENT
MIND:HAYSTACK

3) GIVEN THE FOLLOWING CODE AND DATA,

```
DEVICE:PROC OPTIONS (MAIN);
DCL ARRAY (101) CHAR (1);
DCL (N,OP) CHAR (1);
J,K=51;
A:GET EDIT (OP) (A(1));
IF OP=' ' THEN GOTO L99;
IF OP>'B' THEN
   DO;
   PUT LIST (ARRAY(K));
   K=K-1;
   GOTO A;
   END;
GET EDIT (N) (A(1));
IF OP<'B' THEN
   DO;
   K=K+1;
   ARRAY(K)=N;
   END;
ELSE
   DO;
   ARRAY(J)=N;
   J=J-1;
   END;
GOTO A;
L99:
END DEVICE;
A1B2CDA3A4B5B6CB7A8CDCCC
```

A) GIVE THE FORMAL NAME OF THE DEVICE IT SIMULATES, (2.5 PTS.)

B) GIVE THE OUTPUT RESULTING FROM THE INPUT STRING GIVEN, ASSUMING THAT A MEANS INSERT AT END X, B INSERT AT Y, C DELETE AT X, D DELETE AT Y, (4 PTS)

C) CAN THE INPUT ITEMS, DISREGARDING THE LETTERS BUT RETAINING THE ORDER OF ITEMS, BE PERMUTTED TO 7245683? (5 PTS)

D) IF LINES 14, 19-23 ARE REMOVED, WHAT WOULD THE DEVICE BE CALLED? (2.5 PTS)

E & F) WHEN THEY ARE REMOVED, WHAT ARE THE ANSWERS TO 1B AND 1C ABOVE? (4 PTS AND 5 PTS)

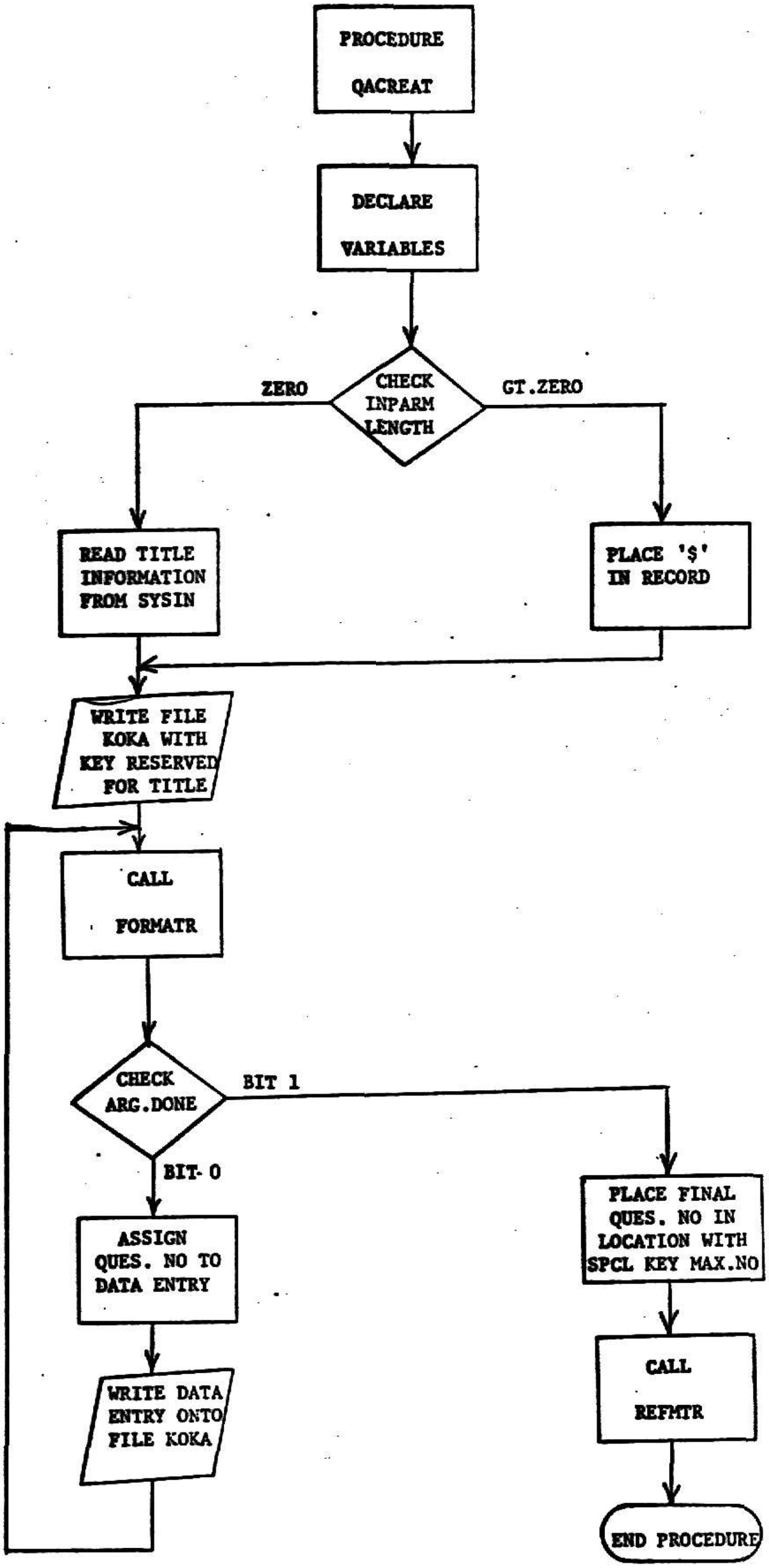
4) GIVEN TWO DISJOINT CIRCULAR LISTS, L1 AND L2 POINTED TO BY PTR1 AND PTR2 RESPECTIVELY, SHOW THE INSERTION OF L2 AT THE RIGHT OF L1. ISE L1=A,B,C,D AND L2=E,F,G,H. BE SURE TO SHOW BOTH THE BEFORE AND AFTER DIAGRAMS, INCLUDING THE POINTERS. (10 PTS)

5) WHY IS THE SKY BLUE?

