A STUDENT INFORMATION SYSTEM
FOR
MICROCOMPUTERS

A THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF SCIENCE

IN THE GRADUATE SCHOOL OF THE

TEXAS WOMAN'S UNIVERSITY

COLLEGE OF NATURAL AND SOCIAL SCIENCES

BY

DOUGLAS A. WAECHTER, B. MATHEMATICS

DENTON, TEXAS

DECEMBER 1981

ACKNOWLEDGEMENTS

I wish to thank the members of my committee for their help and understanding. I thank Dr. Saffer for sharing his expertise, Ms. Ferrell for her many comments and useful suggestions, and Dr. Christy for his patience, understanding, and thoughtfulness.

A special thanks to Norma Burgert for all her helpful hints and comments during our many "DOS" conversations. I also thank my wife, Ruth, for her understanding, encouragement, and patience.

This study is dedicated to my daughter Trisha. May she accept and conquer the challenges of the future.

TABLE OF CONTENTS

APPENDICES

CHAPTER I

INTRODUCTION

Data base management systems (DBMS), and management information systems (MIS) are terms that are frequently used in modern computer installations. Although both concepts originated with the automation of accounting, production, and inventory control, they evolved into major systems as the emphasis in computer processing shifted in an attempt to meet management's information needs.

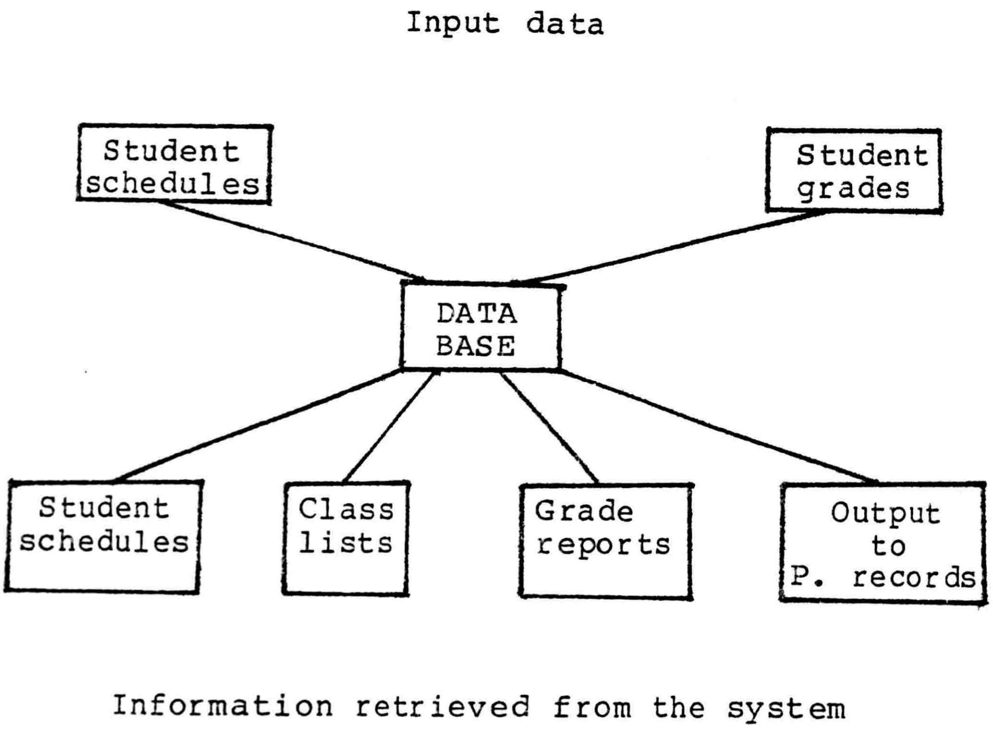
The difference between data base management systems and management information systems is subtle but real. C. J. Date (1) defines a data base management system as a collection of stored operational data used by the application systems of some particular enterprise. An appropriate definition of an management information system is one presented by Riley (2). An information management system is a system for collecting, sorting, retrieving, and processing information which is used, or desired, by one or more managers in the performance of their duties. Thus a data base management system is part of a larger information management system. This study will focus on data base management systems and will investigate the feasibility of using microcomputers to implement and maintain a small, stand-alone data base system.

George M. Scott (3) provides a definition of a modern data base as a collection of computer files of data structured to enable efficient updating, maintenance, reporting, and storage of data and to enable rapid retrieval of all stored data that must be brought together for a particular operation or managerial purpose. This definition implies two major dimensions of a data base. The first dimension relates to structuring data files to enable efficient updating, maintenance, reporting, and storage of data. Data base structuring involves complex data-file design and data retrieval technology. The various data structural models discussed in this study include the hierarchical or tree model, the network model, and the relational model.

The second dimension of data bases referred to in Scott's definition, enabling rapid retrieval of all stored data that must be brought together for a particular operation or managerial purpose, illustrates the relationship that exists between data base systems and management information systems. The functions of query languages and the role they play in data base management design is also included within this second dimension. This study is concerned with the design and implementation of a data base system. The theoretical discussion of query languages, their implementation, and completeness is not included.

To design any data base management system, it is necessary to define the information to be represented within the system. A. T. F. Hutt (4) defines a "Perceived Field" of a data base as that part of the real world which is to be reflected within the system. Hutt stresses the importance of deciding on the size of the perceived field and the necessity of allowing the field of perception to grow via a number of phased expansions of the data system. The "Field of Perception" of the data base to be implemented in this study is outlined in figure 1. Since the purpose of the study is to investigate the feasibility of using micro computers to implement and manage a stand-alone data base, a small environment has been selected. The data base to be implemented is to consist of the schedules of students, class lists, and student grade reports for a high school environment of five hundred students.

In designing a data base management system, several basic functions of the system must be considered. The system must be able to retrieve particular records, insert new records, delete individual records, and update specific



Procedures:

1. Input student schedules.

2. Produce class lists.

3. Input student grades. (mid and end of semester)

4. Produce student grades reports.

5. Prepare output for the permanent record file.

<u>note</u>: An environment of five hundred students is assumed and although this implementation does allow for reporting to the permanent record file, this file is not included in the system system as developed in this study.

Figure 1: Perceived Field

information represented within the system. Fundamental to the design of the data manipulation language (DML) which supports these basic functions is the conceptual view of the data. In particular, the data structure dictates the design of the corresponding data manipulation language.

At present, the three best known data structural models are: the hierarchic or tree model, the network model, and the relational model. The conceptual view of data suggested by these models dictates the design of the DML needed to support the basic functions of inserting records into the system, deleting data from the system, and updating information contained within the system. The hierarchic or tree model employs a hierarchical design of data and was adapted by the Conference of Data Systems Languages (CODASYL) and presented in their 1971 DBTG report. This model suggests that real world data lies within hierarchical structures with one record at the top of the tree, referred to as the root node, and other records dependent on it. In general any record may have any number of dependents and each of these dependents may have any number of dependents, and so on. In the hierarchical model each dependent must have exactly one successor. Figure 2 illustrates a hierarchical view of the data represented in a student information system.

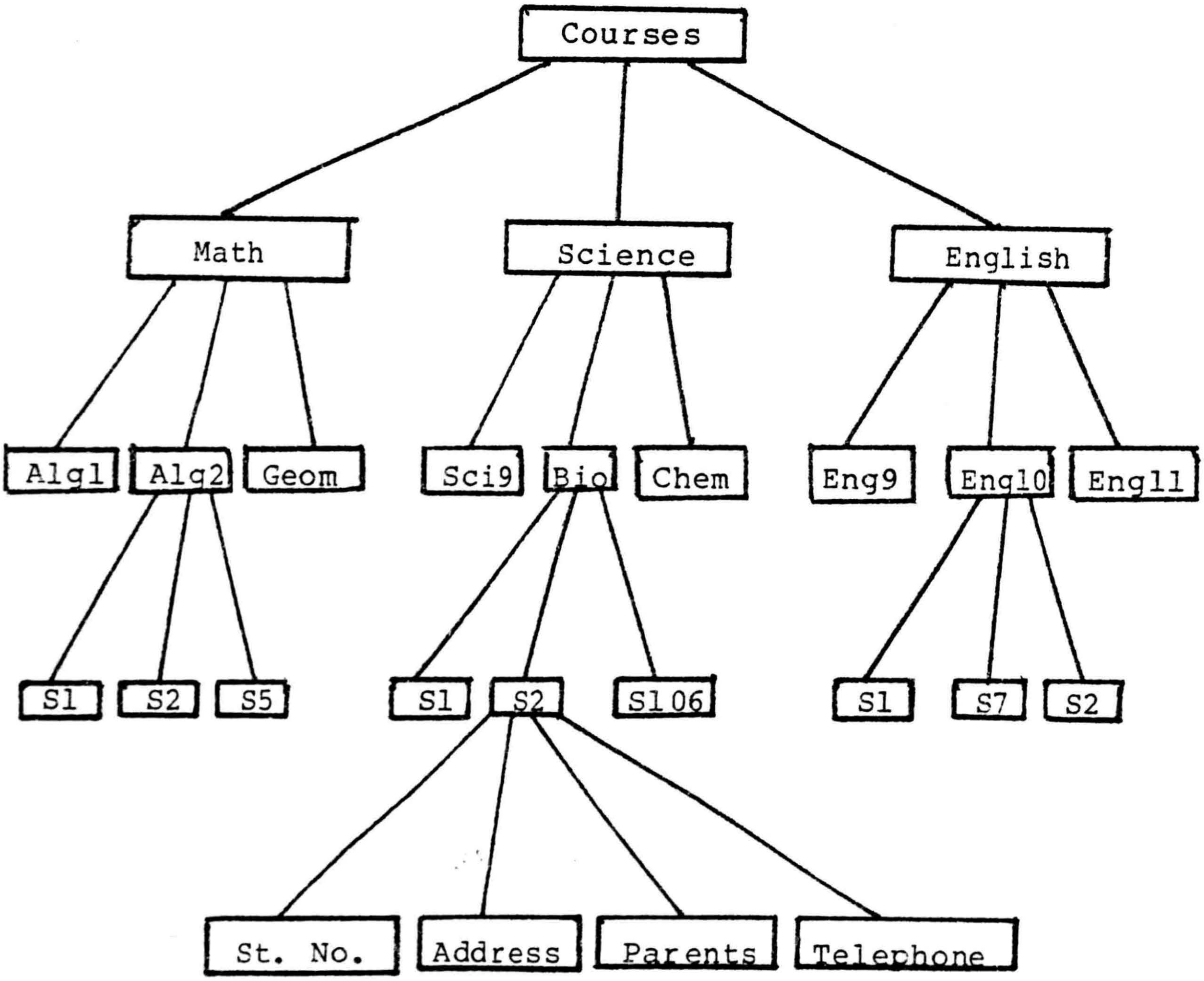


Figure 2: Hierarchical View of Student Information Data

The DML needed to maintain the basic functions of insertion, deletion, and updating of information becomes complex. It is impossible to insert a new student until his classes have been defined. The procedure required to delete an existing student is also complex since the data base must be searched in its entirety and all occurrences of the particular student must be deleted. It also follows that if we delete all classes of a particular student, we have deleted the existence of the student. Thus the insert and delete anomalies reflect a similar problem. The update procedure also involves searching the entire structure. To update a particular student's address, for example, the entire structure must be searched and each occurrence of the address modified.

The network data structural model is a more general structure than the hierarchical model. A given record in the network model may have any number of immediate superiors whereas a record is restricted to exactly one superior in the hierarchical model. The network approach allows many-to-many relationships whereas the hierarchical model is restricted to one-to-many relationships. Figure 3 illustrates a network view of the student record data representation. The procedures for inserting, deleting, and updating information become conceptually simple within the network model and the anomalies discussed with the hierarchical model do not arise. A new student could be added to the structure before his schedule is defined and to delete a student, or update a student's address we need not search the entire structure. The prime disadvantage of the network model, however, lies in the complexity of the DML required. The programming involved becomes extremely sophisticated resulting in relatively high implementation and maintenance costs.

The relational approach, first proposed by Codd (5) in 1971, is a different approach to describing and manipulating data. It views the data base as a simple collection of two dimensional tables called relations. Although conceptually simple, as illustrated in figure 4, a relation can be explicitly defined as a two dimensional table with m rows, each made up of a set of n-tuples. Each of the columns in a relation is a set of values of one attribute and is referred to as a domain. The power of a relational data base exists in the data manipulation language rather than in the data structure. Figure 4 illustrates how the data in the student information system can be represented by a number of relations. The inserting, deleting, and updating procedures become relatively simple. A new student can be inserted into the student-schedule or student-information relations

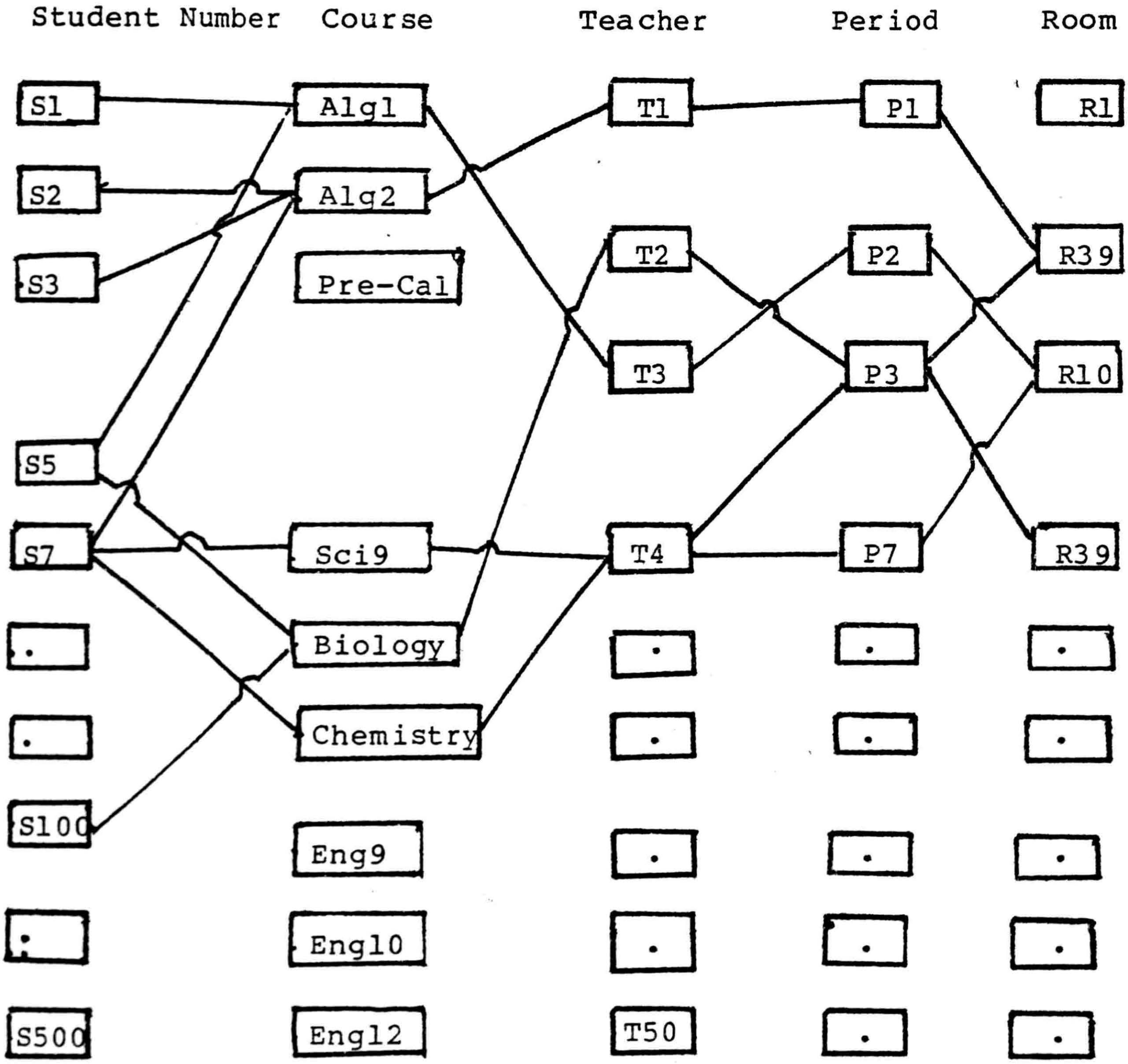


Figure 3: Network View of Student Information Data

Student-Schedule Relation:

Student Number	courses			
S1	Alg1	Eng10	Biology	P.E.
S2	Eng9	French	Sci9	St. Hall
S3	Eng12	Physics	Pre-Cal	German
S4	Hist9	Alg1	Biology	French
S5	Am.Prob.	P.E.	Eng10	Geometry
.	.	.	.	.
.	.	.	.	.
S500	.	.	.	.