ANS: BECAUSE IT IS NOT RED

APPENDIX B.1

PROGRAM QACREAT

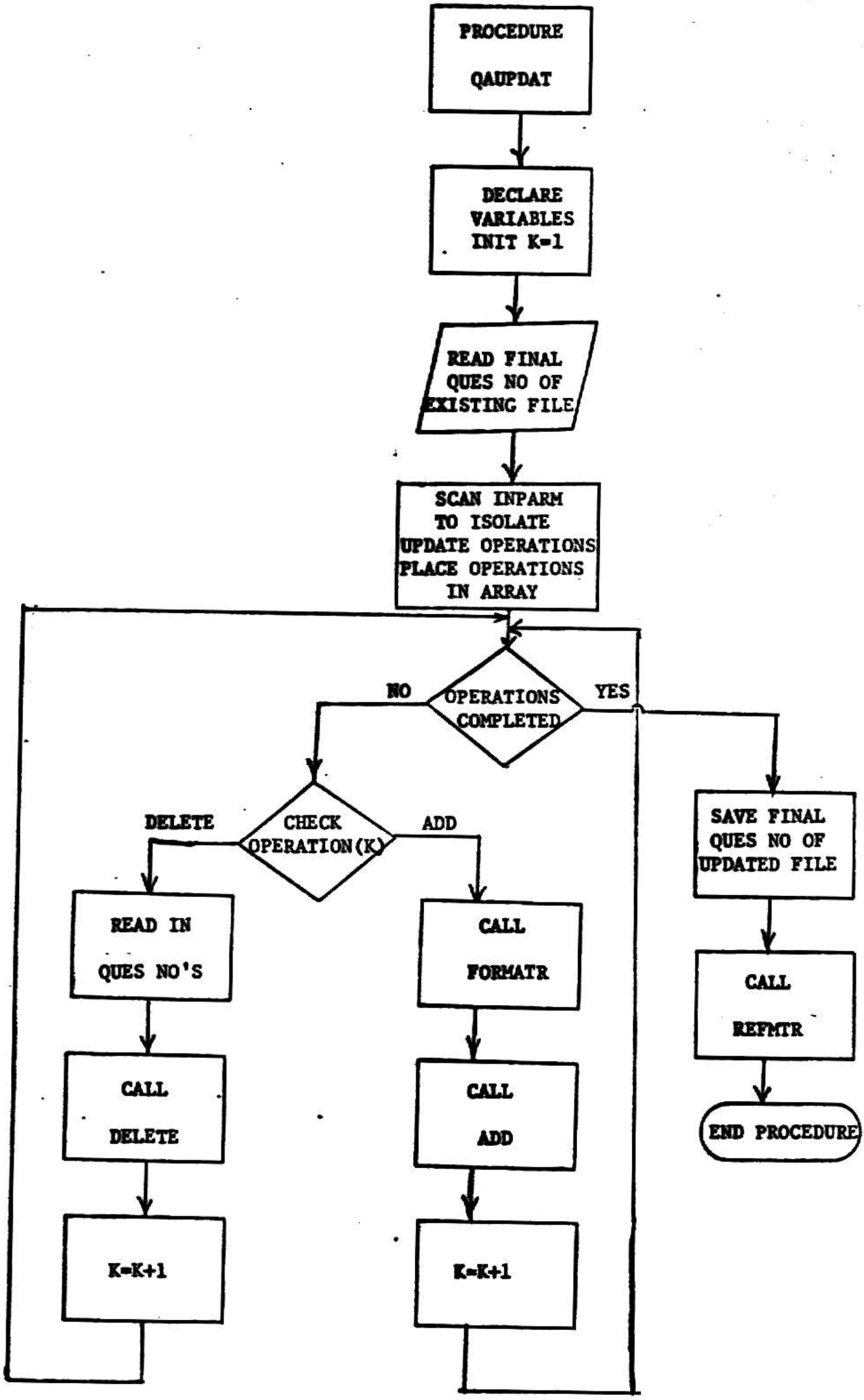

PROCEDURE QACREAT
DECLARE VARIABLES
CHECK INPARM LENGTH
ZERO
GT.ZERO
READ TITLE INFORMATION FROM SYSIN
PLACE '$' IN RECORD
WRITE FILE KOKA WITH KEY RESERVED FOR TITLE
CALL FORMATR
CHECK ARG.DONE
BIT 1
BIT- 0
ASSIGN QUES. NO TO DATA ENTRY
WRITE DATA ENTRY ONTO FILE KOKA
PLACE FINAL QUES. NO IN LOCATION WITH SPCL KEY MAX.NO
CALL REFMTR
END PROCEDURE

```
                 KSU'S PL/I NEATENER AND PRECOMPILER                PAGE       1

(SUBSCRIPTRANGE,STRINGRANGE):DACREAT:                                  1    1
      PROC(INPARM)OPTIONS(MAIN);                                       1    1
/*                                                               */    1    2
/*****************************************************************/    1    2
/* THIS PROGRAM IS DESIGNED TO ACCEPT DATA ENTRIES WHICH CONSISTS */    1    2
/* OF SETS OF QUESTIONS AND ANSWERS,AND TO PROCESS THEM ACCORDING */    1    2
/* TO THE FORMAT AND SPACE CODES. THE PROCESSED INFORMATION IS    */    1    2
/* THEN STORED IN A REGIONAL(3) DIRECT DATASET. THE PROGRAM ALSO  */    1    2
/* PROVIDES THE USER WITH A FORMATTED LISTING OF THE DATASET      */    1    2
/* CREATED                                                        */    1    2
/*                                                               */    1    2
/* AN OPTION 'NO TITLE' CAN BE PASSED TO THIS PROGRAM THROUGH     */    1    2
/* THE PARM PARAMETER AND THIS TELLS THE PROGRAM THAT NO TITLE    */    1    2
/* INFORMATION IS BEING PROVIDED BY THE USER. IF THIS OPTION IS NOT*/   1    2
/* TURNED ON,THEN THE TITLE INFORMATION IS READ IN FIRST AND STORED*/   1    2
/* IN A SPECIAL LOCATION WHICH IS DENOTED BY THE KEY 'INF0000000'.*/    1    2
/* OTHERWISE THE CHARACTER '$' IS PLACED IN THAT LOCATION,        */    1    2
/* INDICATING NO TITLE INFORMATION                                */    1    2
/*                                                               */    1    2
/* THE PROCESSING OF THE DATA ENTRIES IS DONE BY A SUBROUTINE     */    1    2
/* FORMATR WHICH RETURNS THE PROCESSED INFORMATION THROUGH THE    */    1    2
/* ARGUEMENT DATA. ARGUEMENT DONE WHEN TURNED ON INDICATES END OF */    1    2
/* FILE                                                           */    1    2
/*                                                               */    1    2
/* A QUESTION NUMBER IS ASSIGEND TO EACH DATA ENTRY AND A REGION  */    1    2
/* CORRESPONDING TO THAT DETERMINED. THE QUESTION NUMBER          */    1    2
/* CONCATENATED WITH THE REGION FORMS THE KEY FOR THE RECORD      */    1    2
/* TO BE WRITTEN OUT ON DISK                                      */    1    2
/*                                                               */    1    2
/* SUBROUTINE REFMTR IS CALLED WITH MAST TURNED ON AND THE OTHER  */    1    2
/* ARGUEMENTS AS DUMMIES. THIS IS BECAUSE ONLY THE MASTERFILE PRINT*/   1    2
/* OUT IS REQUIRED AND NOT THE OTHER PRINT OUTS.                  */    1    2
/*                                                               */    1    2
/*         DEFINITION OF SOME VARIABLES                           */    1    2
/*                                                               */    1    2
/* INT_QUES:_SINCE QUES_NO AND REGION ARE CHARACTER VARIABLES,    */    1    2
/*           INT_QUES IS USED FOR INCREMENTING QUES_NO EACH TIME  */    1    2
/*           A DATA ENTRY IS READ IN AND ALSO TO DTERMINE THE     */    1    2
/*           REGION                                               */    1    2
/* OVRLY1,OVRLY2:- SINCE RECORD I/O DOES NOT PERMIT A VARYING     */    1    2
/*               STRING IN A STRUCTURE,OVRLY1 IS OVERLAID ON DATA */    1    2
/*               AND THE NECESSARY INFORMATION MOVED TO THE STRING */   1    2
/*              OVRLY2. OVRLY2 IS WRITTEN ON TO THE REGIONAL(3)   */    1    2
/*             FILE,KOKA.                                         */    1    2
/* KEYSAV:- THE HIGHEST QUESTION NUMBER IS SAVED IN A SPECIAL     */    1    2
/*          LOCATION WITH THE KEY '0000000000' SO THAT WHILE      */    1    2
/*          UPDATING, IT CAN BE RETRIEVED AND USED FOR GENERATING */    1    2
/*          QUESTION NUMBERS IN SEQUENCE                          */    1    2
/* KOKA:- DIRECT KEYED REGIONAL(3) FILE.                          */    1    2
/*****************************************************************/    1    2
/*                                                               */    1    2
      DCL (KOKA) FILE RECORD KEYED ENV(REGIONAL(3));                   1    2
      DCL 1 DATA,                                                      1    3
            2 QUES_NO CHAR(3) INIT('000'),                             1    3
            2 FORMAT_CODE CHAR(1),                                     1    3
            2 SPACE_CODE BIN FIXED(15),                                1    3
            2 QUESTION CHAR(4500) VAR;                                 1    3
      DCL INTER FIXED DEC(5),REGION CHAR(8);                           1    4
```

```
                    KSU'S PL/I NEATENER AND PRECOMPILER                     PAGE        2
              DCL OVRLY1 CHAR(4506) BASED(P),OVRLY2 CHAR(4506) VAR;          1      5
              DCL INT_QUES FIXED DEC(5),KEYSAV CHAR(3);                      1      6
              DCL INPARM CHAR(100) VAR,                                      1      7
              1 GENL_INFO,                                                   1      7
               2 COURSE_NO CHAR(7),                                          1      7
               2 COURSE_NAME CHAR(100) VAR,                                  1      7
               2 INSTRUCTOR CHAR(25) VAR,                                    1      7
              INFO CHAR(3) INIT('INF'),MAX_NO CHAR(3) INIT('000');           1      7
              DCL DONE BIT(1) INIT('0'B);                                    1      8
              DCL DUMMY1 CHAR(3),DUMMY2 BIN FIXED INIT(0),                   1      9
              MAST BIT(1) INIT('1'B);                                        1      9
              OPEN FILE(KOKA)DIRECT OUTPUT RECORD;                           1     10
              PUT EDIT('MESSAGES')(COL(50),A);                               1     11
              PUT SKIP(4);                                                   1     12
              P=ADDR(DATA);                                                  1     13
                 IF LENGTH(INPARM)¬=0 THEN                                   2     14
                    DO;                                                      3     15
                    OVRLY2='$';                                              3     16
                    REGION='        0';                                      3     17
                    WRITE FILE(KOKA)FROM(OVRLY2)KEYFROM(INFO||REGION);       3     18
                    GOTO READ1;                                              3     19
                    END;                                                     3     20
              GET LIST(COURSE_NO,COURSE_NAME,INSTRUCTOR);                    1     21
              OVRLY2=COURSE_NO||COURSE_NAME||'@'||INSTRUCTOR;                1     22
              REGION='        0';                                            1     23
              WRITE FILE(KOKA)FROM(OVRLY2)KEYFROM(INFO||REGION);             1     24
     READ1:   CALL FORMATR(DATA,DONE);                                       1     25
                 IF DONE THEN                                                2     26
                 GOTO FINISH;                                                2     27
              GET STRING(QUES_NO)EDIT(INT_QUES)(F(3));                       1     28
              INT_QUES=INT_QUES+1;                                           1     29
              PUT STRING(QUES_NO)EDIT(INT_QUES)(F(3));                       1     30
              INTER=INT_QUES/10;                                             1     31
              REGION=INTER;                                                  1     32
              L=LENGTH(QUESTION);                                            1     33
              OVRLY2=SUBSTR(OVRLY1,1,L+6);                                   1     34
              WRITE FILE(KOKA)FROM(OVRLY2)KEYFROM(QUES_NO||REGION);          1     35
              GOTO READ1;                                                    1     36
     FINISH:  REGION='        0';                                            1     37
              KEYSAV=QUES_NO;                                                1     38
              WRITE FILE(KOKA)FROM(KEYSAV)KEYFROM(MAX_NO||REGION);           1     39
              CALL REFMTR(DUMMY1,DUMMY2,MAST);                               1     40
              END QACREAT;                                                   1     41
```

APPENDIX B.2

PROGRAM QAUPDAT


PROCEDURE
QAUPDAT
DECLARE
VARIABLES
INIT K=1
READ FINAL
QUES NO OF
EXISTING FILE
SCAN INPARM
TO ISOLATE
UPDATE OPERATIONS
PLACE OPERATIONS
IN ARRAY
NO
OPERATIONS
COMPLETED
YES
DELETE
CHECK
OPERATION(K)
ADD
SAVE FINAL
QUES NO OF
UPDATED FILE
READ IN
QUES NO'S
CALL
FORMATR
CALL
REFMTR
CALL
DELETE
CALL
ADD
END PROCEDURE
K=K+1
K=K+1