Student-Information Relation:

Student Number	Address	Parents	Telephone
S1	521 Windsor Dr.	John Jones	383-3995
S2	436 Johnson Dr.	Jane Smith	452-6547
.	.	.	.
.	.	.	.

Figure 4: Relational View of Student Information Data

as required, a student can be deleted from any relation without affecting other relations, and procedures to update information require single operations.

A historical look at the development of Data Base Management Systems shows that although recent trends appear to be moving toward the relational view of data, commercially available systems generally employ the hierarchical or network data structural model. Figure 5, taken from Data Base Management Systems (6), outlines the general data bases that were commercially available in 1977 and illustrates that the CODASYL hierarchical or tree structural design is the most widely used data model. IBM's Information Management System (IMS2) also uses a hierarchical data model but does not follow CODASYL standards and thus receives a classification of its own. Although there are no general relational data base systems commercially available, several experimental relational systems have been implemented with promising results. IBM's IS/1 shows the feasibility of supporting relational algebra, a data manipulation language proposed by Codd (5), and the experimental SYSTEM R, being developed at the IBM Research Laboratory, does classify as a pure relational system.

The specific aims of this study are to investigate the feasibility of using microcomputers to implement and

SYSTEM	SUPPLIER	CLASSIFICATION
DMS-11	Burroughs	Network
DMS-170	Control Data	CODASYL DBTG
TOTAL	CINCOM	Network
IDMS	Cullinane	CODASYL DBTG
DBMS/10	DEC	CODASYL DBTG
IMS2	IBM	Hierarchical
SYSTEM 2000	MRI	Tree, Inverted
ADABAS	Software AC	Network inverted
DMS 1100	UNIVAC	CODASYL DBTG

Figure 5: A partial list of Data Management Systems commercially available

maintain a stand-alone data base to be used in a high school environment. A data base management system, with the field of perception as outlined in figure 1, will be implemented using the relational data model. The multi-step design methodology, presented by Hutt (4), is followed. The steps involved in Hutt's multi-step methodology are:

1. THE OBJECT ENVIRONMENT: The perceived field is identified and real world objects that exist within it are defined.

2. THE INFORMATION ENVIRONMENT: A number of relations in third normal form, as defined in chapter two of this document, are defined.

3. THE STORED ENVIRONMENT: The indexes and storage structures are defined.

4. THE SOFTWARE ENVIRONMENT: The development of the computer software which operates on the stored data to perform the required results.

CHAPTER II

THE ARCHITECTURE

2.1 Definitions

C. J. Date (1) defines a relation R on a collection of N sets D1, D2,...,Dn as a set of ordered n-tuples <d1,d2,...,dn> such that d1 belongs to D1, d2 belongs to D2,...,dn belongs to Dn. Sets D1, D2,..., Dn are called the domains of the relation R and the value of n is the degree of the relation. It is important to distinguish between a domain and an attribute which is drawn from the domain. An attribute represents the use of a domain within a given relation. To emphasize this distinction; the attributes for the relation Student-Schedule given in figure 4 of chapter 1 consist of course1, course2, course3, and course4, which are all taken from the domain of courses offered. Generally speaking, if a relation is considered to be a table of values with m rows and n columns, the attributes can be considered to be the names given to the individual columns.

In any relation there will be at least one attribute, or a combination of attributes, that are unique within the relation. If the relation contains one such attribute, this attribute is known as the primary key of the relation. In the relation Student-Schedule the attribute (student number) serves as the primary key since each occurrence of a specific student number uniquely defines a 5-tuple. If a relation contains more than one attribute combination possessing this unique identification property, one attribute combination is arbitrarily chosen as the primary key while the remaining "candidate keys" are referred to as alternate keys.

The problem of defining what relations are needed and what their attributes should be is the fundamental problem of designing a relational data base. The concept of normalization of relations plays an important role in defining appropriate relations to adequately represent the data. The following discussion introduces the concept of normalization and discusses why it is important to properly define the relations in third normal form when designing a relational data base.

Every relation is considered to be in first normal form if each entry in the relation is atomic (i.e. if each entry is nondecomposable as far as the system is concerned). This definition is fundamental in defining relations and simply means that each entry in the table consists of precisely one value, never a set of values. A relation that is only in first normal form leads to anomalies similar to those encountered by hierarchies discussed in chapter 1. For example, the relation given in figure 6(a) is in first normal form and problems occur with the three basic operations of inserting, deleting, and updating information. We cannot enter the fact that a new student exists until he has defined at least one class, if we delete all classes we delete the student, and if the teacher of a particular course is changed the entire structure must be searched and all appropriate changes made.

In order to resolve some of these difficulties, first normal form relations are transformed into second normal form relations. Before we can define second normal form relations, the concepts of functionally and fully functionally dependencies must be understood. An attribute Y of a relation R is said to be functionally dependent on attribute X of R if and only if each X-value in R is associated with precisely one Y-value in R. Furthermore, attribute Y is fully functionally dependent on attribute X if it is functionally dependent on X and not functionally dependent on any proper subset of X. In the relation given in figure 6(a) the attribute (Sname) is functionally dependent on the primary key (student number,course number) but it is not fully functionally dependent on this attribute since (Sname) is also dependent on (student number) which is a proper subset of the attribute combination (student number,course number).

We now define a relation to be in second normal form if and only if it is in first normal form and every nonkey attribute is fully functionally dependent on the primary key. It is clear that the relation Student-Information given in figure 6(a) is not in second normal form since the nonkey attribute (Sname) is not fully dependent on the key (Snumber,Cnumber). A relation in first normal form can always be transformed into equivalent relations in second normal form. The process involves eliminating the non-full functional dependencies. The functional diagram given in figure 6(b), gives functional dependencies and leads to defining the relations in second normal form given in figure 7. The anomalies discussed above do not exist with relations in second normal form since a student can exist before classes are defined by simply inserting the appropriate tuple into the student-name relation. The relation Teacher(Cnumber, Tnumber, Tname), however, still creates problems. To be specific, the dependency of (Tname) on the primary key is transitive. Each (Cnumber) determines a (Tnumber) which in turn determines a (Tname). The anomality created by the transitivity exists in that we cannot insert a teacher into the data base until a specific class has been defined for him to teach.

To overcome the above anomality we further reduce the second normal form relations to third normal form. A relation is in third normal form if and only if it is in second normal form and every nonkey attribute is nontransitively dependent on the primary key. This reduction process is accomplished by eliminating transitive dependence and the relation Teacher(Cnumber,Tnumber,Tname) is reduced to two relations. The relations CT(Cnumber,Tnumber) with primary key (Cnumber) and TN(Tnumber,Tname) with primary key (Tnumber) replace the relation Student-Information. Figure 8 gives the four relations in third normal form equivalent to the given Student-Information relation.

Several other normal forms have been defined which deal with relations with more than one candidate key but since all relations used in the design and implementation of the data base discussed in this study contain a single key, relations defined in third normal form will ensure that the above anomalies will not occur when implementing the insert, delete, and update procedures.

2.2 THE OBJECT ENVIRONMENT

As illustrated in figure 1, the student courses and grades are to be input into the system and the system is to produce student schedules, class lists, semester grade reports, and output to the permanent record file. This system does not include the permanent record file but produces output containing class codes, grades, teacher codes, and teacher comment codes. The object environment of the relational data base to be implemented is:

Student number: A five digit integer. The first two digits identify the graduation year and the remaining three digits identify the student. Student number 85123, for example, identifies that student number 123 will graduate in 1985.

Course number: A three digit integer representing a particular course. The specific departments are identified by the leading digit and individual courses within the department are identified by the remaining two digits. In this study the departments are identified by the following scheme.

1 identifies an elective.

2 identifies an English course.

3 identifies a history course.

4 identifies a mathematics course.

5 identifies a physical education course.

6 identifies a science course.

This identification is random and has no bearing on the design of the system. A course such as 421 will imply mathematics course number 2, section 1.

Teacher Number: A two digit integer identifying the teacher of a particular course.

Grades: An integer, maximum of two digits, representing the student grade in a particular course. The system allows for a mid-semester and end of semester grades.

Teacher comment: A one digit integer (0-9) representing a teacher comment code. Two comments are represented; a mid-semester comment and a end of semester comment.

Room Number: A two digit number identifying the room number for a particular class.

2.3 THE INFORMATION ENVIRONMENT

The information is represented within the system by a number of relations in third normal form. The relations required are the Student-Schedule relation, the Teacher-Schedule relation, and the Class-Information relation. Although the relation Student-Name is given here for completeness it is not included within the system as implemented in this study.

The Student-Schedule relation has domains student number, course number, grades, and teacher comment. Since a seven period day is assumed, the domains course number, grades, and teacher comment are repeated to form several attributes. Figures 9 and 10 give a detailed description of these relations including the primary keys and descriptions of the attributes of each of the relations.

2.4 THE STORED ENVIRONMENT

In this section the storage structures are defined. The Student-Schedule relation is stored externally as a random access file with a defined record length of one hundred and four bytes. Each record contains one 36-tuple of the relation as outlined in figure 11. Since the system is designed to accommodate a maximum of five hundred students, this file contains five hundred records. The file is indexed by the key attribute student-number. The index is structured as a binary tree and is stored in core memory as a five hundred by four array and on auxiliary disk as a sequential file.

The Teacher-Schedule relation has a structure similar to the Student-Schedule relation with each 8-tuple stored as one record of a random access file. This file has a defined record length of thirty one bytes and figure 12 gives a detailed description of one record of this file. The Teacher-Schedule index is also structured as a binary tree and is stored in core memory as a fifty by four array and on the disk storage as a sequential file.

The Class-Information relation with attributes (Cnumber, Tnumber, Rnumber, Csize,S1,S2,...,S35) is also stored as a random access file where each record has a defined length of two hundred and twenty five bytes. Each record of the file stores the course number, teacher number, period number, room number, class size, and the list of students currently enrolled in the class. The Class-Information index is also structured as a binary tree and is stored in core in an one hundred by fifty array and on the disk as a sequential file.

-a-

Relation: Student-Information

Attributes (Snumber,Sname,Cnumber,Tnumber,Tname)

Primary key (Snumber,Cnumber)

Snumber	Sname	Cnumber	Tnumber	Tname
S1	John Smith	C7	T6	Mr. White
S5	Jack Jones	C9	T1	Ms. Brent
S46	Sam Bright	C7	T6	Mr. White
.	.	.	.	.
S498	Howard Go	C18	T15	Mr. Black

-b-

Functional Dependencies

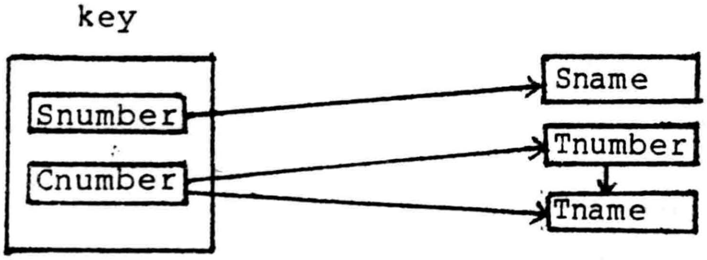


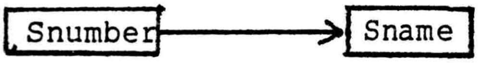
Figure 6: Relation Student-Information in "First Normal Form"

-a-

Relation: Student-Name
Attributes: (Snumber,Sname)
Key: (Snumber)

Snumber	Sname
S1	John Smith
S5	Jack Jones
S45	Sam Bright
S498	Howard Go

Function Dependency



-b-

Relation: Teacher
Attributes: (Cnumber,Tnumber,Tname)

Cnumber	Tnumber	Tname
C7	T6	Mr. White
C9	T1	Ms. Brent
C7	T6	Mr. White
C18	T15	Mr. Black

Functional Dependencies

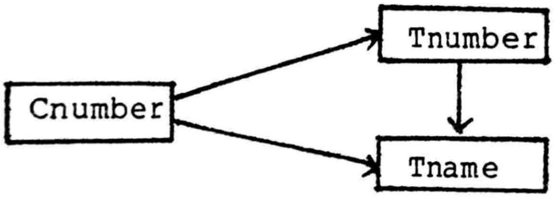


Figure 7: Relation Student reduced to "Second Normal Form"

Relation: Student
Attributes: (Snumber, Sname)
Key: (Snumber)

Snumber	Sname
S1	John Smith
S5	Jack Jones
.	.
S498	Howard Go

Relation: Schedule
Attributes: (Snumber, Cnumber)
Key: (Snumber)

Snumber	Cnumber
S1	C7
S5	C9
.	.
S498	C18

Relation: CT
Attributes: (Cnumber, Tnumber)
Key: (Cnumber)

Cnumber	Tnumber
C7	T6
C9	T1
.	.
C18	T15

Relation: TN
Attributes: (Tnumber, Tname)
Key: (Tnumber)

Tnumber	Tname
T6	Mr. White
T1	Ms. Brent
.	.
T15	Mr. Black

Figure 8: Relations in "Third Normal Form"

-a-

Relation: Student-Schedule

Attributes:

```
          ....... period 1 .......    ...... period 2 ......
(SNumber,c1, g11, co11, g1f, co1f, c2, g21, co21, g2f, co2f,

                         ..... period 7 ......
                      ,g7, g71, co71, g7f, co7f)
```

Key: Snumber

ci....implies...course number corresponding to appropriate period (7 period day).

gi1...implies...the mid-semester grade.

coi1..implies...the mid-semester comment.

gif...implies...the end of semseter grade.

coif..implies...the end of semester comment.

-b-

Relation: Teacher-Schedule

Attributes: (Tnumber, C1, C2, C3, C4, C5, C6, C7)

Key: Tnumber

Tnumber...implies...Teacher number

Ci ....implies...Course number of the ith class period.

Figure 9: Relations Student-Schedule and Teacher-Schedule

-a-

Relation: Class-Information

Attributes:(Cnumber,Tnumber,Pnumber,Rnumber,Csize,S1,...,Sn)
Key: Cnumber

Cnumber..represents the course number.

Tnumber..represents the teacher number.

Pnumber..represents the period number.

Rnumber..represents the room number.

Csize....represents the current class size.

Si.......represents a student number.emacs

-b-

Relation: Student-Name

Attributes: (Snumber, Sname, Pname, Address, Telephone)
Key: Snumber

Snumber...represent the student number.

Sname.....represents the student name.

Pname.....represents the parent's name.

Address...represents the student's address.

Telephone.represents the student's phone number.

Figure 10: Relations Class and Student

```
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
.8.5.1.2.3.).4.1.1.).8.9.).5.).7.2.).6.).1.3.1.).8.5.).6.).
 0         5         10        15        20         25

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
.8.1.).1.). . . . . . . . . . . . . . . . . . . . . . . . .
  30        35        40        45        50        55

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
    60        65        70        75        80        85

. . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . .).
     90        95        100
```

bytes:	0.....4	student number.
	5	end of field marker.
	6,7,8	course number (period 1).
	9	end of field marker.
	10,11	mid-semester grade.
	12	end of field marker.
	13	mid-semester comment.
	14	end of field marker.
	15	end of semester grade.
	17	end of field marker.
	18	end of semester comment.
	19	end of field marker
	.	.
	104	end of record marker.

Figure 11: Byte map of Student-Schedule File

```
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
.5.4.).9.9.9.).5.1.1.).5.1.2.).9.9.9.5.1.3.).5.1.4.).5.1.
 0         5         10        15        20        25

. . .
.1.).
 31
```

bytes:	0,1	teacher number.
	2	end of field marker.
	3,4,5	course (period 1).
	6	end of field marker.
	.	.
	.	.
	30	end of record marker.

Figure 12: BYTE MAP OF TEACHER SCHEDULE FILE

```
. . . . . . . . . . . . . . . . . . . . . . . . . . . . .
.5.1.1.).5.4.).2.).2.2.).1.8.). . . . . . . . . . . . . .
 0      5       10        15        20        25

. . . . . . . .         .    .           . . . .
. . . . . . . .         .    .           . . .).
 30         35                              225
```

bytes:	0,1,2	course number.
	3	end of field marker.
	4,5	teacher number.
	6	end of field marker.
	7	period number.
	8	end of field marker.
	9,10	room number.
	11	end of field marker.
	12,13	class size.
	14	end of field marker.
	15-224	student numbers (maximum of 35).
	225	end of record marker.

Figure 13: BYTE MAP OF CLASS-INFORMATION FILE

CHAPTER III

THE SOFTWARE ENVIRONMENT

The student information system developed in this study consists of a series of eight programs as outlined in figure 14 and listed in appendices A through H. Each program is structurally designed and documentation on the programs and subroutines is given in this chapter. Although most of the procedures are self explanatory, two procedures are documented separately in sections 3.1 and 3.2 of this chapter.

3.1 THE INDEX STRUCTURES

The system programs access the random access files by storing the key elements and file record numbers in an index structured as a binary tree. The index structures are stored externally as sequential files and internally as n by four arrays. The four entries per element represent the left pointer, key element, right pointer, and the random access file record number corresponding to the key element. Figure 15 gives the internal matrix representation of a random selection of class numbers in terms of both the matrix and the abstract binary tree representation. It is possible to search the index to find a given key by following pointers in the array structure. Figure 16 gives the BASIC program for searching the matrix representation of a binary tree.

3.2 THE BALANCE PROCEDURE

The advantage of using a binary tree structure to store an index is a reduction in time required to search the index. The time required to search a binary tree is directly proportional to the length of the tree. Since it is possible to construct various binary trees using the same nodes, as illustrated by figure 17, it is important to optimize the structure of the tree by minimizing its length. The student information system developed in this study contains a tree balancing procedure which produces the optimal tree structure, a balanced binary tree. The procedure, included within the BUILD program listed in appendix III, utilizes the in-order, length, and position pointers subroutines and the algorithm for the procedure is documented below. Figure 18 further documents the balance algorithm by presenting the step by step calculations peformed when the index given in figure 17(a) is balanced to produce the structure presented in figure 17(b).

STEP 1: Traverse the tree structure in left-node-right order

STEP 1: Traverse the tree structure in left-node-right order and store the row numbers in a vector ODR.

STEP 2: Calculate the maximum length of the largest full balanced tree that can be created with the given nodes. The maximum full balanced tree size, number of elements, will be a positive integral power of two minus one.

STEP 3: Determine the number of elements to temporarily delete from the ODR vector, store the first odd elements in a vector OMIT and pack ODR.

STEP 4: Create a full balanced tree by inserting appropriate pointers into the structure.

STEP 4a: Set the left and right pointers of all odd numbered nodes to zero. That is nodes ODR(1),ODR(3),.....,ODR(2i-1) left and right pointers are set to zero.

STEP 4b: Place pointers in the index matrix in a number of passes as outlined below.

```
PASS 1.-n = 2.
       -left pointer of node ODR(n) = node ODR(n-1).
       -right pointer of node ODR(n) = node ODR(n+1).
       -n = n + 4.
       -repeat until n is greater than the length
```

```
            of packed ODR.
    PASS 2.-n = 4.
           -left pointer of node ODR(n) = node ODR(n-2).
           -right pointer of node ODR(n) = node ODR(n+2).
           -n = n + 8.
           -repeat until n is greater than the length of
            of packed ODR.
    PASS I: where 2 to the power I is less than the length
            of packed ODR.
           -n = 2 to the power I.
           -K = n + 1.
           -left pointer of node ODR(n) = node n - 2 ^ (i-1).
           -right pointer of node ODR(n) = node n + 2 ^ (i+1).
           -n = n + 2 to the power K.
           -repeat until n is greater than the length of
            packed ODR.

STEP 5:  Place  the  omited nodes into the tree by extending
               the lower left nodes.

           -left pointer of node ODR(1) = node omit(1)
           -right pointer of node ODR(1) = node OMIT(2).
           -left pointer of node ODR(2) = node OMIT(3).
           -continue until all nodes deleted in step 3 are
            inserted.
```

STEP 6: Set both left and right pointers of the inserted nodes to zero.

STEP 7: Define the root node. The root node will be node ODR(k), where k = (length of packed ODR + 1) divided by two.

3.3 THE SYSTEM PROGRAMS

The eight programs of the student information system are documented in this section with all subroutines documented in section 3.4. Figure 14 outlines the information system in terms of the programs accessed by the system and figure 19 gives a description of the fundamental variables used in the system software.

3.3.1 PROGRAM SYSTEM MENU: This program, listed in appendix A, controls the system software. The menu, which consists of the system programs, appears on the screen when the system is booted and the user selects which program to execute.

3.3.2 PROGRAM INITIALIZE FILES: This program, listed in appendix B, initializes the random access files as outlined in figures 11, 12, and 13. All fields are set equal to zero to allow for error processing. The index files are opened and the first three fields of each index file are set to zero. These fields contain the root node of the binary index, the number of elements in the index, and the number of elements in the index contained when is was last balanced.

3.3.3 PROGRAM BUILD SCHEDULES: This program, listed in appendix C, builds the student and teacher schedule files. The program accepts the schedules from the user, enters the key element into the appropriate index, and writes the schedule to the correct file and record. The class index is also maintained by this program and the class information file is updated as a given teacher or student is added to the data base. Figure 20 gives the structural design of the build schedules program.

3.3.4 PROGRAM RETRIEVE SCHEDULES: This program, outlined in figure 20 and listed in appendix D, allows the user to print one student's schedule, one teacher's schedule, all student schedules, or all teacher schedules. If the user chooses to print all schedules the schedules will be produced in order of increasing student or teacher number.

3.3.5 PROGRAM UPDATE: This program allows the user to make changes in the student schedules, student's grades and teacher comments, teacher schedules, or room numbers. The design, by subroutines, is outlined in figure 22 and the program listing is given in appendix E.

3.3.6 PROGRAM ENTER GRADES: This program asks the user if end of semester or mid-semester grades and comments are to be entered, produces class lists, inputs the grades and comments, and writes the student grades and comments to the student schedule file. Figure 23 gives the structural design of this program and a program listing of the enter grades program is given in appendix F.

3.3.7 PROGRAM PRINT GRADES: A program which prints either the mid-semester or the end of semester grade reports. The program will produce grade reports for an individual student, or all grade reports will be produced in order of increasing student number. The structural design of the print grades program is given in figure 24 and a listing of the program is included in appendix G.

3.3.8 PROGRAM RETRIEVE CLASS LISTS: This program allows the user to retrieve class lists. The user can access lists for one class, all classes for one teacher, or all classes that exist within the data base. Figure 25 gives the structural design of this program and the program listing is given in appendix H.

3.4 SUBROUTINE DOCUMENTATION

The various subroutines accessed by the programs of the student information system are documented in terms of the function performed by each subroutine. The subroutines are standard in design and the documentation given presents an overview of the purpose of the subroutine rather than a detailed description of the algorithms employed.

3.4.1 SUBROUTINE READ TEACHER INDEX: A subroutine which reads the teacher index from the sequential teacher index file and internally stores the index as the TINDX array.

3.4.2 SUBROUTINE READ STUDENT INDEX: A subroutine which reads the student index from the sequential student index file and stores the index internally as the SNDX array.

3.4.3 SUBROUTINE READ CLASS INDEX: A subroutine which reads the class index from the sequential class index file and internally stores the index in the CINDX array.

3.4.4 SUBROUTINE CHANGE SCHEDULE: This subroutine allows the user to make changes in the schedule during the input process. This subroutine calls the display schedule, change number, and change classes subroutines.

3.4.5 SUBROUTINE DISPLAY SCHEDULE: This subroutine is called by the change schedule subroutine and displays the current teacher or student schedule.

3.4.6 SUBROUTINE CHANGE NUMBER: This subroutine allows the user to correct the teacher or student number before the number is stored in the appropriate file.

3.4.7 SUBROUTINE CHANGE CLASSES: This subroutine allows the user to change the classes, by period, in the schedule before it is stored in the appropriate file.

3.4.8 SUBROUTINE FIND STUDENT RECORD: This subroutine searches the student index and inserts the given student into the student index structure. The corresponding record number of the student schedule file is also recorded in the index.

3.4.9 SUBROUTINE WRITE STUDENT SCHEDULE: A subroutine that writes the given student's schedule to the appropriate record of the student schedule file.

3.4.10 SUBROUTINE UPDATE CLASS INDEX: This subroutine searches the class index and returns the corresponding record number of the class information file if a given course exists in the index. If the course does not exist the class is added to the index and the course and period numbers are printed to the appropriate record of the class information file.

3.4.11 SUBROUTINE REVISE CLASS INFO (STUDENT): A subroutine which increments the class size and adds the given student to the class list.

3.4.12 SUBROUTINE REVISE CLASS INFO (TEACHER): A subroutine which adds the given teacher to the appropriate record of the class information file.

3.4.13 SUBROUTINE EXIT: This subroutine controls the exit process. All index structures are balanced if more than five elements have been added to the index since it was last balanced.

3.4.14 SUBROUTINE BALANCE INDEX: This subroutine controls the balance procedure as outlined in section 3.2 of this chapter by calling the in order, length, and the position pointers subroutines.

3.4.15 SUBROUTINE IN-ORDER: A subroutine to traverse the binary tree index structure in (left-node-right) order and produce the vector ODR. The vector ODR contains pointers to the node elements of the index tree to allow processing in order of increasing student, teacher, or course numbers.

3.4.16 SUBROUTINE POSITION POINTERS: This subroutine uses the ODR vector formulated by the in-order subroutine to build a balanced tree structure. This subroutine uses the algorithm presented in section 3.2.

3.4.17 SUBROUTINE LENGTH: This subroutine calculates the size of the largest balanced tree that can be constructed.

3.4.18 SUBROUTINE WRITE TEACHER INDEX: This subroutine writes the teacher index to the teacher index file.

3.4.19 SUBROUTINE WRITE CLASS INDEX: This subroutine writes the teacher index to the class index file.

3.4.20 SUBROUTINE WRITE STUDENT INDEX: This subroutine writes the student index to the student index file.

3.4.21 SUBROUTINE RETRIEVE STUDENT SCHEDULE (ONE STUDENT): This subroutine reads the student schedule for a given student from the student schedule file.

3.4.22 SUBROUTINE FIND CLASS RECORD: A subroutine which searches the class index and returns the corresponding record number of the class information file.

3.4.23 SUBROUTINE READ TEACHER-ROOM NUMBERS: A subroutine which reads the teacher and room number from the appropriate record of the class information file.

3.4.24 SUBROUTINE PRINT STUDENT SCHEDULE: A subroutine to display the student schedule for a given student. The teacher and room numbers are also displayed.

3.4.25 SUBROUTINE RETRIEVE TEACHER SCHEDULE (ONE TEACHER): This subroutine reads the teacher schedule for a given teacher from the teacher schedule file.

3.4.26 SUBROUTINE RETRIEVE STUDENT SCHEDULES: A subroutine to control the processing of all student schedules. The subroutine first calls subroutine in order and then processes each student's schedule by reading the schedule and calling the find class record, read teacher-room numbers, and print student schedule subroutines.

3.4.27 SUBROUTINE RETRIEVE TEACHER SCHEDULES: A subroutine to control the processing of all teacher schedules. The subroutine calls subroutine in-order to produce a list of all teachers in order of increasing teacher number. The class schedule for each teacher is then processed by calling subroutines read teacher schedule, read teacher-room numbers, and print teacher schedule.

3.4.28 SUBROUTINE PUSH STACK: This subroutine is used by subroutine in-order. The current node and a direction (D = 1 represents a left pointer and D = 2 represents a right pointer) are pushed onto the stack.

3.4.29 SUBROUTINE POP STACK: This subroutine returns the node and direction from the top of the stack.

3.4.30 SUBROUTINE PRINT TEACHER SCHEDULE: A subroutine to print the teacher schedule and room numbers for a given teacher.

3.4.31 SUBROUTINE BYTE POINTERS MID-SEMESTER: A subroutine to set B1 and B2 to ten and thirteen respectively to allow access to the mid-semester grades and comments.

3.4.32 SUBROUTINE BYTE POINTERS FINIAL: A subroutine to set B1 and B2 to fifteen and eighteen respectively to allow access to the end of semester grades and comments.

3.4.33 SUBROUTINE FIND TEACHER RECORD: A subroutine to search the teacher index for a given teacher and return the record number of the teacher schedule file where the teacher's schedule is stored.

3.4.34 SUBROUTINE READ TEACHER SCHEDULE: A subroutine which reads a teacher's schedule from the teacher schedule file.

3.4.35 SUBROUTINE READ CLASS LIST: This subroutine reads the class list of a given class. The student numbers of all students enrolled in the class are stored as the vector LST.

3.4.36 SUBROUTINE INPUT AND WRITE GRADES: This subroutine inputs student grades and comments and writes the grades and comments to the student schedule file.

3.4.37 SUBROUTINE UPDATE STUDENT SCHEDULE: This subroutine updates the student schedule stored on the student schedule file and corrects the corresponding class size and class lists in the class information file.

3.4.38 SUBROUTINE MISSING PERSON: This subroutine displays a message that the required teacher, or student, is not currently defined in the data base.

3.4.39 SUBROUTINE UPDATE TEACHER SCHEDULE: This subroutine updates the teacher's schedule in the teacher schedule file and adds the teacher number to appropriate record of the class information file.

3.4.40 SUBROUTINE UPDATE GRADES: A subroutine to find the record number of the student schedule file for a given student, display the current grades, input corrected grades and comments, and store the corrected grades and comments in the student schedule file.

3.4.41 SUBROUTINE SET BYTE POINTERS: B1, B2, B3, and B4 are set to ten, thirteen, fifteen, and eighteen respectively to allow access to the mid-semester and end of semester grades and comments.

3.4.42 SUBROUTINE PRINT GRADES (ONE STUDENT): A subroutine to control inputing and processing of the grades for an individual student. The subroutine calls subroutines find student record, read grades, and returns control to the program menu.

3.4.43 SUBROUTINE READ GRADES: A subroutine to read the student schedule and existing grades from the student schedule file.

3.4.44 SUBROUTINE PRINT MID-SEMESTER GRADES: A subroutine to print the schedule, teacher numbers, room numbers, and mid-semester grades.

3.4.45 SUBROUTINE PRINT FINIAL GRADES: A subroutine to print the student schedule, teacher number, room number, and end of semester grades and comments.

3.4.46 SUBROUTINE INPUT CLASS NUMBER: A subroutine which inputs a class number and searches the class index to find the record number of the class information file corresponding to the given class.

3.4.47 SUBROUTINE READ CLASS DATA: A subroutine to read the class number, teacher number, period number, class size, and class list from the class information file.

3.4.48 SUBROUTINE PRINT CLASS LIST: A subroutine to display the teacher number, period number, room number, class size, and the student numbers of all students currently enrolled in a given class.

3.4.49 SUBROUTINE PROCESS LISTS BY TEACHER: A subroutine which inputs a teacher number, finds the corresponding record number of the teacher schedule file, and reads the teacher schedule from the file. This subroutine then calls the find class record, read class record, read class data, and print class list subroutines for each class.

3.4.50 SUBROUTINE PROCESS ALL CLASS LISTS: A subroutine to print all class lists. The subroutine calls subroutine in-order to produce a list of all teachers in order of increasing teacher number. The subroutine then reads each teacher's schedule and calls subroutines find class record, read class record, read class data, and print class lists for all classes defined in the data base.

3.4.51 SUBROUTINE CHANGE CLASS INDEX: A subroutine to copy the CINDX array into the INDX array to enable the balance routine to balance the class index structure.

3.4.52 SUBROUTINE PRINT GRADES (ALL STUDENTS): A subroutine to control the processing of all student grades. The subroutine first calls subroutine in-order to produce a list of all students in order of increasing student number. Subroutines read student grades and print student grades are then called for all students contained in the data base.

3.4.53 SUBROUTINE UPDATE ROOM NUMBERS: A subroutine that inputs a class number, finds the existing room number, inputs a new room number and updates the class information file.