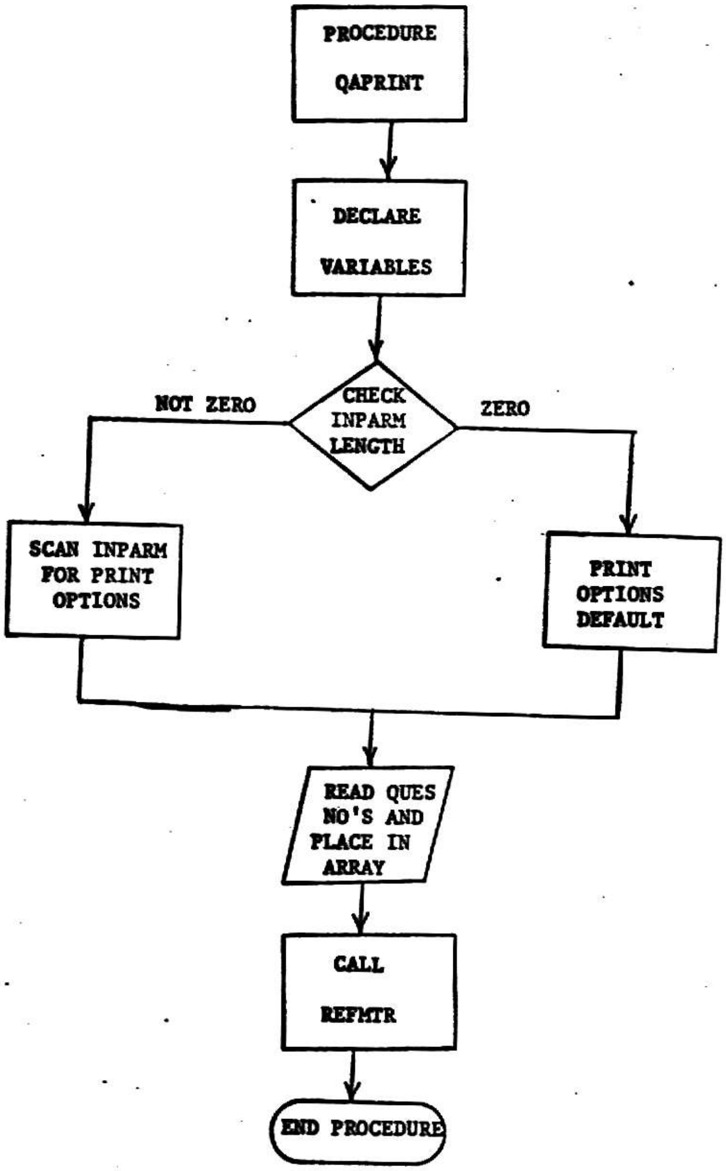
```
KSU'S PL/I NEATENER AND PRECOMPILER                    PAGE        3

(SUBSCRIPTRANGE,STRINGRANGE):QAUPDAT:                                      1    1
        PROC(INPARM)OPTIONS(MAIN);                                         1    1
/*                                                                 */      1    2
/*****************************************************************/       1    2
/* THIS PROGRAM IS DESIGNED TO PERFORM UPDATE OPERATIONS AS        */      1    2
/* SPECIFIED IN THE INPARM PARAMETER. THE OPERATIONS TO BE PERFOR  */      1    2
/* MED ARE ISOLATED FROM INPARM AND STORED IN AN OPERATION FIELD   */      1    2
/* AND THE NUMBER OF TIMES EACH OPERATION IS TO BE PERFORMED IN A  */      1    2
/* COUNT FIELD.                                                    */      1    2
/* THE ADDITION OF A DATA ENTRY IS DONE BY INVOKING SUBROUTINES    */      1    2
/* FORMATR AND ADD. FORMATR FIRST PROCESSES THE INFORMATION ACCOR  */      1    2
/* DING TO THE FORMAT AND SPACE CODES AND ADD WRITES IT OUT ON TO  */      1    2
/* THE FILE AFTER ASSIGNING A QUESTION NUMBER AND DETERMINING A    */      1    2
/* REGION                                                          */      1    2
/*                                                                 */      1    2
/* DELETION IS DONE BY INVOKING SUBROUTINE DELETE,PASSING THE ARR  */      1    2
/* AY OF QUESTION NUMBERS, THE QUESTIONS CORRESPONDING TO WHICH    */      1    2
/* ARE TO BE DELETED,AS AN ARGUEMENT.                              */      1    2
/*                                                                 */      1    2
/* SUBROUTINE REFMTR IS INOKED WITH MAST TURNED ON AND THE OTHER   */      1    2
/* ARGUEMENTS AS DUMMIES ,TO OBTAIN A PRINT OUT OF THE UPDATED     */      1    2
/* MASTERFILE                                                      */      1    2
/*                                                                 */      1    2
/*         DEFINITION OF SOME VARIABLES                            */      1    2
/*                                                                 */      1    2
/* OPERATION: IS AN ARRAY OF CHARACTER STRINGS AND CONTAINS THE    */      1    2
/*            OPERATIONS SPECIFIED IN INPARM.                      */      1    2
/* NO: IS AN ARRAY CONTAINING THE NUMBER OF TIMES THE OPERATIONS   */      1    2
/*     HAVE TO BE PERFORMED. THIS IS IN CORRESPONDENCE WITH ARRAY  */      1    2
/*     OPERATION                                                   */      1    2
/* QUES_ARRAY: CONTAINS THE ARRAY OF QUESTION NUMBERS AND IS THE   */      1    2
/*            ARGUEMENT PASSED TO DELETE.                          */      1    2
/*****************************************************************/       1    2
/*                                                                 */      1    2
        DCL INPARM CHAR(100) VAR,IJ BIN FIXED INIT(0),                     1    2
        K BIN FIXED INIT(1),OPERATION(5) CHAR(6) VAR,NO(5) BIN FIXED,      1    2
        COUNT FIXED BIN(15),QUES_ARRAY(25) CHAR(3),CARD CHAR(80) VAR;      1    2
        DCL 1 DATA,                                                        1    3
          2 QUES_NO CHAR(3),                                               1    3
          2 FORMAT_CODE CHAR(1),                                           1    3
          2 SPACE_CODE BIN FIXED,                                          1    3
          2 QUESTION CHAR(4500) VAR;                                       1    3
        DCL ADD ENTRY(POINTER,CHAR(3),BIN FIXED);                          1    4
        DCL DELETE ENTRY ((*) CHAR(3),FIXED BIN(15),CHAR(3));              1    5
        DCL MAX_NO CHAR(3) INIT('000'),                                    1    6
        REGION CHAR(8) INIT('       0');                                   1    6
        DCL MAX_KEY CHAR(3),KEYSAV CHAR(3);                                1    7
        DCL P POINTER,EQ BIN FIXED INIT(0);                                1    8
        DCL KOKA FILE RECORD KEYED  ENV(REGIONAL(3)),                      1    9
            L BIN FIXED,DONE BIT(1),Q_COUNT FIXED DEC(5);                  1    9
        DCL DUMMY1(1) CHAR(3),DUMMY2 BIN FIXED INIT(0),                    1   10
        MAST BIT(1) INIT('1'B);                                            1   10
        ON ENDFILE(SYSIN)                                                  1   11
        GOTO FINISH;                                                       1   12
        OPEN FILE(KOKA)DIRECT UPDATE;                                      1   13
        PUT EDIT('MESSAGES')(COL(50),A(8));                                1   14
        PUT SKIP(4);                                                       1   15
        P=ADDR(DATA);                                                      1   16
```

```
KSU'S PL/I NEATENER AND PRECOMPILER                   PAGE         4
        READ FILE(KOKA)INTO(KEYSAV)KEY(MAX_NO||REGION);                    1
        MAX_KEY=KEYSAV;                                                    1
        LEN=LENGTH(INPARM);                                                1
           IF LEN=0 THEN                                                   2
              DO;                                                          3
              PUT EDIT('ERROR:UPDATE OPERATION NOT SPECIFIED,PROGRAM TE    3
              RMINATED')(SKIP,COL(10),A);                                  3
              GOTO OUT;                                                    3
              END;                                                         3
/*                                                                */     1
/*****************************************************************/     1
/* THIS SECTION OF THE PROGRAM SCANS INPARM TO ISOLATE THE UPDATE  */     1
/* OPERATIONS SPECIFIED BY THE USER. OPERATIONS ARE PLACED IN AN   */     1
/* ARRAY FIELD                                                     */     1
/*****************************************************************/     1
/*                                                                */     1
COMMA:  J=INDEX(INPARM,',');                                               1
        IJ=IJ+1;                                                           1
           IF J=0 THEN                                                     2
              DO;                                                          3
              EQ=INDEX(INPARM,'=');                                        3
                 IF EQ=0 THEN                                              4
                    DO;                                                    5
                    NO(IJ)=1;                                              5
                    OPERATION(IJ)=INPARM;                                  5
                    GOTO CHECK;                                            5
                    END;                                                   5
              OPERATION(IJ)=SUBSTR(INPARM,1,EQ-1);                         3
              NO(IJ)=SUBSTR(INPARM,EQ+1,LEN-EQ);                           3
              GOTO CHECK;                                                  3
              END;                                                         3
        EQ=INDEX(SUBSTR(INPARM,1,J-1),'=');                                1
           IF EQ=0 THEN                                                    2
              DO;                                                          3
              NO(IJ)=1;                                                    3
              OPERATION(IJ)=SUBSTR(INPARM,1,J-1);                          3
              GOTO CONT;                                                   3
              END;                                                         3
        NO(IJ)=SUBSTR(INPARM,EQ+1,J-(EQ+1));                               1
        OPERATION(IJ)=SUBSTR(INPARM,1,EQ-1);                               1
CONT:   INPARM=SUBSTR(INPARM,J+1);                                         1
        LEN=LENGTH(INPARM);                                                1
        GOTO COMMA;                                                        1
CHECK:     IF K>IJ THEN                                                    2
           GOTO FINISH;                                                    2
           IF OPERATION(K)='ADD' THEN                                      2
           GOTO PROCESS1;                                                  2
        QUES_ARRAY=' ';                                                    1
        I=0;                                                               1
/*                                                                */     1
/*****************************************************************/     1
/*          DELETING DATA ENTRIES                                  */     1
/* THIS SECTION OF THE PROGRAM INVOKES THE SUBROUTINE DELETE       */     1
/* PASSING QUES_ARRAY,AND THE SIZE OF THIS ARRAY.                  */     1
/*****************************************************************/     1
/*                                                                */     1
PROCESS2:  IF I=NO(K)THEN                                                  2
           GOTO INVOK;                                                     2
```

```
KSU'S PL/I NEATENER AND PRECOMPILER                    PAGE         5
          GET EDIT(CARD)(COL(1),A(80));                                 1   60
LOOP:     IK=VERIFY(CARD,' ');                                          1   61
             IF IK=0 THEN                                               2   62
                DO;                                                     3   63
INVOK:          CALL DELETE(QUES_ARRAY,NO(K),MAX_KEY);                  3   64
                K=K+1;                                                  3   65
                GOTO CHECK;                                             3   66
                END;                                                    3   67
          I=I+1;                                                        1   68
          CARD=SUBSTR(CARD,IK);                                         1   69
          BLANK=INDEX(CARD,' ');                                        1   70
             IF BLANK=0 THEN                                            2   71
                DO;                                                     3   72
                LOC=4-LENGTH(CARD);                                     3   73
                SUBSTR(QUES_ARRAY(I),LOC)=SUBSTR(CARD,1);               3   74
                GOTO PROCESS2;                                          3   75
                END;                                                    3   76
          LOC=4-(BLANK-1);                                              1   77
          SUBSTR(QUES_ARRAY(I),LOC)=SUBSTR(CARD,1,BLANK-1);             1   78
          CARD=SUBSTR(CARD,BLANK+1);                                    1   79
          GOTO LOOP;                                                    1   80
/*                                                                 */   1   81
/*****************************************************************/   1   81
/*               ADDING DATA ENTRIES                               */   1   81
/* THIS SECTION OF THE PROGRAM INVOKES SUBROUTINES FORMATR AND     */   1   81
/* ADD WHICH PERFORM THE NECCESSARY PROCESSING AND THEN ADD THE    */   1   81
/* RECORD                                                          */   1   81
/*****************************************************************/   1   81
/*                                                                 */   1   81
PROCESS1:                                                               1   81
          COUNT=1;                                                      1   81
          Q_COUNT=0;                                                    1   82
             DO WHILE(COUNT<=NO(K));                                    2   83
             PUT STRING(QUES_NO)EDIT(Q_COUNT)(F(3));                    2   84
             COUNT=COUNT+1;                                             2   85
             CALL FORMATR(DATA,DONE);                                   2   86
             Q_COUNT=Q_COUNT+1;                                         2   87
             L=LENGTH(QUESTION);                                        2   88
             CALL ADD(P,MAX_KEY,L);                                     2   89
             END;                                                       2   90
          K=K+1;                                                        1   91
          GOTO CHECK;                                                   1   92
FINISH:   KEYSAV=MAX_KEY;                                               1   93
          REGION='       0';                                            1   94
          REWRITE FILE(KOKA)FROM(KEYSAV)KEY(MAX_NO||REGION);            1   95
          CALL REFMTR(DUMMY1,DUMMY2,MAST);                              1   96
OUT:      END QAUPDAT;                                                  1   97
```