Program
SYSTEM MENU

programs

-Initialize files

-Build Schedules

-Retrieve Schedules

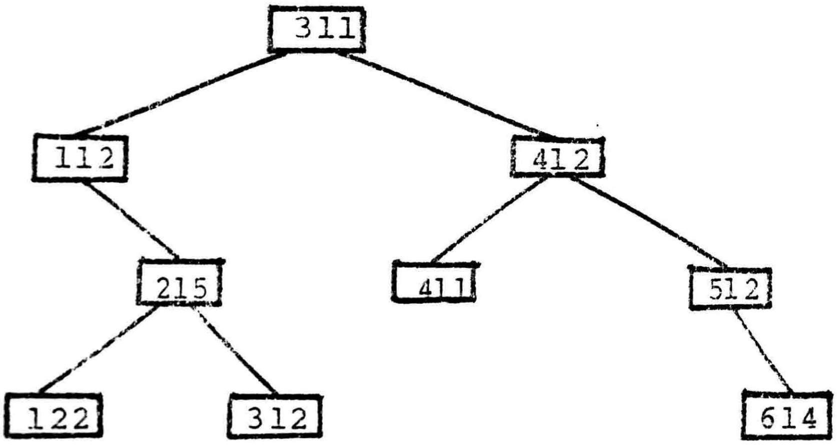
-Retrieve Class Lists

-Enter Grades

-Print Grades

-Update

Figure 14: Programs of the Student Information System



15-a Class-Index as a binary tree

Left pointer	Class Number	Right Pointer	Record Number
4	311	2	0
8	412	3	1
0	512	6	2
0	112	5	3
7	215	9	4
0	614	0	5
0	122	0	6
9	411	0	7
0	312	0	8

15-b Class-Index as an Array

Figure 15: Index Structure as a Binary Tree and an Array

```
10  REM PROCEDURE TO SEARCH BINARY TREE INDEX
20  PT = SP
30  IF PT = 0 THEN GOTO 110
40  IF NUMB < INDEX(PT,2) THEN IK = 1
50  IF NUMB > INDEX(PT,2) THEN IK = 3
60  IF NUMB = INDEX(PT,2) THEN GOTO 90
70  PT = INDEX(PT,IK)
80  GOTO 30
90  REC = INDEX(PT,4)
100 GOTO 120
110 PRINT "CLASS IS NOT IN THE INDEX"
120 RETURN
```

Line 20 Initializes the pointer PT to SP,the root of node of the index tree.

Line 30 Tests to determine if the current node is a terminal node of the tree. If the current node is a terminal node NUMB is not contained in the index.

Line 40 If NUMB is less than the current node set IK, a temporary pointer, to 1.

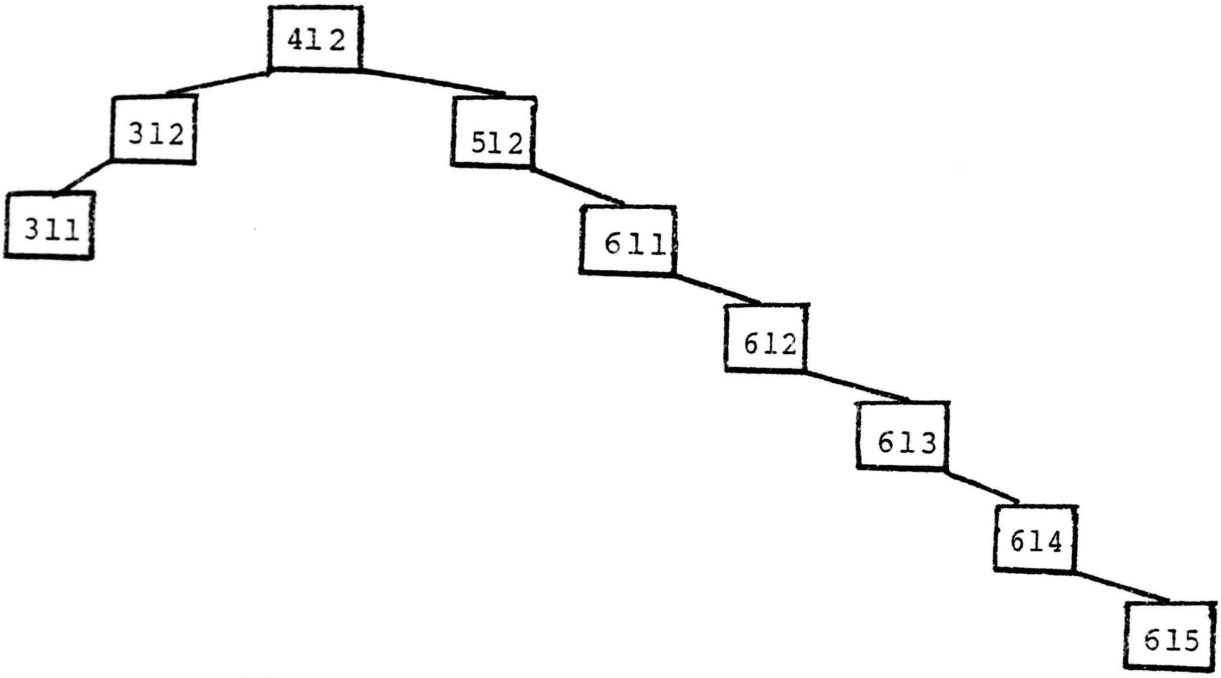
Line 50 If NUMB is greater than the current node set the temporary pointer to 3.

Line 60 If NUMB is equal to the current node the record has been found.

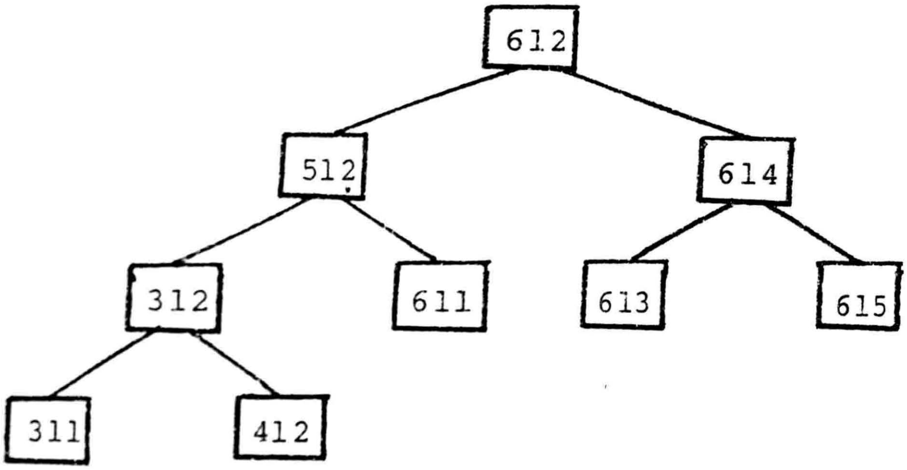
Line 70 Move down the tree by setting the pointer PT equal to the left or right pointer as determined above.

Line 80 Repeat until NUMB is located or a terminal node is located.

Figure 16: Basic code for searching an index structure



17-a Unbalanced Binary Tree Structure



17-b Balanced Binary Tree Structure

Figure 17: Binary Tree Structures

```
Step 1: Traverse in order and create the ODR vector:
          ODR(3 , 2 , 1 , 4 , 5 , 6 , 7 , 8 , 9)
Step 2:   Calculate the maximum length of the full tree.
          LF = 7 = (2 ^ 3 - 1)
Step 3:   Determine the number of elements of delete.
          ND = 2 = (length of index - length of full tree)
        Create the OMIT vector and pack ODR.
        OMIT(1) = ODR(1) = 3
        OMIT(2) = ODR(3) = 1
        Pack ODR.
        ODR(2 , 4 , 5 , 6 , 7 , 8 , 9)

Step 4a: Set all odd pointers to zero.
         INDEX(ODR(I),1) = INDEX(ODR(I),3) = 0; I odd.

Step 4b:  Position pointers in following passes:
 pass 1:  INDEX(ODR(2),1) = ODR(1)...INDEX(4,1) = 2
          INDEX(ODR(2),3) = ODR(3)...INDEX(4,3) = 5
          INDEX(ODR(6),1) = ODR(5)...INDEX(8,1) = 7
          INDEX(ODR(6),3) = ODR(7)...INDEX(8,3) = 9
 pass 2:  INDEX(ODR(4),1) = ODR(2)...INDEX(6,1) = 4
          INDEX(ODR(4),3) = ODR(6)...INDEX(6,3) = 8
Step 5:   Place omitted nodes into the tree.
          INDEX(ODR(1),1) = OMIT(1)...INDEX(2,1) = 3
          INDEX(ODR(1),3) = OMIT(2)...INDEX(2,3) = 1
Step 6:   Set pointers of the inserted nodes to zero.
          INDEX(OMIT(1),1) = 0.......INDEX(3,1) = 0
          INDEX(OMIT(1),1) = 0.......INDEX(3,3) = 0
          INDEX(OMIT(2),2) = 0.......INDEX(1,1) = 0
          INDEX(OMIT(2),3) = 0.......INDEX(1,3) = 0
Step 7:   Define the root node:
          Root node = ODR((7 + 1)/ 2) = ODR(4) = 6
```

Unbalanced index.

2	412	4	0
0	312	3	1
0	311	0	2
0	512	5	3
0	611	6	4
0	612	7	5
0	613	8	6
0	614	9	7
0	615	0	8

Balanced index.

0	412	0	0
2	312	1	1
0	311	0	2
2	512	5	3
0	611	0	4
4	612	8	5
0	613	0	6
7	614	9	7
0	615	0	8

Figure 18: Balance Routine

Variable	Description
INDX(500,4)	Array to internally store the student index structure.
STACK(256,2)	Array used as a stack by the in-order subroutine.
OMIT(256)	Vector to temporarily store nodes deleted in the balance procedure.
ODR(256)	Vector to store the pointers to the nodes in increasing order.
SCHD(7)	Vector to internally store the student or teacher schedule.
CINDX(150,4)	Array to internally store the class index structure.
SNDX(500,4)	Array to internally store the student index structure.
TINDX(50,4)	Array to internally store the teacher index structure.
CREC(7)	Vector to store the record numbers of the class information file.
TCH(7)	Vector to store the teacher schedule.
GMT(7)	Vector to store the mid-semester grades.
GF(7)	Vector to store the end of semester grades.
CMT(7)	Vector to store the mid-semester teacher comments.
CF(7)	Vector to store the end of semester comments.
INFO(5)	Vector to store class information: course number, teacher, period number room number, and class size.
LST(35)	Vector to store class lists.
GDE(35)	Vector to store grades for a class.
COM(35)	Vector to store comments for a class.

Figure 19: Variable Documentation

```
Program                 Subroutines

BUILD SCHEDULES:  -Read Student Index       (3.4.2)
                  -Read Teacher Index       (3.4.1)
                  -Read Class Index         (3.4.3)
                  -Change Schedule          (3.4.4)
                           -Display Schedule
                                            (3.4.5)
                           -Change Number   (3.4.6)
                                  -Display Schedule
                                            (3.4.5)
                           -Change Classes  (3.4.7)
                                  -Display Schedule
                                            (3.4.5)
                  -Find Student Record      (3.4.8)

                  -Write Student Schedule   (3.4.9)

                           -Update Class Index
                                            (3.4.10)
                                  -Revise Class Info(student)
                                            (3.4.11)
                  -Write Teacher Schedule   (3.4.30)
                          -Update Class Index
                                            (3.4.10)
                                  -Revise Class Info(teacher)
                                            (3.4.12)
                  -Balance Index            (3.4.14)
                                 -In-Order  (3.4.15)
                                 -Push Stack(3.4.28)
                                 -Pop Stack (3.4.29)
                          -Length           (3.4.17)

                          -Position Pointers
                                            (3.4.16)
                  -Write Class Index        (3.4.19)

                  -Write Teacher Index      (3.4.18)

                  -Write Student Index      (3.4.20)

                          -Change Class Index
                                            (3.4.19)
```

Figure 20: Program Build Schedules (Structure)

```
Program:                         Subroutines
Retrieve Schedules:
                    -Read Student Index (3.4.2)
                    -Read Teacher Index (3.4.1)
                    -Read Class Index   (3.4.3)

                    -Retrieve Student Schedule
                        (one student)   (3.4.21)
                              -Find Class Record (3.4.22)

                              -Read Teacher/Room number
                                                 (3.4.23)
                              -Print Student Schedule
                                                 (3.4.24)
                    -Retrieve Teacher Schedule
                         (one teacher)    (3.4.25)
                              -Find Class Record (3.4.22)

                              -Read Teacher/Room number
                                                 (3.4.23)
                              -Print Teacher Schedule
                                                 (3.4.30)
                    -Retrieve Student Schedules
                                       (3.4.26)
                              -In Order (3.4.15)
                                     -Push Stack (3.4.28)
                                     -Pop Stack  (3.4.29)
                              -Find Class Record (3.4.22)

                              -Read Teacher/Room number
                                                 (3.4.23)
                              -Print Student Schedule
                                                 (3.4.24)
                    -Retrieve Teacher Schedules.
                                       (2.4.27)
                              -In Order (3.8.15)
                                     -Push Stack (3.4.28)
                                     -Pop Stack  (3.4.29)
                              -Find Class Record (3.4.22)

                              -Print Teacher Schedule
                                                 (3.4.30)
```

Figure 21: Program Retrieve Schedules (Structure)

```
Program:                Subroutines

Update            -Read Student Index    (3.4.2)

                  -Read Class Index      (3.4.3)

                  -Read Teacher Index    (3.4.1)

                  -Update Student Schedule
                                         (3.4.37)

                           -Missing Person
                                         (3.4.38)

                           -Find Class Record
                                         (3.4.22)

                  -Update Teacher Schedule
                                         (3.4.39)

                           -Missing Person
                                         (3.4.38)

                           -Find Class Record
                                         (3.4.22)

                  -Update Grades         (3.4.40)

                  -Update Room numbers   (3.4.53)
```

Figure 22: Program Update (Structure)

Program	Subroutines	
Enter Grades:		
	-Read Teacher Index	(3.4.1)
	-Read Class Index	(3.4.3)
	-Read Student Index	(3.4.2)
	-Byte Pointers mid-semester	(3.4.31)
	-Byte Pointers finial	(3.4.32)
	-Find Teacher Record	(3.4.33)
	-Read Teacher Schedule	(3.4.34)
	-Find Class Record	(3.4.42)
	-Find Student Records	(3.4.8)
	-Input and write grades	(3.4.36)

Figure 23: Program Enter Grades (Structure)

```
Program                           Subroutines

Print Grades:
                        -Read Student Index   (3.4.2)

                        -Read Class Index     (3.4.3)

                        -Set Byte Pointers    (3.4.41)

                        -Print Grades (one student)
                                              (3.4.42)

                                -Find Student record  (3.4.8)

                                -Print mid-semester grades
                                                      (3.4.44)

                                -Print finial grades  (3.4.45)

                        -Print grades (all students)
                                              (3.4.52)
                                -In Order  (3.4.15)
                                         -Push Stack (3.4.28)
                                         -Pop Stack  (3.4.29)

                                -Read Grades          (3.4.43)

                                -Print mid-semester grades
                                                      (3.4.44)
                                -Print finial grades  (3.4.45)
```

Figure 24: Program Retrieve Grades (Structure)

```
Program                          Subroutines

Retrieve Class Lists:
                         -Read Teacher Index    (3.4.1)

                         -Read Class Index      (3.4.3)

                         -Input Class Number    (3.4.46)

                                -Find Class Record
                                                (3.4.22)
                                -Read Class Data
                                                (3.4.47)
                                -Print Class Lists
                                                (3.4.48)

                         -Process All Class Lists
                                                (3.4.50)

                                 -In Order      (3.4.15)

                                         -Push Stack
                                                (3.4.28)
                                         -Pop Stack
                                                (3.4.29)

                                 -Find Class Record
                                                (3.4.22)

                                 -Read Class Data
                                                (3.4.47)

                                 -Print Class List
                                                (3.4.48)
```

Figure 25: Program Retrieve Class Lists (Structure)

CHAPTER 1V

CONCLUSION AND REMARKS

The purpose of this study was to investigate the feasibility of using microcomputers to maintain a student information system to serve a small high school. The memory limitations, both core and direct access, and the time required to retrieve information where the primary concerns in the system's design and implementation. To discuss the results of the study it is first necessary to discuss the specifications of the specific micro computer used in the implementation of the system developed. The system developed in this study was implemented on an Apple II PLUS micro computer with 48K random access memory and external storage in the form of five and a quarter inch magnetic floppy diskettes. The 48K RAM proved to be sufficient to execute the system software and the maximum number of students the system can maintain is dependent upon the external storage limitations. To analysis the maximum capacity of the system we must first analysis the storage capacity of a floppy diskette when initialized on an Apple II micro computer.

When the Apple's disk operating system initializes a diskette, the diskette is divided into thirty five tracks with each track containing sixteen sectors. Each sector will store two hundred and fifty six bytes of data and four tracks are reserved to store the disk operating system and disk directory. This leaves thirty one tracks or four hundred and ninety six sectors available to the user to store software or data files. The number of sectors required to store a file is given in the catalog listing of the directory of the diskette. The catalog listing of the diskette which stores the system software for the system developed in this study is given in figure 26 and shows that a total of one hundred and seventy three sectors are required to store the system software. This leaves three hundred and twenty three sectors available to store the student schedule, teacher schedule, class information, and the three index files. Figure 27(a) gives the disk directory when the six data files contain the data for three hundred students, fifty teachers, and one hundred and twenty classes. As figure 27(a) illustrates, the system developed in this study will maintain a student information system for three hundred students when implemented on an Apple II micro computer with 48K RAM and a single disk drive. Further analysis of the unused forty eight sectors on the disk imply that the student information system designed in this study can accommodate a maximum of three hundred and fifty students when implemented on a one disk drive Apple II PLUS micro computer configuration.

The addition of a second disk drive greatly increases the capacity of the system. If the system software is stored on diskette number one, operating in drive one, and the data files are stored on diskette number two, operating in drive two, the maximum number of students increases to over five hundred. Figure 27(b) shows the catalog listing when the data files contain the data for five hundred students, fifty teachers, and one hundred and fifty courses. A total of three hundred and ninety six sectors are required to store the data files which leaves one hundred unused sectors. Further analysis of the available space indicates that a two disk drive Apple II PLUS with 48K RAM hardware configuration, will maintain a data base for six hundred and fifty students. An added advantage of using a dual disk drive system is that the three hundred and twenty three unused sectors on diskette number one could be utilized to store the student and teacher name relations.

Three hundred and twenty three sectors represents approximately 82K bytes of storage. If each name is restricted to thirty characters, each record (number,name) would require thirty seven bytes when end of field and record markers are included. Since the analysis above gives the maximum size of the system as six hundred and fifty students with sixty five teachers, the name relations would require a total of approximately 30K bytes. Thus diskette number one would accommodate the files required to store the name relations. The addition of these relations would allow student schedules, teacher schedules, and student grade reports to be printed by name.

Although the above analysis of the direct access storage capabilities of the Apple II indicate that a maximum of six hundred and fifty students could be maintained with this system, problems do occur with the software. The system, as designed, will accommodate a maximum of five hundred students since the program update files, listed in appendix V, utilizes the complete 48K of random access memory. The system, as presented in this study, would have to be slightly modified to accommodate more than five hundred students.

In conclusion, this investigation did find that the microcomputer is a viable alternative for maintaining a small stand-alone data base. It is indeed feasible for a school, enrollment around five hundred students, to consider implementing its student information system on a dual disk drive micro computer hardware configuration. The student information system presented in this study will maintain the data base required for five hundred students and slight modifications of the system design would increase the maximum number of students to six hundred and fifty. This study further indicates that student and teacher name relations could be added to the system to allow for schedule and grade reports to be printed by teacher and student name. Further modifications should be made to the system software before it is implemented. The system contains all the information needed to assist in the scheduling process of a school. The system could easily be modified to allow student and teacher classes to be entered without regard to period. The system could supply the period and room numbers and report schedule conflicts.

```
A 036 BUILD SCHEDULES
A 020 RETRIEVE CLASS LISTS
A 005 MENU
A 026 RETRIEVE SCHEDULES
A 020 ENTER GRADES
A 024 PRINT GRADES
A 036 UPDATE FILES
A 006 INITIALIZE DATA FILES
```

Total of 173 sectors required to store the system software.

Figure 26: Storage requirement for the system software

```
T 025 STUDENT-INDEX
T 005 TEACHER-INDEX
T 008 CLASS-INDEX
T 008 TEACHER-SCHEDULE
T 106 CLASS-INFORMATION
T 123 STUDENT-SCHEDULE
```

27-a Files initialized for 300 students (275 sectors)

```
T 033 STUDENT-INDEX
T 005 TEACHER-INDEX
T 010 CLASS-INDEX
T 008 TEACHER-SCHEDULES
T 134 CLASS-INFORMATION
T 206 STUDENT-SCHEDULE
```

27-b Files initialized for 500 students (396 sectors)

Figure 27: Data files storage requirements

APPENDIX A

PROGRAM SYSTEM MENU

```
100  HOME : PRINT : PRINT
110  D$ =  CHR$ (4)
120  PRINT  TAB( 7)"STUDENT  INFORMATION  SYSTEM"
130  PRINT
140  PRINT  TAB( 15)"CREATED BY"
150  PRINT
160  PRINT  TAB( 13)"DOUG  WAECHTER"
170  PRINT : PRINT : PRINT
180  PRINT "  MENU"
190  PRINT : PRINT
200  PRINT "    1.....BUILD SCHEDULES"
210  PRINT "    2.....RETRIEVE SCHEDULES"
220  PRINT "    3.....RETRIEVE CLASS LISTS"
230  PRINT "    4.....ENTER GRADES"
240  PRINT "    5.....PRINT GRADES"
250  PRINT "    6.....UPDATE FILES"
260  PRINT "    7.....EXIT SYSTEM"
270  PRINT : PRINT
280  INPUT "ENTER THE NUMBER OF YOUR CHOICE   ";ANS
290  IF ANS < 1 OR ANS > 7 THEN  GOTO 100
300  HOME : PRINT : PRINT
310  PRINT  TAB( 13)"LOADING PROGRAM"
320  PRINT : PRINT
330  PRINT  TAB( 14)"PLEASE WAIT"
340  IF ANS = 1 THEN  PRINT D$;"RUN BUILD SCHEDULES"
350  IF ANS = 2 THEN  PRINT D$;"RUN RETRIEVE SCHEDULES"
360  IF ANS = 3 THEN  PRINT D$;"RUN RETRIEVE CLASS LISTS"
370  IF ANS = 4 THEN  PRINT D$;"RUN ENTER GRADES"
380  IF ANS = 5 THEN  PRINT D$;"RUN PRINT GRADES"
390  IF ANS = 6 THEN  PRINT D$;"RUN UPDATE FILES"
400  IF ANS = 7 THEN  GOTO 420
410  GOTO 100
420  HOME
430  END
```

APPENDIX B

PROGRAM INITIALIZE FILES

```
100  REM  PROGRAM TO INITIALIZE FILES.
110  REM
120  D$ =  CHR$ (4)
130  PRINT D$;"OPEN STUDENT-INDEX"
140  PRINT D$;"WRITE STUDENT-INDEX"
150  PRINT 0: PRINT 0: PRINT 0
160  PRINT D$;"OPEN TEACHER-INDEX"
170  PRINT D$;"WRITE TEACHER-INDEX"
180  PRINT 0: PRINT 0: PRINT 0
190  PRINT D$;"OPEN CLASS-INDEX"
200  PRINT D$;"WRITE CLASS-INDEX"
210  PRINT 0: PRINT 0: PRINT 0
220  PRINT D$;"CLOSE"
230  PRINT D$;"OPEN TEACHER-SCHEDULE,L31"
240  FOR I = 1 TO 65
250  I1 = I - 1
260  PRINT D$;"WRITE TEACHER-SCHEDULE,R";I1;",B0
270  PRINT 0
280  FOR K = 3 TO 27 STEP 4
290  PRINT D$;"WRITE TEACHER-SCHEDULE,R";I1;",B";K
300  PRINT 0
310  NEXT K
320  NEXT I
330  PRINT D$;"CLOSE"
340  REM
350  PRINT D$;"OPEN CLASS-INFO,L225"
360  FOR I = 1 TO 150
370  I1 = I - 1
380  PRINT D$;"WRITE CLASS-INFO,R";I1;",B0"
390  PRINT 0
400  PRINT D$;"WRITE CLASS-INFO,R";I1;",B4"
410  PRINT 0
420  PRINT D$;"WRITE CLASS-INFO,R";I1;",B7"
430  PRINT 0
440  PRINT D$;"WRITE CLASS-INFO,R";I1;",B9"
450  PRINT 0
460  PRINT D$;"WRITE CLASS-INFO,R";I1;",B12"
470  PRINT 0
```

```
480  NEXT I
490  PRINT D$;"CLOSE"
500  D$ =  CHR$ (4)
510  PRINT D$;"OPEN STUDENT-SCHEDULE,L104"
520  FOR I = 1 TO 500
530  BYT = 0
540  GOSUB 700
550  FOR K = 1 TO 7
560  BYT = 6 + 14 * (K - 1)
570  GOSUB 700
580  BYT = 10 + 14 * (K - 1)
590  GOSUB 700
600  BYT = 13 + 14 * (K - 1)
610  GOSUB 700
620  BYT = 15 + 14 * (K - 1)
630  GOSUB 700
640  BYT = 18 + 14 * (K - 1)
650  GOSUB 700
660  NEXT K
670  NEXT I
680  PRINT D$;"CLOSE STUDENT-SCHEDULE"
690  END
700  REM  SUBROUTINE
710  PRINT D$;"WRITE STUDENT-SCHEDULE,R";I - 1;",B";BYT
720  PRINT 0
730  RETURN
```