APPENDIX B.3

PROGRAM QAPRINT

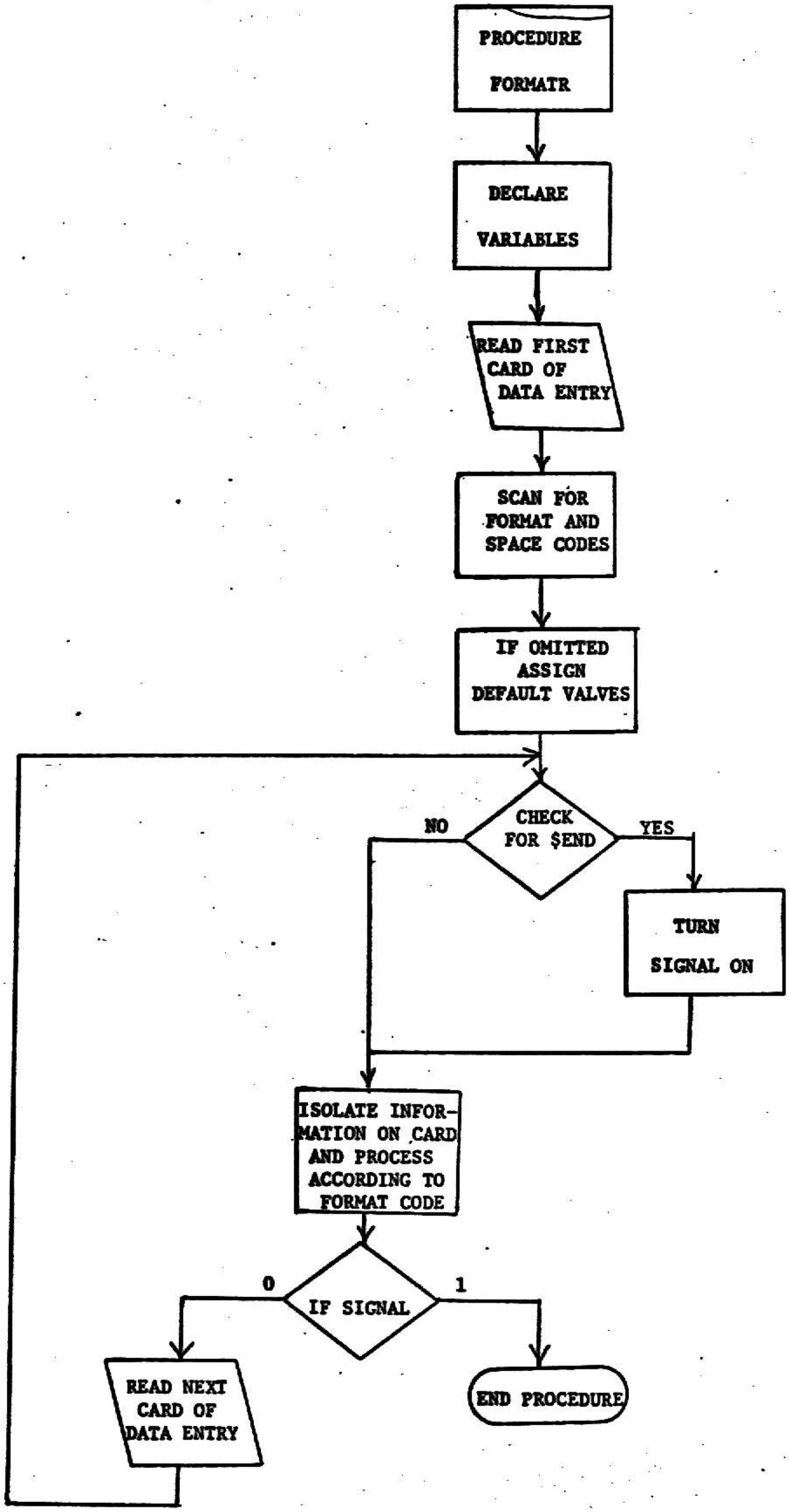

PROCEDURE
QAPRINT
DECLARE
VARIABLES
CHECK
INPARM
LENGTH
NOT ZERO
ZERO
SCAN INPARM
FOR PRINT
OPTIONS
PRINT
OPTIONS
DEFAULT
READ QUES
NO'S AND
PLACE IN
ARRAY
CALL
REFMTR
END PROCEDURE

```
KSU'S PL/I NEATENER AND PRECOMPILER                  PAGE        6

(SUBSCRIPTRANGE,STRINGRANGE):QAPRINT:                                     1     1
        PROC(INPARM)OPTIONS(MAIN);                                        1     1
/*                                                                 */     1     2
/*******************************************************************/     1     2
/* THIS PROGRAM IS DESIGNED TO ACCEPT THE QUESTION NUMBERS         */     1     2
/* SPECIFIED BY THE USER AND TO STORE THEM IN AN ARRAY. THE        */     1     2
/* PROGRAM THEN INVOKES THE SUBROUTINE REFMTR PASSING THE ARRAY    */     1     2
/* OF QUESTION NUMBERS AS AN ARGUEMENT,AND THE SUBROUTINE REFMTR   */     1     2
/* GENERATES AN EXAM,AN EXAM WITH SOLUTIONS AND A PRINTOUT OF THE  */     1     2
/* MASTER FILE,IF REQUESTED.                                       */     1     2
/*                                                                 */     1     2
/* THE PRINT OPTIONS CAN BE PASSED TO THE INPARM PARAMETER IN THE  */     1     2
/* PROC OPTIONS (MAIN) STATEMENT THROUGH THE EXEC CARD INVOKING    */     1     2
/* QAPRINT. REFER TO THE USER'S MANUAL FOR A DETAILED EXPLANATION  */     1     2
/* OF THE PRINT OPTIONS.                                           */     1     2
/*                                                                 */     1     2
/* MAST IS SET TO BIT(1) IF MASTFIL IS SPECIFIED IN INPARM.        */     1     2
/* NOTE: A PRINTOUT OF ONLY THE MASTERFILE CAN BE OBTAINED BY NOT  */     1     2
/* SPECIFYING ANY QUESTION NUMBERS AND BY PASSING MASTFIL IN        */     1     2
/* INPARM                                                          */     1     2
/*******************************************************************/     1     2
/*                                                                 */     1     2
        DCL INPARM CHAR(100) VAR,RET_KEY(25) CHAR(3) INIT('   '),          1     2
        MAST BIT(1) INIT('0'B),(J,LOC) BIN FIXED,                         1     2
        (I,K) BIN FIXED INIT(0),CARD CHAR(80) VAR;                        1     2
        ON ENDFILE(SYSIN)                                                 1     3
        GOTO INVOK;                                                       1     4
           IF LENGTH(INPARM)=0 THEN                                       2     5
              DO;                                                         3     6
              PUT EDIT('FILE TO BE PRINTED NOT SPECIFIED.QUESFIL ASSUME   3     7
              D')(SKIP,COL(5),A);                                         3     7
              GOTO KEYS;                                                  3     8
              END;                                                        3     9
        J=INDEX(INPARM,',');                                              1    10
           IF J=0 THEN                                                    2    11
              DO;                                                         3    12
              MAST=('MASTFIL'=INPARM);                                    3    13
                 IF MAST THEN                                             4    14
                 GOTO INVOK;                                              4    15
              GOTO KEYS;                                                  3    16
              END;                                                        3    17
        MAST='1'B;                                                        1    18
KEYS:   RET_KEY=' ';                                                      1    19
RLOOP:  GET EDIT(CARD)(A(80));                                            1    20
LOOP:   K=VERIFY(CARD,' ');                                               1    21
           IF K=0 THEN                                                    2    22
           GOTO RLOOP;                                                    2    23
        I=I+1;                                                            1    24
        CARD=SUBSTR(CARD,K);                                              1    25
        J=INDEX(CARD,' ');                                                1    26
           IF J=0 THEN                                                    2    27
              DO;                                                         3    28
              LOC=4-LENGTH(CARD);                                         3    29
              SUBSTR(RET_KEY(I),LOC)=SUBSTR(CARD,1);                      3    30
              GOTO RLOOP;                                                 3    31
              END;                                                        3    32
        LOC=4-(J-1);                                                      1    33
        SUBSTR(RET_KEY(I),LOC)=SUBSTR(CARD,1,J-1);                        1    34
```

```
KSU'S PL/I NEATENER AND PRECOMPILER                  PAGE        7

         CARD=SUBSTR(CARD,J+1);                                                 1   35
         GOTO LOOP;                                                             1   36
INVOK:   CALL REFMTR(RET_KEY,I,MAST);                                           1   37
         END QAPRINT;                                                           1   38
```

APPENDIX B.4

FORMATR


PROCEDURE FORMATR
DECLARE VARIABLES
READ FIRST CARD OF DATA ENTRY
SCAN FOR FORMAT AND SPACE CODES
IF OMITTED ASSIGN DEFAULT VALVES
CHECK FOR $END
NO
YES
TURN SIGNAL ON
ISOLATE INFORMATION ON CARD AND PROCESS ACCORDING TO FORMAT CODE
IF SIGNAL
0
1
READ NEXT CARD OF DATA ENTRY
END PROCEDURE