APPENDIX C

PROGRAM BUILD SCHEDULES

```
100  REM  PROGRAM BUILD-SCHEDULES
110  REM
120  DIM INDXS(500,4)
130  DIM STACK(400,2)
140  DIM OMIT(256)
150  DIM ODR(500)
160  DIM SCHD(7)
170  DIM CINDX(150,4)
180  DS =  CHR$ (4)
190  HOME : PRINT : PRINT
200  PRINT "   THIS PROGRAM BUILDS BOTH"
210  PRINT
220  PRINT "THE STUDENT-SCHEDULE AND THE "
230  PRINT : PRINT "   THE TEACHER-SCHEDULE FILES."
240  PRINT : PRINT : PRINT "ENTER THE NUMBER OF YOUR CHOICE"
250  PRINT : PRINT : PRINT
260  PRINT "1..FOR STUDENT-SCHEDULE.."
270  PRINT : PRINT
280  PRINT "2..FOR TEACHER-SCHEDULE.."
290  PRINT : PRINT : INPUT ANS
300  GOSUB 1470
310  IF ANS = 1 THEN  GOSUB 1190
320  IF ANS = 2 THEN  GOSUB 1340
330  IF ANS <  > 1 AND ANS <  > 2 THEN  GOTO 190
340  IF ANS = 1 THEN N$ = "STUDENT"
350  IF ANS = 2 THEN N$ = "TEACHER"
360  GOSUB 1470
370  HOME : PRINT : PRINT
380  PRINT "THIS PROGRAM BUILDS THE ";N$;"-SCHEDULE";
385  PRINT "  AND THE ";N$;"-INDEX FILES."
390  PRINT : PRINT : PRINT
400  PRINT "YOU CAN TERMINATE THIS PROGRAM AT ANY"
410  PRINT
420  PRINT "TIME AND THE DATA WILL BE SAVED"
430  PRINT : PRINT : PRINT
440  PRINT "SIMPLY PRESS THE 'E' KEY AT THE PROMPT"
450  PRINT : PRINT : PRINT
460  PRINT "      <PRESS 'E'  TO EXIT>      "
```

```
470  PRINT : PRINT : PRINT
480  PRINT "  PRESS THE RETURN KEY TO CONTINUE   ":INPUT Z$
490  HOME : PRINT : PRINT : PRINT : PRINT : PRINT
500  PRINT "ENTER  'E' TO EXIT......": PRINT
510  INPUT "RETURN TO CONTINUE.....";A$
520  IF A$ = "E" THEN  GOTO 720
530  HOME
540  PRINT : PRINT : PRINT "ENTER THE ";N$;" NUMBER     ";
550  INPUT STUNO
560  PRINT : PRINT "ENTER COURSES BY PERIOD  :"
570  PRINT
580  FOR I = 1 TO 7
590  PRINT : PRINT "PERIOD   ";I;"  ";
600  INPUT SCHD(I)
610  NEXT I
620  PRINT
630  PRINT "DO YOU WANT TO MAKE ANY CHANGES"
640  INPUT "....ENTER (Y/N)...";Z$
650  IF Z$ = "Y" THEN  GOSUB 2500
660  NST = NST + 1
670  GOSUB 900
680  IF ANS = 1 THEN  GOSUB 1040
690  IF ANS = 2 THEN  GOSUB 3510
700  GOTO 490
710  REM
720  GOSUB 760
730  PRINT D$;"RUN MENU,D1"
740  REM *************************************************
750  REM  SUBROUTINE EXIT
760  REM *************************************************
770  PT = SS
780  IF ANS = 1 AND NST - OBAL > 25 THEN  GOSUB 3750
790  IF ANS = 1 THEN  GOSUB 3960
800  IF ANS = 2 AND NST - OBAL > 10 THEN  GOSUB 3750
810  IF ANS = 2 THEN  GOSUB 4080
820  GOSUB 3660
830  FLAG = 1
840  IF NCLASS - OCBAL > 5 THEN  GOSUB 3750
850  GOSUB 3850
860  RETURN
870  REM *************************************************
880  REM  SUBROUTINE FIND STUDENT-RECORD
890  REM *************************************************
900  PT = SS
910  IF PT <  > 0 THEN  GOTO 940
920  PT = 1:SS = 1
```

```
930   GOTO 1010
940   IF STNO < INDXS(PT,2) THEN IK = 1
950   IF STNO > INDXS(PT,2) THEN IK = 3
960   IF STNO = INDXS(PT,2) THEN  RETURN
970   LAST = PT
980   PT = INDXS(PT,IK)
990   IF PT <  > 0 THEN  GOTO 940
1000   INDXS(LAST,IK) = NST
1010   INDXS(NST,2) = STNO
1020   INDXS(NST,4) = NST - 1
1030   RETURN
1040   REM **************************************************
1050   REM  SUBROUTINE WRITE STUDENT-SCHEDULE
1060   REM **************************************************
1070   PRINT D$;"OPEN STUDENT-SCHEDULE,L104"
1080   PRINT D$;"WRITE STUDENT-SCHEDULE,R";NST - 1
1090   PRINT STNO
1100   FOR I = 1 TO 7
1110   BYT = 6 + (I - 1) * 14
1120   PRINT D$;"WRITE STUDENT-SCHEDULE,R";NST - 1;",B";BYT
1130   PRINT SCHD(I)
1140   NEXT I
1150   PRINT D$;"CLOSE STUDENT-SCHEDULE"
1160   GOSUB 3000
1170   RETURN
1180   REM **************************************************
1190   REM  SUBROUTINE READ STUDENT-INDEX
1200   REM **************************************************
1210   REM
1220   PRINT D$;"OPEN STUDENT-INDEX"
1230   PRINT D$;"READ STUDENT-INDEX"
1240   INPUT NST: INPUT SS: INPUT OBAL
1250   IF NST = 0 THEN  GOTO 1310
1260   FOR I = 1 TO NST
1270   FOR J = 1 TO 4
1280   INPUT INDXS(I,J)
1290   NEXT J
1300   NEXT I
1310   PRINT D$;"CLOSE STUDENT-INDEX"
1320   RETURN
1330   REM **************************************************
1340   REM  SUBROUTINE READ TEACHER-INDEX
1350   REM **************************************************
1360   PRINT D$;"OPEN TEACHER-INDEX"
1370   PRINT D$;"READ TEACHER-INDEX"
1380   INPUT NST: INPUT SS: INPUT OBAL
1390   IF NST = 0 THEN  GOTO 1440
```

```
1400  FOR I = 1  TO NST
1410  FOR J = 1  TO 4
1420  INPUT INDX(I,J)
1430  NEXT J: NEXT I
1440  PRINT D$;"CLOSE TEACHER-INDEX"
1450  RETURN
1460  REM ***********************************************
1470  REM  SUBROUTINE READ CLASS-INDEX
1480  REM ***********************************************
1490  PRINT D$;"OPEN CLASS-INDEX,D2"
1500  PRINT D$;"READ CLASS-INDEX"
1510  INPUT NCLASS: INPUT S2C: INPUT OCBAL
1520  IF NCLASS = 0 THEN  GOTO 1580
1530  FOR I = 1  TO NCLASS
1540  FOR J = 1  TO 4
1550  INPUT CINDX(I,J)
1560  NEXT J
1570  NEXT I
1580  PRINT D$;"CLOSE CLASS-INDEX"
1590  RETURN
1600  REM ***********************************************
1610  REM  SUBROUTINE IN-ORDER
1620  REM ***********************************************
1630  PT = SS
1640  IF INDXS(PT,1) <  > 0 THEN  GOTO 1760
1650  LN = LN + 1
1660  ODR(LN)  = PT
1670  D = 1
1680  GOSUB 1850
1690  D = 0
1700  PT = INDXS(PT,3)
1710  IF PT <  > 0 THEN  GOTO 1640
1720  GOSUB 1920
1730  IF D = 3  THEN  GOTO 1810
1740  IF D = 1  THEN  GOTO 1720
1750  GOTO 1650
1760  D = 2
1770  GOSUB 1850
1780  D = 0
1790  PT = INDXS(PT,1)
1800  GOTO 1640
1810  RETURN
1820  REM ***********************************************
1830  REM  SUBROUTINE PUSH
1840  REM ***********************************************
1850  LST = LST + 1
1860  STACK(LST,1)  = PT
```

```
1870  STACK(LST,2) = D
1880  RETURN
1890  REM **************************************************
1900  SUBROUTINE POP
1910  REM **************************************************
1920  IF LST = O THEN  GOTO 1970
1930  PT = STACK(LST,1)
1940  D = STACK(LST,2)
1950  LST = LST - 1
1960  RETURN
1970  D = 3
1980  RETURN
1990  REM **************************************************
2000  REM  SUBROUTINE LENGTH
2010  REM **************************************************
2020  K1 = 1
2030  IF 2 ^ K1 > LN THEN  GOTO 2050
2040  K1 = K1 + 1: GOTO 2030
2050  EX = K1 - 1
2060  LF = 2 ^ EX - 1
2070  ND = LN - LF
2080  K2 = 0
2090  FOR K1 = 1 TO 500 STEP 2
2100  K2 = K2 + 1
2110  OMIT(K2) = ODR(K1)
2120  IF K2 = ND GOTO 2140
2130  NEXT K1
2140  K5 = 2 * ND
2150  FOR K3 = 1 TO LN
2160  IF K3 > ND THEN  GOTO 2190
2170  ODR(K3) = ODR(2 * K3)
2180  GOTO 2210
2190  K5 = K5 + 1
2200  ODR(K3) = ODR(K5)
2210  NEXT K3
2220  RETURN
2230  REM **************************************************
2240  REM  SUBROUTINE ENTER-POINTERS
2250  REM **************************************************
2260  FOR K1 = 1 TO LF STEP 2
2270  INDX(ODR(K1),1) = 0
2280  INDX(ODR(K1),3) = 0
2290  NEXT K1
2300  PASS = 1
2310  KS = PASS + 1
2320  KB = 2 ^ PASS
2330  FOR I = KB TO LF STEP 2 ^ KS
```

```
2340  INDX(ODR(I),1) = ODR(I - 2 ^ (PASS - 1))
2350  INDX(ODR(I),3) = ODR(I + 2 ^ (PASS - 1))
2360  NEXT I
2370  PASS = PASS + 1
2380  IF 2 ^ PASS < LF THEN  GOTO 2310
2390  KD = 1
2400  INDXS(ODR(KD),1) = OMIT(KD)
2410  INDXS(ODR(KD),3) = OMIT(KD + 1)
2420  KD = KD + 2
2430  IF KD <  = ND THEN  GOTO 2400
2440  SS = ODR(2 ^ (EX - 1))
2450  FOR KZ = 1 TO ND
2460  INDXS(OMIT(KZ),1) = 0
2470  INDXS(OMIT(KZ),3) = 0
2480  NEXT KZ
2490  RETURN
2500  REM ******************************************************
2510  REM  SUBROUTINE CHANGE-SCHEDULE
2520  REM ******************************************************
2530  PRINT : PRINT
2540  GOSUB 2660
2550  PRINT "DO YOU WANT TO CHANGE THE"
2560  PRINT "     ";N$;"  NUMBER (Y/N)"
2570  INPUT H$
2580  IF H$ = "Y" THEN  GOSUB 2890
2590  IF H$ = "N" THEN  GOSUB 2800
2600  IF H$ <  > "Y" AND H$ <  > "N" THEN  GOTO 2550
2610  PRINT : PRINT "DO YOU WANT TO MAKE ANY MORE CHANGES"
2615  PRINT "        ";: INPUT "(Y/N) ";H$
2620  IF H$ = "Y" THEN  GOTO 2530
2630  IF H$ = "N" THEN  RETURN
2640  IF H$ <  > "Y" AND H$ <  > "N" THEN  GOTO 2610
2650  GOTO 2610
2660  REM ******************************************************
2670  REM  SUBROUTINE DISPLAY SCHEDULE
2680  REM ******************************************************
2690  HOME
2700  PRINT "SCHEDULE GIVEN IS: "
2710  PRINT
2720  PRINT "    ";N$;"  NUMBER  ";STN
2730  PRINT
2740  PRINT "PERIOD       COURSE"
2750  FOR I = 1 TO 7
2760  PRINT
2770  PRINT I,SCHD(I)
2780  NEXT I: PRINT
2790  RETURN
```

```
2795  REM
2800  REM **************************************************
2810  REM  SUBROUTINE CHANGE CLASSES
2820  REM **************************************************
2830  GOSUB 2660
2840  PRINT : PRINT
2845  PRINT "ENTER PERIOD NUMBER, COURSE NUMBER"
2850  INPUT J,COU
2860  SCHD(J) = COUD
2870  GOSUB 2660
2880  RETURN
2890  REM **************************************************
2900  REM  SUBROUTINE CHANGE-NUMBER
2910  REM **************************************************
2920  HOME : PRINT : PRINT : PRINT : PRINT
2930  PRINT "OLD ",N$;" NUMBER IS: ";STUNO: PRINT
2940  PRINT : PRINT : PRINT : PRINT
2950  PRINT : PRINT : PRINT : PRINT
2960  PRINT "ENTER NEW  ";N$;"NUMBER  ";
2970  INPUT STNO
2980  GOSUB 2660
2990  RETURN
3010  REM **************************************************
3020  REM  SUBROUTINE UPDATE-CLASS-INFO
3030  REM **************************************************
3040  PRINT D$;"OPEN CLASS-INFO,L225"
3050  FOR J = 1  TO 7
3060  PT = S2C
3070  IF PT <  > 0 THEN 3110
3080  S2C = 1:CSIZE = 1
3090  PT = 1
3100  NCLASS = 1: GOTO 3190
3110  IF SCHD(J)  < CINDX(PT,2)  THEN IK = 1
3120  IF SCHD(J)  > CINDX(PT,2)  THEN IK = 3
3130  IF SCHD(J)  = CINDX(PT,2)  THEN  GOTO 3260
3140  LAST = PT
3150  PT = CINDX(PT,IK)
3160  IF PT <  > 0 THEN 3110
3170  NCLASS = NCLASS + 1
3180  CINDX(LAST,IK) = NCLASS
3190  CINDX(NCLASS,2) = SCHD(J)
3200  CINDX(NCLASS,4) = NCLASS - 1
3210  PRINT D$;"WRITE CLASS-INFO,R";NCLASS - 1
3220  PRINT SCHD(J)
3230  PRINT D$;"WRITE CLASS-INFO,R";NCLASS - 1;",B7"
3240  PRINT J
3250  PT = NCLASS
```

```
3260  REC = CINDX(PT,4)
3270  IF ANS = 1 THEN  GOSUB 3320
3280  IF ANS = 2 THEN  GOSUB 3450
3290  NEXT J
3300  PRINT D$;"CLOSE CLASS-INFO"
3310  RETURN
3320  REM *************************************************
3330  REM  SUBROUTINE REVISE CLASS-INFO
3340  REM                (STUDENT)
3350  REM *************************************************
3360  PRINT D$;"READ CLASS-INFO,R";REC;",B12"
3370  INPUT CSIZE
3380  CSIZE = CSIZE + 1
3390  PRINT D$;"WRITE CLASS-INFO,R";REC;",B12"
3400  PRINT CSIZE
3410  BYT = 15 + (CSIZE - 1) * 6
3420  PRINT D$;"WRITE CLASS-INFO,R";REC;",B";BYT
3430  PRINT STNO
3440  RETURN
3450  REM *************************************************
3460  REM  SUBROUTINE REVISE CLASS-INFO(TEACHER)
3470  REM *************************************************
3480  PRINT D$;"WRITE CLASS-INFO,R";REC;",B4"
3490  PRINT STNO
3500  RETURN
3510  REM *************************************************
3520  REM  SUBROUTINE WRITE TEACHER-SCHEUDLE
3530  REM *************************************************
3540  PRINT D$;"OPEN TEACHER-SCHEDULE,L31"
3550  REM
3560  PRINT D$;"WRITE TEACHER-SCHEDULE,R";NST - 1
3570  PRINT STNO
3580  FOR I = 1  TO 7
3590  BYT = 3 + (I - 1) * 4
3600  PRINT D$;"WRITE TEACHER-SCHEDULE,R";NST - 1",B";BYT
3610  PRINT SCHD(I)
3620  NEXT I
3630  PRINT D$;"CLOSE TEACHER-SCHEDULE"
3640  GOSUB 3000
3650  RETURN
3660  REM *************************************************
3670  REM  SUBROUTINE CHANGE CLASS INDEX)
3680  REM *************************************************
3690  FOR I = 1  TO NCLASS
3700  FOR J = 1  TO 4
3710  INDX(I,J)  = CINDX(I,J)
3720  NEXT J: NEXT I
```

```
3730  SS = S2C
3740  RETURN
3750  REM **************************************************
3760  REM  SUBROUTINE BALANCE-INDEX
3770  REM **************************************************
3780  GOSUB 1610
3790  GOSUB 2000
3800  GOSUB 2240
3810  IF ANS = 1 OR ANS = 2 THEN OBAL = NST
3820  IF FLAG = 1 THEN OCBAL = NCLASS
3830  RETURN
3840  REM **************************************************
3850  REM  SUBROUTINE WRITE CLASS-INDEX
3860  REM **************************************************
3870  PRINT D$;"OPEN CLASS-INDEX"
3880  PRINT D$;"WRITE CLASS-INDEX"
3890  PRINT NCLASS: PRINT SS: PRINT OCBAL
3900  FOR I = 1 TO NCLASS
3910  FOR J = 1 TO 4
3920  PRINT INDX(I,J)
3930  NEXT J: NEXT I
3940  PRINT D$;"CLOSE CLASS-INDEX
3950  RETURN
3960  REM **************************************************
3970  REM  SUBROUTINE WRITE STUDENT-INDEX"
3980  REM **************************************************
3990  PRINT D$;"OPEN STUDENT-INDEX"
4000  PRINT D$;"WRITE STUDENT-INDEX"
4010  PRINT NST: PRINT SS: PRINT OBAL
4020  FOR I = 1 TO NST
4030  FOR J = 1 TO 4
4040  PRINT INDX(I,J)
4050  NEXT J: NEXT I
4060  PRINT D$;"CLOSE STUDENT-INDEX"
4070  RETURN
4080  REM **************************************************
4090  REM  SUBROUTINE WRITE TEACHER-INDEX
4100  REM **************************************************
4110  PRINT D$;"OPEN TEACHER-INDEX"
4120  PRINT D$;"WRITE TEACHER-INDEX"
4130  PRINT NST: PRINT SS: PRINT OBAL
4140  FOR I = 1 TO NST
4150  FOR J = 1 TO 4
4160  PRINT INDX(I,J)
4170  NEXT J: NEXT I
4180  PRINT D$;"CLOSE TEACHER-INDEX"
4190  RETURN
```