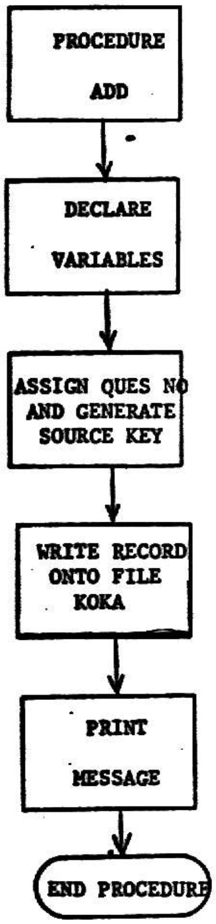
```
KSU'S PL/I NEATENER AND PRECOMPILER                    PAGE        1

/*                                                                  */    1    1
/********************************************************************/    1    1
/* SUBROUTINE FORMATR IS DESIGNED TO READ IN DATA ENTRIES WHICH     */    1    1
/* ARE IN THE FORM OF A QUESTION AND ANSWER(OPTIONAL) AND THEN      */    1    1
/* TO PROCESS IT ACCORDING TO THE FORMAT AND SPACE CODES. THE       */    1    1
/* PROCESSED INFORMATION IS THEN RETURNED THROUGH THE STRUCTURE     */    1    1
/* DATA WHICH IS PASSED AS AN ARGUEMENT. ARGUEMENT A IS BIT(1)      */    1    1
/* AND IS TURNED ON WHEN ALL THE DATA HAS BEEN READ IN.             */    1    1
/********************************************************************/    1    1
/*                                                                  */    1    1
(SUBSCRIPTRANGE,STRINGRANGE):FORMATR:                                     1    1
        PROC(DATA,A);                                                     1    1
        DCL 1 DATA,                                                       1    2
         2 QUES_NO CHAR(3),                                               1    2
         2 F_CODE CHAR(1),                                                1    2
         2 S_CODE BIN FIXED,                                              1    2
         2 QUES CHAR(*) VAR;                                              1    2
        DCL INT_QUES FIXED DEC(5),A BIT(1);                               1    3
        DCL CARD CHAR(80) VAR,SAVCARD CHAR(80) INIT(' '),SIGN BIT(1);     1    4
        DCL (BEGIN,CUM,J,I) BIN FIXED,FLAG BIT(1);                        1    5
        DCL SIGNAL BIT(1) INIT('0'B);                                     1    6
        ON ENDFILE(SYSIN)                                                 1    7
        GOTO FINISH;                                                      1    8
        INT_QUES=QUES_NO+1;                                               1    9
        I=0;                                                              1   10
        J=1;                                                              1   11
        QUES=' ';                                                         1   12
        FLAG='0'B;                                                        1   13
        SIGN='1'B;                                                        1   14
/*                                                                  */    1   15
/* ************************************************************     */    1   15
/* THIS SECTION OF THE PROGRAM CHECKS THE FORMAT AND SPACE CODES    */    1   15
/* BY SCANNING THE FIRST CARD OF THE DATA ENTRY FOR '&&'. IF        */    1   15
/* ABSENT,THEN DEFAULT VALUES ARE ASSIGNED AND APPROPRIATE          */    1   15
/* MESSAGES PRINTED OUT.                                            */    1   15
/* ************************************************************     */    1   15
/*                                                                  */    1   15
LOOP:   GET EDIT(CARD)(COL(1),A(80));                                     1   15
AND:    K=INDEX(CARD,'&&');                                               1   16
           IF K=0 THEN                                                    2   17
              DO;                                                         3   18
                 IF¬SIGN THEN                                             4   19
                    DO;                                                   5   20
                    PUT EDIT('DATA',INT_QUES,'SPACE CODE NOT SPECIFIED.   5   21
                    DEFAULT VALUE ASSUMED')(SKIP(2),COL(5),A(4),F(2),X(   5   21
                    2),A);                                                5   21
                    S_CODE=2;                                             5   22
                    GOTO LOOP1;                                           5   23
                    END;                                                  5   24
              PUT EDIT('DATA',INT_QUES,'SPACE AND FORMAT CODES NOT SPEC   3   25
              IFIED.DEFAULT VALUES ASSUMED')(SKIP(2),COL(5),A(4),F(2),X   3   25
              (2),A);                                                     3   25
              F_CODE='D';                                                 3   26
              S_CODE=2;                                                   3   27
              GOTO LOOP1;                                                 3   28
              END;                                                        3   29
           IF SUBSTR(CARD,K+2,1)='U' THEN                                 2   30
              DO;                                                         3   31
```

```
KSU'S PL/I NEATENER AND PRECOMPILER                     PAGE         2
              FLAG='1'B;                                                  3   32
              F_CODE='U';                                                 3   33
              GOTO CONT;                                                  3   34
              END;                                                        3   35
           IF SIGN THEN                                                   2   36
              DO;                                                         3   37
              PUT EDIT('DATA',INT_QUES,'FORMAT CODE NOT SPECIFIED.DEFAU   3   38
              LTS TO D')(SKIP(2),COL(5),A(4),F(2),X(2),A);                3   38
              F_CODE='D';                                                 3   39
              END;                                                        3   40
        IJ=VERIFY(SUBSTR(CARD,K+2,2),'0123456789');                       1   41
           IF IJ=0 THEN                                                   2   42
              DO;                                                         3   43
              S_CODE=SUBSTR(CARD,K+2,2);                                  3   44
              CARD=SUBSTR(CARD,K+4);                                      3   46
              END;                                                        3   46
           ELSE                                                           2   47
              IF IJ=2 THEN                                                3   47
                 DO;                                                      4   48
                 S_CODE=SUBSTR(CARD,K+2,1);                               4   49
                 CARD=SUBSTR(CARD,K+3);                                   4   50
                 END;                                                     4   51
              ELSE                                                        3   52
                 DO;                                                      4   52
                 PUT EDIT('DATA',INT_QUES,'IMPROPER SPACE CODE SPECIFIE   4   53
                 D.DEFAULT VALUE ASSUMED')(SKIP(2),COL(5),A(4),F(2),X(2   4   53
                 ),A);                                                    4   53
                 S_CODE=2;                                                4   54
                 END;                                                     4   55
        GOTO LOOP1;                                                       1   56
CONT:   CARD=SUBSTR(CARD,K+3);                                            1   57
        SIGN='0'B;                                                        1   58
        GOTO AND;                                                         1   59
/*                                                                  */    1   60
/* ************************************************************** */    1   60
/* THIS SECTION SCANS  FOR THE SPECIAL SYMBOL '$END' WHICH          */    1   60
/* INDICATES THE END OF INFORMATION FOR A DATA ENTRY AND TURNS      */    1   60
/* SIGNAL ON,IF IT FINDS  '$END'.                                   */    1   60
/* ************************************************************** */    1   60
/*                                                                  */    1   60
RLOOP:  GET EDIT(CARD)(COL(1),A(80));                                     1   60
LOOP1:  EOD=INDEX(CARD,'$END');                                           1   61
           IF EOD¬=0 THEN                                                 2   62
              DO;                                                         3   63
                 IF EOD=1 THEN                                            4   64
                 GOTO FINITO;                                             4   65
              SAVCARD=SUBSTR(CARD,1,EOD-1);                               3   66
              CARD=SUBSTR(CARD,1,EOD-1);                                  3   67
              SIGNAL='1'B;                                                3   68
              GOTO LAST;                                                  3   69
              END;                                                        3   70
        SAVCARD=CARD;                                                     1   71
LAST:   CUMJ=0;                                                           1   72
/*                                                                  */    1   73
/* ************************************************************** */    1   73
/* THIS SECTION OF THE PROGRAM PROCESSES INFORMATION ON A DATA      */    1   73
/* CARD ACCORDING TO THE FORMAT CODE. IT SUPPRESSES PRECEEDING      */    1   73
/* AND FOLLOWING BLANKS IF THE CODE IS 'D' AND BUILDS UP A          */    1   73
```

```
KSU'S PL/I NEATENER AND PRECOMPILER                   PAGE         3

/* LOGICAL RECORD  BY CONCATENATING INFORMATION FROM SUCCESSIVE      */     1   73
/* CARDS OF A DATA ENTRY. IT SUPPRESSES ONLY FOLLOWING BLANKS        */     1   73
/* IF THE FORMAT CODE IS 'U' AND ALSO PLACES THE CHARACTER 'a'      */     1   73
/*  AT THE END OF INFORMATION ON EACH CARD.                          */     1   73
/* ************************************************************      */     1   73
/*                                                                   */     1   73
         BEGIN=VERIFY(CARD,' ');                                            1   73
         FEGIN=BEGIN-1;                                                     1   74
         IANS=INDEX(CARD,'&ANS');                                           1   75
LOOP2:   I=VERIFY(CARD,' ');                                                1   76
            IF IANS¬=0 THEN                                                 2   77
            IANS=IANS-1;                                                    2   78
            IF I=0 THEN                                                     2   79
               DO;                                                          3   80
                  IF BEGIN=0 THEN                                           4   81
                     IF FLAG THEN                                           5   82
                     QUES=QUES||SAVCARD||'a';                               5   83
                     ELSE                                                   5   84
                     QUES=QUES||SAVCARD;                                    5   84
                  ELSE                                                      4   85
LOOP3:               IF FLAG THEN                                           5   85
                     QUES=QUES||SUBSTR(SAVCARD,IANS+1,CUMJ)||'a';           5   86
                     ELSE                                                   5   87
                     QUES=QUES||' '||SUBSTR(SAVCARD,BEGIN,CUMJ-FEGIN);      5   87
                  IF SIGNAL THEN                                            4   88
                  GOTO FINITO;                                              4   89
               GOTO RLOOP;                                                  3   90
               END;                                                         3   91
         CARD=SUBSTR(CARD,I);                                               1   92
         J= INDEX(CARD,' ');                                                1   93
         CUMJ=CUMJ+J+1-2;                                                   1   94
            IF J=0 THEN                                                     2   95
               DO;                                                          3   96
               CUMJ=80;                                                     3   97
               GOTO LOOP3;                                                  3   98
               END;                                                         3   99
         CARD=SUBSTR(CARD,J);                                               1  100
         GOTO LOOP2;                                                        1  101
FINISH:  A='1'B;                                                            1  102
FINITO:  END FORMATR;                                                       1  103
```

APPENDIX B.5

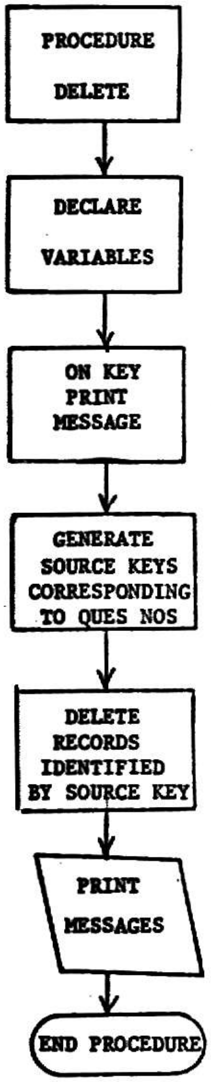
ADD


PROCEDURE ADD
DECLARE VARIABLES
ASSIGN QUES NO AND GENERATE SOURCE KEY
WRITE RECORD ONTO FILE KOKA
PRINT MESSAGE
END PROCEDURE

```
KSU'S PL/I NEATENER AND PRECOMPILER                    PAGE        9

(SUBSCRIPTRANGE,STRINGRANGE):ADD:                                           1    1
       PROC(Q,M_KEY,LEN);                                                   1    1
/*                                                                    */    1    2
/*                                                                    */    1    2
/**********************************************************************/    1    2
/* SUBROUTINE ADD ACCESSES THE INFORMATION POINTED TO BY THE          */    1    2
/* ARGUEMENT Q WHICH IS A POINTER TO THE STRUCTURE  DATA  AND AFT     */    1    2
/* ER ASSIGNING A QUESTION NUMBER,WRITES IT OUT ON TO FILE KOKA.      */    1    2
/* OVRLY1 AND OVRLY2 ARE USED AS RECORD I/O DOES NOT PERMIT           */    1    2
/* VARYING STRUCTURE ELEMENTS                                         */    1    2
/* A MESSAGE INDICATING THE QUESTION NUMBERS ADDED IS PRINTED         */    1    2
/* OUT                                                                */    1    2
/* ARGUMENT LEN IS THE LENGTH OF THE QUESTION AND ANSWER SO THAT      */    1    2
/* THE NECESSARY INFORMATION WILL BE MOVED TO OVRLY2                  */    1    2
/**********************************************************************/    1    2
/*                                                                    */    1    2
       DCL Q POINTER, INTER FIXED DEC(5),REGION CHAR(8),                    1    2
           OVRLY1 CHAR(4506) BASED(Q),OVRLY2 CHAR(4506) VAR,                1    2
           Q_NO CHAR(3),LEN BIN FIXED,                                      1    2
           M_KEY CHAR(3);                                                   1    2
       GET STRING(M_KEY)EDIT(INTER)(F(3));                                  1    3
       INTER=INTER+1;                                                       1    4
       PUT STRING(M_KEY)EDIT(INTER)(F(3));                                  1    5
       INTER=INTER/10;                                                      1    6
       REGION=INTER;                                                        1    7
       OVRLY2=M_KEY||SUBSTR(OVRLY1,4,LEN+3);                                1    8
       Q_NO=M_KEY;                                                          1    9
       WRITE FILE(KOKA)FROM(OVRLY2)KEYFROM(Q_NO||REGION);                   1   10
       PUT SKIP EDIT(M_KEY,'HAS BEEN ADDED TO THE DATASET')(COL(13),A(      1   11
       3),X(2),A);                                                          1   11
       END ADD;                                                             1   12
```

APPENDIX B.6

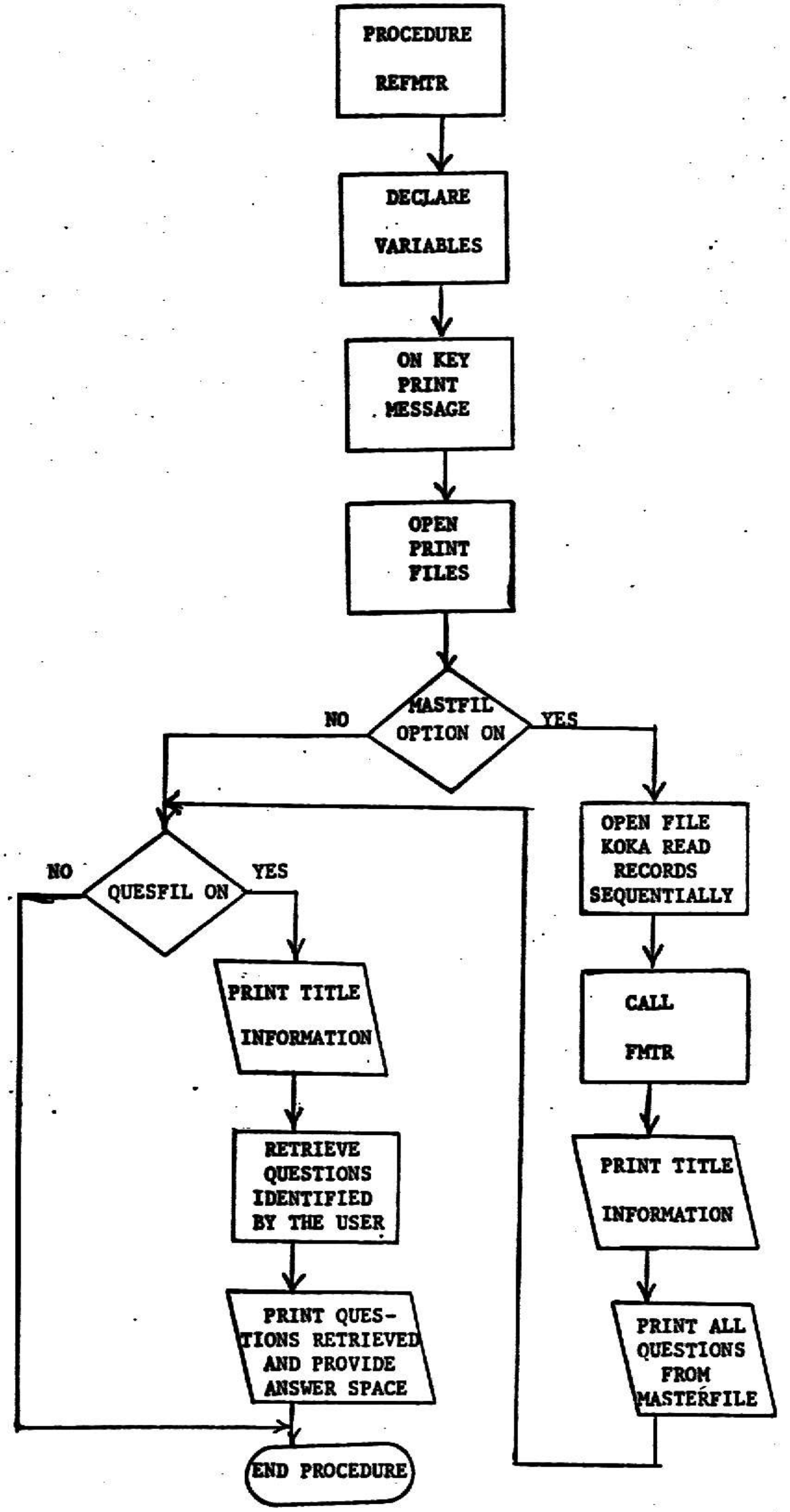
DELETE


PROCEDURE
DELETE
DECLARE
VARIABLES
ON KEY
PRINT
MESSAGE
GENERATE
SOURCE KEYS
CORRESPONDING
TO QUES NOS
DELETE
RECORDS
IDENTIFIED
BY SOURCE KEY
PRINT
MESSAGES
END PROCEDURE

```
KSU'S PL/I NEATENER AND PRECOMPILER               PAGE          8

(SUBSCRIPTRANGE,STRINGRANGE):DELETE:                                      1    1
        PROC(Q_ARRAY,N,M_KEY);                                            1    1
/*                                                                */      1    2
/******************************************************************/      1    2
/* SUBROUTINE DELETE ACCEPTS AN ARRAY OF QUESTION NUMBERS         */      1    2
/* Q_ARRAY AND AFTER DETERMINING THE REGION FOR EACH NUMBER,      */      1    2
/* DELETES THE CORRESPONDING RECORD FROM THE DATASET. IT PRINTS   */      1    2
/* OUT A MESSAGE TO THAT EFFECT.                                  */      1    2
/* ON KEY TAKES CARE OF WRONG SPECIFICATIONS OF QUESTION NUMBERS  */      1    2
/* BY THE USER                                                    */      1    2
/* M_KEY IS THE HIGHEST QUESTION NUMBER IN THE EXISTING DATASET   */      1    2
/* AND IF THIS QUESTION IS DELETED,THEN M_KEY IS UPDATED,SO THAT  */      1    2
/* THE QUESTION NUMBERS ASSIGNED WHILE ADDING WILL BE IN SEQUENCE */      1    2
/******************************************************************/      1    2
/*                                                                */      1    2
        DCL N BIN FIXED,K BIN FIXED INIT(1),Q_ARRAY(*) CHAR(3);           1    2
        DCL Q_NO CHAR(3),INTER FIXED DEC(5),REGION CHAR(8);               1    3
        DCL M_KEY CHAR(3);                                                1    4
        ON KEY(KOKA)                                                      1    5
           BEGIN;                                                         2    6
           PUT EDIT(Q_NO,'DOES NOT EXIST IN THE DATASET.DELETE IGNORED'   2    7
           )(SKIP,COL(10),A(3),X(2),A);                                   2    7
           GOTO LOOP;                                                     2    8
           END;                                                           2    9
           DO WHILE(K<=N);                                                2   10
           Q_NO=Q_ARRAY(K);                                               2   11
           K=K+1;                                                         2   12
           GET STRING(Q_NO)EDIT(INTER)(F(3));                             2   13
           INTER=INTER/10;                                                2   14
           REGION=INTER;                                                  2   15
           DELETE FILE(KOKA)KEY(Q_NO||REGION);                            2   16
              IF Q_NO=M_KEY THEN                                          3   17
                 DO;                                                      4   18
                 GET STRING(M_KEY)EDIT(INTER)(F(3));                      4   19
                 INTER=INTER-1;                                           4   20
                 PUT STRING(M_KEY)EDIT(INTER)(F(3));                      4   21
                 END;                                                     4   22
           PUT EDIT('QUES_NO',Q_NO,'HAS BEEN DELETED FROM DATASET')(SKI   2   23
           P(2),COL(13),A(7),A(3),X(2),A);                                2   23
LOOP:      END;                                                           2   24
        END DELETE;                                                       1   25
```

APPENDIX B.7

REFMTR


PROCEDURE
REFMTR
DECLARE
VARIABLES
ON KEY
PRINT
MESSAGE
OPEN
PRINT
FILES
MASTFIL
OPTION ON
NO
YES
OPEN FILE
KOKA READ
RECORDS
SEQUENTIALLY
NO
QUESFIL ON
YES
PRINT TITLE
INFORMATION
CALL
FMTR
RETRIEVE
QUESTIONS
IDENTIFIED
BY THE USER
PRINT TITLE
INFORMATION
PRINT QUES-
TIONS RETRIEVED
AND PROVIDE
ANSWER SPACE
PRINT ALL
QUESTIONS
FROM
MASTERFILE
END PROCEDURE