APPENDIX D

PROGRAM RETRIEVE SCHEDULES LISTING

```
 45  REM
 50  REM  PROGRAM RETRIEVE SCHEDULES
 52  REM
 55  REM
 60  DIM SNDX(500,4)
 65  DIM TINDX(50,4)
 70  DIM CINDX(150,4)
 75  DIM CREC(7)
 80  DIM RM(7)
 85  DIM TCH(7)
 90  DIM SCHC(7)
 95  DIM ODR(500)
105  DIM STACK(256,2)
110  D$ =  CHR$ (4)
115  GOSUB 235
120  GOSUB 295
125  GOSUB 360
130  HOME : PRINT : PRINT
135  PRINT " THIS PROGRAM DISPLAYS EITHER "
140  PRINT : PRINT
145  PRINT "1.....STUDENT SCHEDULE...
150  PRINT : PRINT
155  PRINT "2.....TEACHER SCHEDULE...
160  PRINT : PRINT
165  PRINT "3.....TO PRINT ALL STUDENT SCHEDULES"
170  PRINT : PRINT
175  PRINT "4.....TO PRINT ALL TEACHER SCHEDULES"
180  PRINT : PRINT
185  PRINT "5.....TO END THE PROGRAM"
190  PRINT : PRINT
195  PRINT " ENTER THE NUMBER OF YOUR CHOICE...."
200  PRINT : PRINT : PRINT : INPUT ANS
205  IF ANS = 1 THEN  GOSUB 425
210  IF ANS = 2 THEN  GOSUB 560
215  IF ANS = 3 THEN  GOSUB 1080
220  IF ANS = 4 THEN  GOSUB 1620
225  IF ANS = 5 THEN  GOTO 233
230  GOTO 130
```

```
233  REM  TERMINATE EXECUTION AND RUN MENU PROGRAM
234  PRINT D$;"RUN MENU,D1"
235  REM
240  REM ************************************************
242  REM SUBROUTINE READ STUDENT INDEX
243  REM***************************************************
245  PRINT D$;"OPEN STUDENT-INDEX,D2"
250  PRINT D$;"READ STUDENT-INDEX"
255  INPUT NST: INPUT SP: INPUT OBAL
260  FOR I = 1  TO NST
265  FOR J = 1  TO 4
270  INPUT SNDX(I,J)
275  NEXT J
280  NEXT I
285  PRINT D$;"CLOSE STUDENT-INDEX"
290  RETURN
293  REM ************************************************
295  REM  SUBROUTINE READ TEACHER INDEX
300  REM ************************************************
305  PRINT D$;"OPEN TEACHER-INDEX"
310  PRINT D$;"READ TEACHER-INDEX"
315  INPUT NTEA: INPUT TP: INPUT OBAL
320  IF NTEA = 0 THEN  GOTO 350
325  FOR I = 1  TO NTEA
330  FOR J = 1  TO 4
335  INPUT TINDX(I,J)
340  NEXT J
345  NEXT I
350  PRINT D$;"CLOSE"
352  RETURN
353  REM ************************************************
360  REM  SUBROUTINE READ CLASS INDEX
365  REM ************************************************
370  PRINT D$;"OPEN CLASS-INDEX"
375  PRINT D$;"READ CLASS-INDEX"
380  INPUT NCLASS: INPUT CP: INPUT OCBAL
385  IF NCLASS = 0 THEN  GOTO 415
390  FOR I = 1  TO NCLASS
395  FOR J = 1  TO 4
400  INPUT CINDX(I,J)
405  NEXT J
410  NEXT I
415  PRINT D$;"CLOSE"
420  RETURN
422  REM ************************************************
425  REM  SUBROUTINE RETREIVE STUDENT
```

```
426  REM                    (ONE STUDNENT)
430  REM **********************************************
435  HOME : PRINT : PRINT
440  PRINT "  ENTER STUDENT NUMBER"
445  PRINT : PRINT : INPUT NUMB
448  SNUMB = NUMB
450  PT = SP
455  IF PT = 0 THEN  GOTO 481
460  IF NUMB < SNDX(PT,2) THEN IK = 1
465  IF NUMB > SNDX(PT,2) THEN IK = 3
470  IF NUMB = SNDX(PT,2) THEN  GOTO 488
475  PT = SNDX(PT,IK)
480  GOTO 455
481  PRINT : PRINT
482  PRINT "STUDENT     ";NUMB
483  PRINT : PRINT "IS NOT IN THE DATA BASE"
485  PRINT : PRINT
486  INPUT "PRESS RETURN TO CONTINUE  ";ZZ$
487  GOTO 550
488  REC = SNDX(PT,4)
490  PRINT D$;"OPEN STUDENT-SCHEDULE,L104"
495  FOR I = 1 TO 7
500  BYT = 6 + (I - 1) * 14
505  PRINT D$;"READ STUDENT-SCHEDULE,R";REC;",B";BYT
510  INPUT SCHD(I)
515  NEXT I
520  PRINT D$;"CLOSE"
525  FOR M = 1 TO 7
530  CNUMB = SCHD(M)
535  PT = CP
540  GOSUB 1430
542  CREC(M) = REC
544  NEXT M
546  GOSUB 1500
548  GOSUB 1560
550  RETURN
555  REM **********************************************
560  REM  SUBROUTINE FIND TEACHER RECORD
565  REM **********************************************
570  HOME : PRINT : PRINT
575  PRINT "  ENTER TEACHER NUMBER  "
580  PRINT : PRINT : PRINT : INPUT NUMB
582  TNUMB = NUMB
585  PT = TP
590  IF PT = 0 THEN  GOTO 615
595  IF NUMB < TINDX(PT,2) THEN IK = 1
600  IF NUMB > TINDX(PT,2) THEN IK = 3
```

```
605  IF NUMB = TINDX(PT,2) THEN  GOTO 625
610  PT = TINDX(PT,IK)
612  GOTO 590
615  PRINT:PRINT"TEACHER ";NUMB;" IS NOT IN THE DATA BASE"
618  PRINT : PRINT : INPUT "PRESS RETURN TO CONTINUE  ";ZZ$
620  GOTO 710
625  REC = TINDX(PT,4)
628  PRINT D$;"OPEN TEACHER-SCHEDULE,L31"
630  FOR I = 1 TO 7
635  BYT = 3 + (I - 1) * 4
640  PRINT D$;"READ TEACHER-SCHEDULE,R";REC;",B";BYT
645  INPUT SCHD(I)
650  NEXT I
655  PRINT D$;"CLOSE"
660  FOR M = 1 TO 7
670  CNUMB = SCHD(M)
675  PT = CP
680  GOSUB 1430
685  CREC(M) = REC
690  NEXT M
695  GOSUB 1500
700  GOSUB 1800
710  RETURN
1070  REM ***********************************************
1080  REM  SUBROUTINE RETREIVE ALL STUDENT SCHEDULES
1085  REM ***********************************************
1110  PT = SP
1115  GOSUB 1220
1120  FOR T = 1 TO NST
1125  PRINT D$;"OPEN STUDENT-SCHEDULE,L104"
1130  PRINT D$;"READ STUDENT-SCHEDULE,R";ODR(T);",B0"
1135  INPUT SNUMB
1140  FOR M = 1 TO 7
1145  BYT = 6 + (M - 1) * 14
1150  PRINT D$;"READ STUDENT-SCHEDULE,R";ODR(T);",B";BYT
1155  INPUT SCHD(M)
1160  NEXT M
1165  PRINT D$;"CLOSE STUDENT-SCHEDULE"
1170  FOR M = 1 TO 7
1175  CNUMB = SCHD(M)
1180  PT = CP
1185  GOSUB 1425
1190  CREC(M) = REC
1195  NEXT M
1200  GOSUB 1495
1205  GOSUB 1555
1210  NEXT T
```

```
1215  RETURN
1220  REM ***********************************************
1225  REM  SUBROUTINE IN-ORDER
1230  REM ***********************************************
1232  LN = 0
1235  LT = 0
1240  IF ANS = 3 AND SNDX(PT,1) <  > 0 THEN  GOTO 1310
1242  IF ANS = 4 AND TINDX(PT,1) <  > 0 THEN  GOTO 1310
1245  LN = LN + 1
1250  IF ANS = 3 THEN ODR(LN)  = SNDX(PT,4)
1252  IF ANS = 4 THEN ODR(LN)  = TINDX(PT,4)
1255  D = 1
1260  REM  PUSH STAACK
1265  GOSUB 1340
1270  D = 0
1275  IF ANS = 3 THEN PT = SNDX(PT,3)
1277  IF ANS = 4 THEN PT = TINDX(PT,3)
1280  IF PT <  > 0 THEN  GOTO 1240
1285  REM  POP STACK
1290  GOSUB 1375
1295  IF D = 3 THEN  GOTO 1335
1300  IF D = 1 THEN  GOTO 1285
1305  GOTO 1245
1310  D = 2
1315  GOSUB 1340
1320  D = 0
1325  IF ANS = 3 THEN PT = SNDX(PT,1)
1327  IF ANS = 4 THEN PT = TINDX(PT,1)
1330  GOTO 1240
1335  RETURN
1340  REM ***********************************************
1345  REM  SUBROUTINE PUSH
1350  REM ***********************************************
1355  LT = LT + 1
1360  STACK(LT,1)  = PT
1365  STACK(LT,2)  = D
1370  RETURN
1375  REM ***********************************************
1380  REM  SUBROUTINE POP
1385  REM ***********************************************
1390  IF LT = 0 THEN  GOTO 1415
1395  PT = STACK(LT,1)
1400  D = STACK(LT,2)
1405  LT = LT - 1
1410  GOTO 1420
1415  D = 3
1420  RETURN
```

```
1426  REM
1425  REM *************************************************
1430  REM  SUBROUTINE TO FIND CLASS RECORD
1435  REM *************************************************
1440  IF PT = 0 THEN  GOTO 1470
1445  IF CNUMB < CINDX(PT,2) THEN IK = 1
1450  IF CNUMB > CINDX(PT,2) THEN IK = 3
1455  IF CNUMB = CINDX(PT,2) THEN  GOTO 1485
1460  PT = CINDX(PT,IK)
1465  GOTO 1440
1470  REM  COURSE IS NOT IN DATA BASE
1475  FLAG = 1
1480  GOTO 1490
1485  REC = CINDX(PT,4)
1490  RETURN
1495  REM *************************************************
1500  REM  SUBROUTINE TO READ TEACHER AND ROOM NUMBER
1505  REM *************************************************
1510  PRINT D$;"OPEN CLASS-INFO,L225"
1515  FOR I = 1 TO 7
1520  PRINT D$;"READ CLASS-INFO,R";CREC(I);",B4"
1525  INPUT TCH(I)
1530  PRINT D$;"READ CLASS-INFO,R";CREC(I);",B9"
1535  INPUT RM(I)
1540  NEXT I
1545  PRINT D$;"CLOSE CLASS-INFO"
1550  RETURN
1555  REM *************************************************
1560  REM  PRINT SUBROUTINE
1565  REM *************************************************
1570  HOME : PRINT : PRINT
1575  PRINT "SCHEDULE FOR STUDENT  ";SNUMB
1580  PRINT : PRINT
1585  PRINT "PERIOD"; TAB( 13)"COURSE";
1586  PRINT TAB( 24)"TEACHER"; TAB( 35)"ROOM"
1590  FOR I = 1 TO 7
1595  PRINT
1600  PRINT  TAB( 3)I; TAB( 15)SCHD(I);
1605  PRINT TAB( 26)TCH(I);  TAB( 36)RM(I)
1605  NEXT I
1610  PRINT : PRINT : INPUT "PRESS RETURN TO CONTINUE";ZZ$
1615  RETURN
1620  REM *************************************************
1625  REM  SUBROUTINE RETRIEVE ALL TEACHER SCHEDULES
1630  REM *************************************************
1635  IF NTEA = 0 THEN  GOTO 1770
1660  PT = TP
```

```
1665  GOSUB 1220
1670  FOR T = 1  TO NTEA
1675  PRINT D$;"OPEN TEACHER-SCHEDULE,L31"
1680  PRINT D$;"READ TEACHER-SCHEDULE,R";ODR(T);",B0"
1685  INPUT TNUMB
1690  FOR M = 1  TO 7
1695  BYT = 3 + (M - 1) * 4
1700  PRINT D$;"READ TEACHER-SCHEDULE,R";ODR(T);",B";BYT
1705  INPUT SCHD(M)
1710  NEXT M
1715  PRINT D$;"CLOSE TEACHER-SCHEDULE"
1720  FOR M = 1  TO 7
1725  CNUMB  =  SCHD(M)
1730  PT = CP
1735  GOSUB 1425
1740  CRC(M)  = REC
1745  NEXT M
1750  GOSUB 1495
1755  GOSUB 1800
1760  NEXT T
1765  GOTO 1795
1770  REM  TEACHER-INDEX HAS NOT BEEN BUILD
1775  HOME : PRINT : PRINT : PRINT
1780  PRINT "TEACHER-INDEX HAS NOT BEEN BUILD"
1785  PRINT : PRINT
1790  PRINT :PRINT:INPUT "PRESS RETURN  TO CONTINUE  ";ZZ$
1795  RETURN
1800  REM *************************************************
1805  REM  SUBROUTINE PRINT TEACHER SCHEDULE
1806  REM *************************************************
1809  HOME : PRINT : PRINT
1810  PRINT "SCHEUDLE FOR TEACHER  ";TNUMB
1815  PRINT : PRINT
1820  PRINT "PERIOD"; TAB( 15)"CLASS"; TAB( 25)"ROOM"
1825  FOR I = 1  TO 7
1830  PRINT
1835  PRINT  TAB( 3)I; TAB( 16)SCHD(I); TAB( 26)RM(I)
1840  NEXT I
1845  PRINT : PRINT : INPUT "PRESS RETURN TO CONTINUE  ";ZZ$
1850  RETURN
```