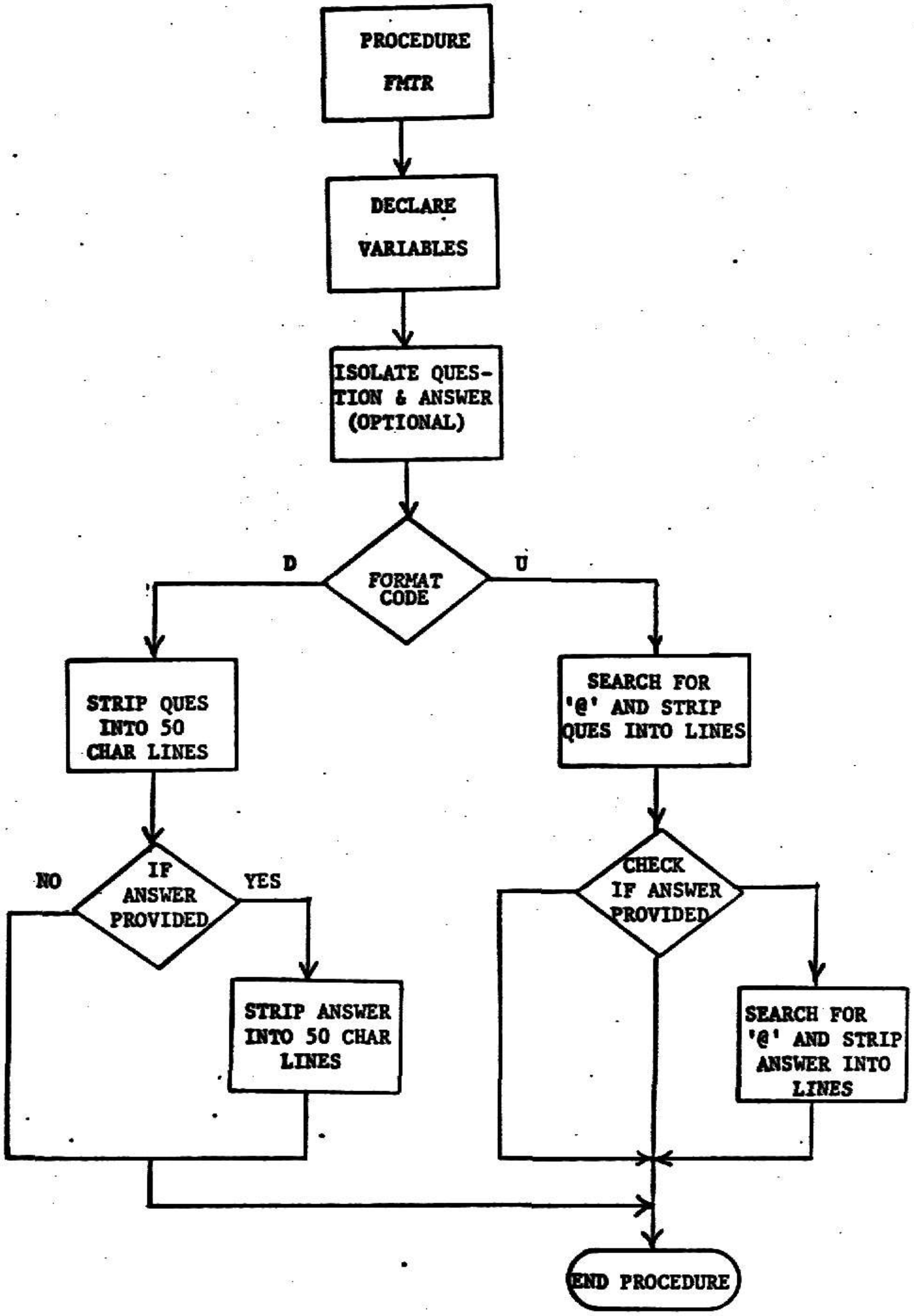
```
KSU'S PL/I NEATENER AND PRECOMPILER                    PAGE        4

(SUBSCRIPTRANGE,STRINGRANGE):REFMTR:                                     1    1
        PROC(R_KEY,N,M);                                                 1    1
/*                                                                 */    1    2
/*******************************************************************/    1    2
/* SUBROUTINE  REFMTR IS DESIGNED TO ACCEPT AN ARRAY OF QUESTION   */    1    2
/* NUMBERS AS AN ARGUMENT AND THEN TO RETRIEVE THE QUESTIONS       */    1    2
/* CORRESPONDING TO THESE NUMBERS. THE RETRIEVED INFORMATION       */    1    2
/* IS PROCESSED BY ANOTHER SUBROUTINE FMTR WHICH REFMTR INVOKES.   */    1    2
/* THE PROCESSED INFORMATION WHEN RETURNED BY FMTR IN AN ARRAY     */    1    2
/* CALLED LINE IS PRINTED OUT IN THE PRINT FILES. ARGUMENT         */    1    2
/* N OF THE SUBROUTINE INDICATES THE SIZE OF THE ARRAY OF          */    1    2
/* NUMBERS R_KEY. ARGUMENT M IS BIT(1) AND WHEN TURNED ON INDICAT  */    1    2
/* ES THAT THE MASTFIL PRINT OUT IS REQUIRED                       */    1    2
/*                                                                 */    1    2
/* THE ON KEY CONDITION CHECKS FOR MISTAKES IN QUESTION NUMBERS    */    1    2
/* SPECIFIED BY THE USER                                           */    1    2
/*******************************************************************/    1    2
/*                                                                 */    1    2
        DCL R_KEY(*) CHAR(3),(N,IJ,ANSNO,LINSIZ) BIN FIXED,              1    2
        M BIT(1),INTER FIXED DEC(5),REGION CHAR(8) INIT('       0'),     1    2
        1 NODE,                                                          1    2
         2 QUES_NO CHAR(3),                                              1    2
         2 F_CODE CHAR(1),                                               1    2
         2 S_CODE BIN FIXED,                                             1    2
        OVRLY CHAR(6) BASED(P),                                          1    2
        OVRLY2 CHAR(4506) VAR,LINE (100) CHAR(80),                       1    2
         KOKA FILE RECORD KEYED ENV(REGIONAL(3)),                        1    2
        FMTR ENTRY(CHAR(*) VAR,(*) CHAR(80),BIN FIXED,BIN FIXED,BIT(1))  1    2
        FFLAG BIT(1) INIT('0'B),I BIN FIXED INIT(0);                     1    2
        DCL INFO CHAR(3) INIT('INF');                                    1    3
        ON KEY(KOKA)                                                     1    4
           BEGIN;                                                        2    5
           PUT EDIT(R_KEY(I),'QUES_NO SPECIFIED DOES NOT EXIST IN THE D  2    6
           ATASET')(SKIP,COL(10),A(3),X(2),A);                           2    6
           GOTO CREAT_QUESFIL;                                           2    7
           END;                                                          2    8
        P=ADDR(NODE);                                                    1    9
        OPEN FILE(MASTFIL)PRINT LINESIZE(101);                           1   10
        OPEN FILE(QUESFIL)PRINT LINESIZE(100);                           1   11
        OPEN FILE(QANSFIL)PRINT LINESIZE(101);                           1   12
           IF M THEN                                                     2   13
           GOTO CREAT_MAST;                                              2   14
CONT:      IF N=0 THEN                                                   2   15
           GOTO FINISH;                                                  2   16
        CLOSE FILE(KOKA);                                                1   17
        OPEN FILE(KOKA)DIRECT INPUT;                                     1   18
/*                                                                 */    1   19
/*******************************************************************/    1   19
/* THIS SECTION OF THE PROGRAM RETRIEVES THE INFORMATION FROM THE  */    1   19
/* LOCATION RESERVED FOR TITLE INFORMATION. IF A $ SIGN IS         */    1   19
/* FOUND THEN NO TITLE INFORMATION IS PRINTED OUT. ELSE THE        */    1   19
/* RETRIEVED INFORMATION IS PRINTED IN APPROPRIATE COLUMNS         */    1   19
/*******************************************************************/    1   19
/*                                                                 */    1   19
        READ FILE(KOKA)INTO(OVRLY2)KEY(INFO||REGION);                    1   19
           IF OVRLY2¬='$' THEN                                           2   20
              DO;                                                        3   21
              J=INDEX(OVRLY2,'@');                                       3   22
```

```
KSU'S PL/I NEATENER AND PRECOMPILER                       PAGE        5
              PUT FILE(QUESFIL)EDIT(SUBSTR(OVRLY2,1,7),SUBSTR(OVRLY2,8,    3   23
              J-8),SUBSTR(OVRLY2,J+1))(SKIP,COL(15),A(7),COL(50),A,COL(    3   23
              80),A);                                                      3   23
              END;                                                         3   24
        PUT FILE(QUESFIL)EDIT('EXAM#','DATE:')(SKIP(2),COL(55),A(5),COL    1   25
        (75),A(5));                                                        1   25
        PUT SKIP(2)FILE(QUESFIL);                                          1   26
        PUT FILE(QANSFIL)EDIT(SUBSTR(OVRLY2,1,7),'QANSFILE')(SKIP,COL(1    1   27
        5),A(7),COL(50),A(8));                                             1   27
        PUT SKIP(4)FILE(QANSFIL);                                          1   28
/*                                                                   */    1   29
/*******************************************************************/    1   29
/* THIS SECTION OF THE PROGRAM PRINTS THE EXAM AND THE EXAM WITH     */    1   29
/* SOLUTIONS AFTER THE INFORMATION IS FORMATTED BY SUBROUTINE FMTR   */    1   29
/*            EXPLANATION OF FMTR ARGUMENTS                          */    1   29
/*                                                                   */    1   29
/* OVRLY2 HAS THE RETRIEVED QUESTION AND ANSWER.                     */    1   29
/* LINE IS THE ARRAY THROUGH WHICH THE FORMATTED LINES WILL BE       */    1   29
/* RETURNED                                                          */    1   29
/* ANSNO INDICATES THE LINE NUMBER WHERE THE ANSWER BEGINS. IF       */    1   29
/* ZERO IT MEANS THAT THERE IS NO ANSWER PROVIDED                    */    1   29
/* LINSIZ IS THE TOTAL NUMBER OF LINES OF THE ARRAY LINE -           */    1   29
/* INCLUDING THE QUESTION AND THE ANSWER.                            */    1   29
/* FUNCTION LINENO IS USED TO AVOID PRINTING A QUESTION AT THE       */    1   29
/* BOTTOM OF A PAGE, AND THEN PROVIDING BLANK SPACE FOR IT ON THE    */    1   29
/* NEXT PAGE. LINENO KEEPS TRACK OF THE LINE NUMBER BEING            */    1   29
/* PRINTED AND WHEN IT EXCEEDS 40, THEN THE PUT PAGE IS USED TO      */    1   29
/* SKIP TO A FRESH PAGE                                              */    1   29
/*******************************************************************/    1   29
/*                                                                   */    1   29
CREAT_QUESFIL:                                                             1   29
        I=I+1;                                                             1   29
           IF I>N THEN                                                     2   30
           GOTO FINISH;                                                    2   31
        GET STRING(R_KEY(I))EDIT(INTER)(F(3));                             1   32
        INTER=INTER/10;                                                    1   33
        REGION=INTER;                                                      1   34
        READ FILE(KOKA)INTO(OVRLY2)KEY(R_KEY(I)||REGION);                  1   35
        OVRLY=SUBSTR(OVRLY2,1,6);                                          1   36
        FFLAG='0'B;                                                        1   37
           IF F_CODE='U' THEN                                              2   38
           FFLAG='1'B;                                                     2   39
        CALL FMTR(OVRLY2,LINE,ANSNO,LINSIZ,FFLAG);                         1   40
           IF ANSNO=0 THEN                                                 2   41
              DO;                                                          3   42
                 IF LINENO(QUESFIL)>40 THEN                                4   43
                 PUT PAGE FILE(QUESFIL);                                   4   44
              PUT FILE(QUESFIL)EDIT(I,')',LINE(1))(SKIP(2),COL(10),F(2)    3   45
              ,A(1),X(2),A(80));                                           3   45
              PUT FILE(QANSFIL)EDIT(I,')',LINE(1))(SKIP(2),COL(10),F(2)    3   46
              ,A(1),X(2),A(80));                                           3   46
                 IF LINSIZ>1 THEN                                          4   47
                    DO IJ=2 TO LINSIZ;                                     5   48
                    PUT FILE(QUESFIL)EDIT(LINE(IJ))(SKIP,COL(15),A(80))    5   49
                    ;                                                      5   49
                    PUT FILE(QANSFIL)EDIT(LINE(IJ))(SKIP,COL(15),A(80))    5   50
                    ;                                                      5   50
                    END;                                                   5   51
                 GOTO SPACE;                                                    3   52
                 END;                                                           3   53
              IF LINENO(QUESFIL)>40 THEN                                        2   54
              PUT PAGE FILE(QUESFIL);                                           2   55
           PUT FILE(QUESFIL)EDIT(I,')',LINE(1))(SKIP(2),COL(10),F(2),A(1),     1   56
           X(2),A(80));                                                         1   56
           PUT FILE(QANSFIL)EDIT(I,')',LINE(1))(SKIP(2),COL(10),F(2),A(1),     1   57
           X(2),A(80));                                                         1   57
              IF ANSNO-1>1 THEN                                                 2   58
                 DO IJ=2 TO(ANSNO-1);                                           3   59
                 PUT FILE(QUESFIL)EDIT(LINE(IJ))(SKIP,COL(15),A(80));           3   60
                 PUT FILE(QANSFIL)EDIT(LINE(IJ))(SKIP,COL(15),A(80));           3   61
                 END;                                                           3   62
           GOTO CREAT_QANSFIL;                                                  1   63
SPACE:     PUT SKIP(S_CODE)FILE(QUESFIL);                                       1   64
           GOTO CREAT_QUESFIL;                                                  1   65
CREAT_QANSFIL:                                                                  1   66
           PUT FILE(QANSFIL)EDIT('ANS:',LINE(ANSNO))(SKIP(2),COL(10),A,X(1     1   66
           ),A(80));                                                            1   66
              IF ANSNO¬=LINSIZ THEN                                             2   67
                 DO IJ=(ANSNO+1)TO LINSIZ;                                      3   68
                 PUT FILE(QANSFIL)EDIT(LINE(IJ))(SKIP,COL(15),A(80));           3   69
                 END;                                                           3   70
           GOTO SPACE;                                                          1   71
/*                                                                   */         1   72
/*******************************************************************/          1   72
/* THIS SECTION OF THE PROGRAM PRINTS OUT THE FORMATTED MASTER      */          1   72
/* FILE OF QUESTIONS AND ANSWERS. THE PROCEDURE IS THE SAME AS      */          1   72
/* WHILE PRINTING THE EXAM, EXCEPT THAT ALL THE QUESTIONS IN        */          1   72
/* THE FILE ARE RETRIEVED.                                          */          1   72
/* NOTE:  THE FILE KOKA IS OPENED INPUT SEQUENTIAL HERE INSTEAD     */          1   72
/* OF DIRECT INPUT AS IN THE BEGINNING OF THE PROGRAM WHERE ONLY    */          1   72
/* SPECIFIC QUESTIONS WERE TO BE RETRIEVED                          */          1   72
/*******************************************************************/          1   72
/*                                                                   */         1   72
CREAT_MAST:                                                                     1   72
           ON ENDFILE(KOKA)                                                     1   72
           GOTO CONT;                                                           1   73
           CLOSE FILE(KOKA);                                                    1   74
           OPEN FILE(KOKA)SEQUENTIAL INPUT;                                     1   75
           READ FILE(KOKA)INTO(OVRLY2);                                         1   76
              IF OVRLY2¬='$' THEN                                               2   77
                 DO;                                                            3   78
                 J=INDEX(OVRLY2,'@');                                           3   79
                 PUT FILE(MASTFIL)EDIT(SUBSTR(OVRLY2,1,7),SUBSTR(OVRLY2,8,      3   80
                 J-9),SUBSTR(OVRLY2,J+1))(SKIP,COL(15),A(7),COL(50),A,COL(      3   80
                 80),A);                                                        3   80
                 END;                                                           3   81
LOOP:      READ FILE(KOKA)INTO(OVRLY2);                                         1   82
              IF LENGTH(OVRLY2)=3 THEN                                          2   83
              GOTO LOOP;                                                        2   84
           OVRLY=SUBSTR(OVRLY2,1,6);                                            1   85
           FFLAG='0'B;                                                          1   86
              IF F_CODE='U' THEN                                                2   87
              FFLAG='1'B;                                                       2   88
           CALL FMTR(OVRLY2,LINE,ANSNO,LINSIZ,FFLAG);                           1   89
              IF ANSNO=0 THEN                                                   2   90
                 DO;                                                            3   91
```

```
KSU'S PL/I NEATENER AND PRECOMPILER                    PAGE        7
                PUT FILE(MASTFIL)EDIT(QUES_NO,')',LINE(1))(SKIP(2),COL(10   3    92
                ),A(3),A(1),X(1),A(80));                                    3    92
                   IF LINSIZ>1 THEN                                         4    93
                      DO IJ=2 TO LINSIZ;                                    5    94
                      PUT FILE(MASTFIL)EDIT(LINE(IJ))(SKIP,COL(15),A(80))   5    95
                      ;                                                     5    95
                      END;                                                  5    96
                GOTO LOOP;                                                  3    97
                END;                                                        3    98
          PUT FILE(MASTFIL)EDIT(QUES_NO,')',LINE(1))(SKIP(2),COL(10),A(3)   1    99
          ,A(1),X(1),A(80));                                                1    99
             IF ANSNO-1>1 THEN                                              2   100
                DO IJ=2 TO(ANSNO-1);                                        3   101
                PUT FILE(MASTFIL)EDIT(LINE(IJ))(SKIP,COL(15),A(80));        3   102
                END;                                                        3   103
          PUT FILE(MASTFIL)EDIT('ANS:',LINE(ANSNO))(SKIP(2),COL(11),A,X(1   1   104
          ),A(80));                                                         1   104
             IF ANSNO¬=LINSIZ THEN                                          2   105
                DO IJ=(ANSNO+1)TO LINSIZ;                                   3   106
                PUT FILE(MASTFIL)EDIT(LINE(IJ))(SKIP,COL(15),A(80));        3   107
                END;                                                        3   108
          GOTO LOOP;                                                        1   109
FINISH: END REFMTR;                                                         1   110
```

APPENDIX B.8

FMTR


PROCEDURE FMTR
DECLARE VARIABLES
ISOLATE QUESTION & ANSWER (OPTIONAL)
FORMAT CODE
D
U
STRIP QUES INTO 50 CHAR LINES
SEARCH FOR '@' AND STRIP QUES INTO LINES
IF ANSWER PROVIDED
NO
YES
STRIP ANSWER INTO 50 CHAR LINES
CHECK IF ANSWER PROVIDED
SEARCH FOR '@' AND STRIP ANSWER INTO LINES
END PROCEDURE

```
(SUBSCRIPTRANGE,STRINGRANGE):FMTR:
       PROC(OVRLY,LIN,ANS,I,FFLAG);
/*                                                                 */
/*****************************************************************/
/* SUBROUTINE FMTR IS DESIGNED TO ACCEPT THE RETRIEVED QUESTION    */
/* AND ANSWER(OPTIONAL) AND THEN TO FORMAT THIS INFORMATION INTO   */
/* LINES ACCORDING TO THE FORMAT CODE SPECIFIED. THE ARGUEMENT     */
/* FFLAG,IF TURNED ON,INDICATES THE FORMAT CODE 'U',ELSE THE       */
/* FORMAT CODE IS 'D'.                                             */
/*                                                                 */
/*          DEFINITION OF ARGUEMENTS                               */
/* OVRLY IS THE QUESTION AND ANSWER PASSED BY REFMTR               */
/* LIN IS THE ARRAY THROUGH WHICH FMTR WILL RETURN THE FORMATTED   */
/* INFORMATION.                                                    */
/*    ANS INDICATES THE LINE NUMBER WHERE THE ANSWER BEGINS.       */
/* I  INDICATES THE SIZE OF THE ARRAY LIN.                         */
/*****************************************************************/
/*                                                                 */
       DCL OVRLY CHAR(*) VAR,LIN(*) CHAR(80),
       (ANS,I) BIN FIXED,
       (J,L,IJ) BIN FIXED,FLAG_ANS BIT(1) INIT('0'B),
       FFLAG BIT(1),BEGIN BIN FIXED INIT(1),
       QUES CHAR(4500) VAR,ANSWER CHAR(1500) VAR;
       LIN=' ';
       ANS=0;
       I=0;
/*                                                                 */
/* ************************************************************** */
/* THIS SECTION OF THE PROGRAM CHECKS TO SEE IF THE ANSWER IS      */
/* PROVIDED AND IF SO,TURNS ON FLAG_ANS. THE LENGTHS OF THE        */
/* QUESTION AND THE ANSWER ARE ALSO OBTAINED.                      */
/* ************************************************************** */
/*                                                                 */
       J=INDEX(OVRLY,'&ANS');
          IF J=0 THEN
             DO;
             FLAG_ANS='1'B;
             QUES=SUBSTR(OVRLY,7);
             L=LENGTH(QUES);
             GOTO CHECK_FLAG;
             END;
       QUES=SUBSTR(OVRLY,7,J-7);
       ANSWER=SUBSTR(OVRLY,J+4);
       L=LENGTH(QUES);
       ANSL=LENGTH(ANSWER);
CHECK_FLAG:IF FFLAG THEN
          GOTO STRIP_QUESU;
/*                                                                 */
/* ************************************************************** */
/*      FORMAT CODE D                                              */
/* THIS SECTION OF THE PROGRAM FORMATS THE INFORMATION INTO AS     */
/* MANY 80 CHARACTER LINES AS POSSIBLE  AND MOVES THE REMAINING    */
/* INFORMATION TO LINE,LEFT JUSTIFIED.                             */
/* ************************************************************** */
/*                                                                 */
STRIP_QUESD:
       L=L-50;
          IF L<0 THEN
```

```
                 DO;                                                      3   22
                 I=I+1;                                                   3   23
                 L=L+50;                                                  3   24
                 SUBSTR(LIN(I),1,L)=SUBSTR(QUES,BEGIN,L);                 3   25
                 GOTO STRIP_ANSD;                                         3   26
                 END;                                                     3   27
        I=I+1;                                                            1   28
        LIN(I)=SUBSTR(QUES,BEGIN,50);                                     1   29
        BEGIN=BEGIN+50;                                                   1   30
        GOTO STRIP_QUESD;                                                 1   31
STRIP_ANSD:IF FLAG_ANS THEN                                               2   32
           GOTO FINISH;                                                   2   33
        BEGIN=1;                                                          1   34
        ANS=I+1;                                                          1   35
LOOP1:  ANSL=ANSL-50;                                                     1   36
        I=I+1;                                                            1   37
           IF ANSL<O THEN                                                 2   38
              DO;                                                         3   39
              ANSL=ANSL+50;                                               3   40
              SUBSTR(LIN(I),1,ANSL)=SUBSTR(ANSWER,BEGIN,ANSL);            3   41
              GOTO FINISH;                                                3   42
              END;                                                        3   43
        LIN(I)=SUBSTR(ANSWER,BEGIN,50);                                   1   44
        BEGIN=BEGIN+50;                                                   1   45
        GOTO LOOP1;                                                       1   46
/*                                                                   */   1   47
/* ***************************************************************** */   1   47
/*          FORMAT CODE U                                            */   1   47
/* THIS SECTION OF THE PROGPAM CHECKS FOR THE CHARACTER'@' TO        */   1   47
/* BREAK DOWN THE RETRIEVED INFORMATION INTO LINES.                  */   1   47
/* ***************************************************************** */   1   47
/*                                                                   */   1   47
STRIP_QUESU:                                                              1   47
        J=1;                                                              1   47
STRIP_LOOP:                                                               1   48
        IJ=INDEX(QUES,'@');                                               1   48
        I=I+1;                                                            1   49
        LIN(I)=SUBSTR(QUES,1,IJ-1);                                       1   50
        J=J+IJ;                                                           1   51
           IF J>L THEN                                                    2   52
           GOTO STRIP_ANSU;                                               2   53
        IJ=IJ+1;                                                          1   54
        QUES=SUBSTR(QUES,IJ);                                             1   55
        GOTO STRIP_LOOP;                                                  1   56
STRIP_ANSU:IF FLAG_ANS THEN                                               2   57
           GOTO FINISH;                                                   2   58
        ANS=I+1;                                                          1   59
LOOP2:  IJ=INDEX(ANSWER,'@');                                             1   60
        I=I+1;                                                            1   61
        LIN(I)=SUBSTR(ANSWER,1,IJ-1);                                     1   62
        IJ=IJ+1;                                                          1   63
           IF IJ>ANSL THEN                                                2   64
           GOTO FINISH;                                                   2   65
        ANSWER=SUBSTR(ANSWER,IJ);                                         1   66
        GOTO LOOP2;                                                       1   67
FINISH: END FMTR;                                                         1   68
```

REFERENCES

1. G. Salton, Automatic Information Organization and Retrieval, McGraw Hill Book Co., New York, 1968.

2. IBM SYSTEM/360 Operating System, PL/1 (F), Programmer's Guide, Order No. GC28-6594-8.

3. IBM SYSTEM/360 Oerating System, PL/1 (F), Language Reference Manual, Order No. GC28-8201-3.

4. IBM SYSTEM/360, Job Control Language, G. D. Brown, John Wiley & Sons Inc., New York.

5. IBM SYSTEM/360, OS Utilities, Order No. GC28-6586-13.

ACKNOWLEDGEMENTS

I am grateful to my Major Professor Dr. R. K. Brewer, Assistant Professor, Department of Computer Science for the help and guidance he offered while doing this work. I am also thankful to him for editing my report which has helped in refining this work.

I am also grateful to John Devore, Chief Consultant, Computing Center, Kansas State University for the assistance he provided in debugging my programs.

EXAM GENERATION SYSTEM

by

KOKA RAVINDRA

B. E., (Electronics and Communication Engineering)

University of Madras, India, 1971

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1973

ABSTRACT

This work was undertaken with the objective to develop and implement an Exam Generation System which would facilitate automatic generation of course examinations, the questions appearing in the examination being retrieved from a previously created database. Three PL/1 programs have been developed to create and update the database and to obtain a formatted examination with answer space provided for each question. The programs have been established as cataloged procedures.