APPENDIX E

PROGRAM UPDATE FILES

```
100   REM  PROGRAM TO UPDATE RECORDS
110   REM
120   DIM SNDX(500,4)
130   DIM TINDX(50,4)
140   DIM CINDX(150,4)
150   DIM SCHD(7)
160   DIM CH(7,2)
170   D$ =  CHR$ (4)
180   GOSUB 450
190   GOSUB 570
200   GOSUB 700
210   HOME : PRINT : PRINT
220   PRINT "THIS PROGRAM UPDATES:"
230   PRINT : PRINT
240   PRINT "ENTER THE NUMBER OF YOUR CHOICE"
250   PRINT
260   PRINT "1.....STUDENT-SCHEDULE"
270   PRINT
280   PRINT "2.....TEACHER-SCHEDULE"
290   PRINT
300   PRPINT "3.....STUDENT GRADES"
310   PRINT
320   PRINT "4.....ROOM NUMBERS"
330   PRINT
340   PRINT "5.....ENTER 5 TO END PROGRAM"
350   PRINT : PRINT : INPUT ANS
360   IF ANS = 1 THEN  GOSUB 830
370   IF ANS = 2 THEN  GOSUB 1810
380   IF ANS = 3 THEN  GOSUB 2960
390   IF ANS = 4 THEN  GOSUB 3680
400   IF ANS = 5 THEN  GOTO 430
410   IF NCLASS < > CLASSO THEN  GOSUB 2840
420   GOTO 210
430   REM  EXIT PROGRAM AND RUN MENU PROGRAM
440   PRINT D$;"RUN MENU,D1"
450   REM ***************************************************
460   REM  SUBROUTINE TO READ ST-INDEX"
470   REM ***************************************************
```

```
480  PRINT D$;"OPEN STUDENT-INDEX,D2"
490  PRINT D$;"READ STUDENT-INDEX"
500  INPUT NST: INPUT SP: INPUT OBAL
510  FOR I = 1  TO NSTU
520  FOR J = 1  TO 4
530  INPUT SNDX(I,J)
540  NEXT J: NEXT I
550  PRINT D$;"CLOSE ST-INDEX"
560  RETURN
570  REM ***************************************************
580  REM  SUBROUINE TO READ CLASS-INDEX"
590  REM ***************************************************
600  PRINT D$;"OPEN CLASS-INDEX"
610  PRINT D$;"READ CLASS-INDEX"
620  INPUT NCLASS: INPUT CP: INPUT OCBAL
630  CLASS0 = NCLASS
640  FOR I = 1  TO NCLASS
650  FOR J = 1  TO 4
660  INPUT CINDX(I,J)
670  NEXT J: NEXT I
680  PRINT D$;"CLOSE CLASS-INDEX"
690  RETURN
700  REM ***************************************************
710  REM  SUBROUTIONE TO READ TEACHER-INDEX"
720  REM ***************************************************
730  PRINT D$;"OPEN TEACHER-INDEX"
740  PRINT D$;"READ TEACHER-INDEX"
750  INPUT NTEA: INPUT TP: INPUT OBAL
760  IF NTEA = 0 THEN  GOTO 810
770  FOR I = 1  TO NTEA
780  FOR J = 1  TO 4
790  INPUT TINDX(I,J)
800  NEXT J: NEXT I
810  PRINT D$;"CLOSE TEACHER-INDEX"
820  RETURN
830  REM ***************************************************
840  REM  SUBROUTINE STUDENT-SCHEDULE
850  REM ***************************************************
860  HOME : PRINT : PRINT
870  PRINT "ENTER STUDENT NUMBER"
880  PRINT : PRINT : INPUT NUMB
890  REM  FIND STUDENT RECORD NUMBER
900  PT = SP
910  IF PT = 0 THEN  GOTO 970
920  IF NUMB < SNDX(PT,2) THEN IK = 1
930  IF NUMB > SNDX(PT,2) THEN IK = 3
```

```
940  IF NUMB = SNDX(PT,2) THEN  GOTO 1000
950  PT = SNDX(PT,IK)
960  GOTO 910
970  REM  STUDENT IS NOT IN DATA BASE
980  GOSUB 2490
990  GOTO 1800
1000  REC = SNDX(PT,4)
1010  TREC = REC
1020  PRINT D$;"OPEN STUDENT-SCHEDULE,L104"
1030  FOR I = 1 TO 7
1040  BYT = 6 + (I - 1) * 14
1050  PRINT D$;"READ STUDENT-SCHEDULE,R";REC;",B";BYT
1060  INPUT SCHD(I)
1070  NEXT I
1080  PRINT D$;"CLOSE STUDENT-SCHEDULE"
1090  HOME : PRINT : PRINT
1100  PRINT "STUDENT ";NUMB;" CURRENT SCHEDULE IS:": PRINT
1110  PRINT "PERIOD          CLASS"
1120  FOR I = 1 TO 7
1130  PRINT
1140  PRINT "   ";I;"                 ";SCHD(I)
1150  NEXT I
1160  PRINT : PRINT
1170  INPUT "ENTER PERIOD,NEW COURSE  ";PN,COURSE
1180  CNUMB = SCHD(PN)
1190  FLAG = 1
1200  GOSUB 2630
1210  IF FLAG = 3 THEN  GOTO 1720
1220  FLAG = 0
1230  SCHD(PN)  = COURSE
1240  PRINT D$;"OPEN STUDENT-SCHEDULE,L104"
1250  BYT = 6 + (PN - 1) * 14
1260  PRINT D$;"WRITE STUDENT-SCHEDULE,R";TREC;",B";BYT
1270  PRINT SCHD(PN)
1280  REM  UPDATE CLASS-INFO RELATION
1290  PRINT D$;"CLOSE STUDENT-SCHEDULE"
1300  REM  UPDATE OLD CLASS LIST
1310  PRINT D$;"OPEN CLASS-INFO,L225"
1320  REM  DECREMENT CSIZE
1330  PRINT D$;"READ CLASS-INFO,R";REC;",B12"
1340  INPUT CSIZE
1350  CSIZE = CSIZE - 1
1360  PRINT D$;"WRITE CLASS-INFO,R";REC;",B12"
1370  PRINT CSIZE
1380  FOR I = 1 TO (CSIZE + 1)
1390  BYT = 15 + (I - 1) * 6
1400  PRINT D$;"READ CLASS-INFO,R";REC;",B";BYT
```

```
1410  INPUT LST(I)
1420  NEXT I
1430  REM  FIND AND DELETE STUDENT
1440  FOR J = 1 TO (CSIZE + 1)
1450  IF NUMB = LST(J) THEN  GOTO 1470
1460  NEXT J
1470  IF J = (CSIZE + 1) THEN  GOTO 1540
1480  FOR I = J TO CSIZE
1490  BYT = 15 + (I - 1) * 6
1500  PRINT D$;"WRITE CLASS-INFO,R";REC;",B";BYT
1510  PRINT LST(I + 1)
1520  NEXT I
1530  GOTO 1570
1540  BYT = 15 + CSIZE * 6
1550  PRINT D$;"WRITE CLASS-INFO,R";REC;",B";BYT
1560  PRINT 0
1570  REM  ADD STUDENT TO NEW CLASS LIST
1580  CNUMB = SCHD(PN)
1590  FLAG = 0
1600  GOSUB 2630
1610  PRINT D$;"READ CLASS-INFO,R";REC;",B12"
1620  INPUT CSIZE
1630  CSIZE = CSIZE + 1
1640  PRINT D$;"WRITE CLASS-INFO,R";REC;",B12"
1650  PRINT CSIZE
1660  BYT = 15 + (CSIZE - 1) * 6
1670  PRINT D$;"WRITE CLASS-INFO,R";REC;",B";BYT
1680  PRINT NUMB
1690  PRINT D$;"CLOSE CLASS-INFO"
1700  GOTO 1740
1710  PRINT D$;"CLOSE CLASS-INFO"
1720  PRINT : PRINT "STUDENT ";NUMB;
1725  PRINT " IS NOT ENROLLED IN COURSE  ";COURSE;
1726  PRINT "  IN PERIOD ";PN
1730  FLAG = 0
1740  HOME : PRINT : PRINT : PRINT
1750  PRINT "DO YOU WANT TO MAKE ANY MORE CHANGES"
1760  PRINT "  FOR THIS STUDENT "
1770  INPUT "...(Y/N)..";A$
1780  IF A$ = "Y" THEN  GOTO 1090
1790  IF A$ < > "N" THEN  GOTO 1750
1800  RETURN
1810  REM ***********************************************
1820  REM  SUBROUTINE TEACHER-SCHEDULE
1830  REM ***********************************************
1840  HOME : PRINT : PRINT
1850  PRINT "ENTER TEACHER NUMBER"
```

```
1860  PRINT : PRINT : INPUT NUMB
1870  PT = TP
1880  IF PT = 0 THEN  GOTO 1940
1890  IF NUMB < TINDX(PT,2) THEN IK = 1
1900  IF NUMB > TINDX(PT,2) THEN IK = 3
1910  IF NUMB = TINDX(PT,2) THEN  GOTO 1970
1920  PT = TINDX(PT,IK)
1930  GOTO 1880
1940  REM  TEACHER IS NOT IN DATA BASE
1950  GOSUB 2490
1960  GOTO 2480
1970  TREC = TINDX(PT,4)
1980  PRINT D$;"OPEN TEACHER-SCHEDULE,L31"
1990  FOR I = 1 TO 7
2000  BYT = 3 + (I - 1) * 4
2010  PRINT D$;"READ TEACHER-SCHEDULE,R";TREC;",B";BYT
2020  INPUT SCHD(I)
2030  NEXT I
2040  PRINT D$;"CLOSE TEACHER-SCHEDULE"
2050  HOME : PRINT : PRINT : PRINT
2060  PRINT "TEACHER  ";NUMB;"  CURRENT SCHEDULE IS:  "
2070  PRINT : PRINT "PERIOD     CLASS"
2080  FOR I = 1 TO 7
2090  PRINT
2100  PRINT I;"          ";SCHD(I)
2110  NEXT I
2120  PRINT : PRINT
2130  INPUT "ENTER PERIOD,NEW COURSE";PN,COURSE
2140  IF SCHD(PN) <  > COURSE THEN  GOTO 2180
2150  PRINT "TEACHER ALREADY ASSIGNED TO CLASS  ";CN
2155  PRINT "  IN PERIOD  ";PN
2160  INPUT "PRESS RETURN TO CONTINUE";ZZ$
2170  GOTO 2235
2180  FLAG = 1
2185  CNUMB = SCHD(PN)
2190  GOSUB 2630
2200  IF FLAG = 3 THEN  GOTO 2420
2210  SCHD(PN)  = COURSE
2220  PRINT D$;"OPEN TEACHER-SCHEDULE,L31"
2230  BYT = 3 + (PN - 1) * 4
2240  PRINT D$;"WRITE TEACHER-SCHEDULE,R";TREC;",B";BYT
2250  PRINT SCHD(PN)
2260  PRINT D$;"CLOSE TEACHER-SCHEDULE"
2270  ROLD = REC
2280  CNUMB = SCHD(PN)
2290  FLAG = 0
2300  GOSUB 2630
```

```
2310  RNW = REC
2320  PRINT D$;"OPEN CLASS-INFO,L225"
2330  PRINT D$;"READ CLASS-INFO,R";ROLD;",B4"
2340  INPUT TEMP
2350  IF TEMP <  > NUMB GOTO 2380
2360  PRINT D$;"WRITE CLASS-INFO,R";ROLD;",B4"
2370  PRINT 0
2380  PRINT D$;"WRITE CLASS-INFO,R";RNW;",B4"
2390  PRINT NUMB
2400  PRINT D$;"CLOSE CLASS-INFO"
2410  GOTO 2430
2420  PRINT : PRINT "TEACHER ";NUMB;
2415  PRINT " IS NOT IN COURSE ";COURSE;" IN PERIOD ";PN
2430  PRINT : PRINT :"DO YOU WANT TO MAKE ANY MORE CHANGES"
2440  PRINT : PRINT "    FOR THIS TEACHER"
2450  PRINT : INPUT "....(Y/N)....";A$
2460  IF A$ = "Y" THEN  GOTO 2050
2470  IF A$ <  > "N" THEN  GOTO 2430
2480  RETURN
2490  REM ***********************************************
2500  REM  SUBROUTINE MISSING-PERSON
2510  REM ***********************************************
2520  HOME : PRINT : PRINT : PRINT : PRINT : PRINT
2530  IF ANS = 1 THEN  PRINT "STUDENT";
2540  IF ANS = 2 THEN  PRINT "TEACHER";
2550  PRINT "  ";NUMB;" IS NOT IN THE INDEX"
2560  PRINT : PRINT : PRINT :
2570  PRINT "USE THE BUILD PROGRAM TO ENTER"
2580  PRINT : PRINT
2590  PRINT "TEACHERS AND STUDENTS INTO THE DATA BASE"
2600  PRINT : PRINT : PRINT : PRINT
2610  INPUT "PRESS RETURN TO CONTINUE";ZZ$
2620  RETURN
2630  REM ***********************************************
2640  REM  SUBROUTINE TO FIND CLASS RECORD NUMBER
2650  REM ***********************************************
2660  PT = CP
2670  IF PT = 0 THEN  GOTO 2750
2680  IF CNUMB < CINDX(PT,2) THEN IK = 1
2690  IF CNUMB > CINDX(PT,2) THEN IK = 3
2700  IF CNUMB = CINDX(PT,2) THEN  GOTO 2730
2710  PT = CINDX(PT,IK)
2720  GOTO 2670
2730  REC = CINDX(PT,4)
2740  GOTO 2830
2750  IF FLAG = 1 THEN  GOTO 2820
2760  NCLASS = NCLASS + 1
```

```
2770  CINDX(LAST,IK) = NCLASS
2780  CINDX(NCLASS,2) = SCHD(PN)
2790  CINDX(NCLASS,4) = NCLASS - 1
2800  REC = NCLASS - 1
2810  GOTO 2830
2820  FLAG = 3
2830  RETURN
2840  REM **************************************************
2850  REM  SUBROUTINE WRITE CLASS-INDEX
2860  REM **************************************************
2870  PRINT D$;"OPEN CLASS-INDEX"
2880  PRINT D$;"OPEN CLASS-INDEX"
2890  PRINT NCLASS: PRINT CP: PRINT OCBAL
2900  FOR I = 1 TO NCLASS
2910  FOR J = 1 TO 4
2920  PRINT CINDX(I,J)
2930  NEXT J: NEXT I
2940  PRINT D$;"CLOSE CLASS-INDEX"
2950  RETURN
2960  REM **************************************************
2970  REM  SUBOUTINE UPDATE-GRADES
2980  REM **************************************************
2990  HOME : PRINT : PRINT : PRINT
3000  PRINT "1.....TO UPDATE MID-SEMESTER GRADE"
3010  PRINT : PRINT
3020  PRINT "2.....TO UPDATE FINIAL-GRADE"
3030  PRINT : PRINT : PRINT
3040  PRINT "ENTER THE NUMBER OF YOUR CHOICE"
3050  INPUT BANS
3060  IF BANS = 1 THEN  GOSUB 3560
3070  IF BANS = 2 THEN  GOSUB 3620
3080  IF BANS <  > 1 AND BANS <  > 2 THEN  GOTO 2990
3090  HOME : PRINT : PRINT : PRINT
3100  INPUT "ENTER STUDENT NUMBER  ";NUMB
3110  PT = SP
3120  IF PT = 0 THEN  GOTO 3180
3130  IF NUMB < SNDX(PT,2) THEN IK = 1
3140  IF NUMB > SNDX(PT,2) THEN IK = 3
3150  IF NUMB = SNDX(PT,2) THEN  GOTO 3210
3160  PT = SNDX(PT,IK)
3170  GOTO 3120
3180  PRINT : PRINT "STUDENT  ";NUMB;
3185  PRINT "  IS NOT IN THE DATA BASE"
3190  PRINT : INPUT "PRESS RETURN TO CONTINUE";ZZ$
3200  GOTO 3550
3210  REC = SNDX(PT,4)
3220  PRINT D$;"OPEN STUDENT-SCHEDULE,L104"
```

```
3230  FOR I = 1 TO 7
3240  BYT = 6 + (I - 1) * 14
3250  PRINT D$;"READ STUDENT-SCHEDULE,R";REC;",B";BYT
3260  INPUT SCHD(I)
3270  BYT = B1 + (I - 1) * 14
3280  PRINT D$;"READ STUDENT-SCHEDULE,R";REC;",B";BYT
3290  INPUT GDE(I)
3300  BYT = B2 + (I - 1) * 14
3310  PRINT D$;"READ STUDENT-SCHEDULE,R";REC;",B";BYT
3320  INPUT COM(I)
3330  NEXT I
3340  PRINT D$;"CLOSE STUDENT-SCHEDULE"
3350  HOME : PRINT : PRINT
3360  PRINT "CURRENT GRADES FOR STUDENT ";NUMB;" ARE:"
3370  PRINT : PRINT
3380  PRINT "PERIOD"; TAB( 11)"CLASS"; TAB( 20)"GRADE";
3385  PRINT TAB( 30)"COMMENT"
3390  FOR I = 1 TO 7
3400  PRINT
3410  PRINT  TAB( 3)I; TAB( 12)SCHD(I); TAB( 22)GDE(I);
3405  PRINT TAB( 33)COM(I)
3420  NEXT I
3430  PRINT
3440  PRINT "ENTER PERIOD, NEW GRADE, NEW COMMENT"
3450  INPUT PN,GD,CM
3460  IF PN > 7 OR PN < 1 THEN  GOTO 3440
3470  PRINT D$;"OPEN STUDENT-SCHEDULE,L104"
3480  BYT = B1 + (PN - 1) * 14
3490  PRINT D$;"WRITE STUDENT-SCHEDULE,R";REC;",B";BYT
3500  PRINT GD
3510  BYT = B2 + (PN - 1) * 14
3520  PRINT D$;"WRITE STUDENT-SCHEDULE,R";REC;",B";BYT
3530  PRINT CM
3540  PRINT D$;"CLOSE STUDENT-SCHEDULE"
3550  RETURN
3560  REM ***********************************************
3570  REM  SUBROUTINE BYTE POINTERS FOR MID-SEMESTER
3580  REM ***********************************************
3590  B1 = 10
3600  B2 = 13
3610  RETURN
3620  REM ***********************************************
3630  REM  SUBROUTINE BYTE-POINTERS-MID-SEMESTER
3640  REM ***********************************************
3650  B1 = 15
3660  B2 = 18
3670  RETURN
```

```
3675  REM
3680  REM *************************************************
3690  REM  SUBROUTINE CHANGE-ROOM
3700  REM *************************************************
3710  HOME : PRINT : PRINT : PRINT
3720  INPUT "ENTER THE CLASS NUMBER  ";CNUMB
3730  PT = CP
3740  IF PT = 0 THEN  GOTO 3800
3750  IF CNUMB < CINDX(PT,2) THEN IK = 1
3760  IF CNUMB > CINDX(PT,2) THEN IK = 3
3770  IF CNUMB = CINDX(PT,2) THEN  GOTO 3830
3780  PT = CINDX(PT,IK)
3790  GOTO 3740
3800  PRINT : PRINT : PRINT "COURSE  ";CNUMB;
3805  PRINT  "IS NOT IN THE DATA BASE"
3810  INPUT "PRESS RETURN TO CONTINUE ";ZZ$
3820  GOTO 3970
3830  REC = CINDX(PT,4)
3840  PRINT D$;"OPEN CLASS-INFO,L225"
3850  PRINT D$;"READ CLASS-INFO,R";REC;",B9"
3860  INPUT RN
3870  PRINT D$;"CLOSE CLASS-INFO"
3880  HOME : PRINT : PRINT : PRINT
3890  PRINT "CLASS  ";CNUMB;"  IS CURRENTLY SCHEDULED IN"
3900  PRINT : PRINT " ROOM          ";RN
3910  PRINT : PRINT : PRINT : PRINT
3920  INPUT "ENTER NEW ROOM NUMBER   ";NRM
3930  PRINT D$;"OPEN CLASS-INFO,L225"
3940  PRINT D$;"WRITE CLASS-INFO,R";REC;",B9"
3950  PRINT NRM
3960  PRINT D$;"CLOSE CLASS-INFO"
3970  RETURN
```

APPENDIX F

PROGRAM ENTER GRADES

```
100  REM
110  REM  PROGRAM TO ENTER GRADES
120  REM
130  DIM TINDX(50,4)
140  DIM CINDX(150,4)
150  DIM SCHD(7)
160  DIM GDE(35)
170  DIM COM(35)
180  DIM SREC(35)
190  D$ =  CHR$ (4)
200  GOSUB 740
210  GOSUB 870
220  GOSUB 1000
230  HOME : PRINT : PRINT
240  PRINT : PRINT : PRINT : PRINT
250  PRINT "1.....FOR MID-SEMESTER GRADES"
260  PRINT : PRINT
270  PRINT "2.....OR FINIAL GRADES"
280  PRINT : PRINT
290  INPUT "ENTER NUMBER OF YOUR CHOICE";ANS
300  IF ANS = 1 THEN  GOSUB 620
310  IF ANS = 2 THEN  GOSUB 680
320  IF ANS <  > 1 AND ANS <  > 2 THEN  GOTO 230
330  HOME : PRINT : PRINT : PRINT
340  PRINT "   ENTER THE TEACHER NUMBER"
350  PRINT : PRINT : PRINT
360  PRINT "              OR"
370  PRINT : PRINT : PRINT
380  PRINT "ENTER ..-9..TO TERMINATE PROGRAM"
390  PRINT : PRINT
400  INPUT NUMB
410  IF NUMB =  - 9 THEN  GOTO 600
420  GOSUB 1130
430  IF FTEAC = 1 THEN  GOTO 530
440  GOSUB 1280
450  FOR KK = 1 TO 7
460  NUMB = SCHD(KK)
470  IF NUMB = 999 GOTO 510
```

```
480  GOSUB 1390
490  GOSUB 1530
500  GOSUB 1920
510  NEXT KK
520  GOTO 310
530  REM  GIVEN TEACHER IS NOT IN DATA BASE
540  PRINT : PRINT : PRINT
550  PRINT "TEACHER   ";NUMB;"  IS NOT IN THE DATA BASE"
560  PRINT : PRINT : PRINT
570  INPUT "PRESS THE RETURN KEY TO CONTINUE";ZZ$
580  FTEACH = 0
590  GOTO 310
600  REM  EXIT PROGRAM AND RUN MENU
610  PRINT D$;"RUN MENU,D1"
620  REM *****************************************************
630  REM  SUBROUTINE TO SET BYTE POINTERS
640  REM *****************************************************
650  B1 = 10
660  B2 = 13
670  RETURN
680  REM *****************************************************
690  REM  SUROUTINE TO SET BYTE POINTERS FOR FINIAL GRADES
700  REM *****************************************************
710  B1 = 15
720  B2 = 18
730  RETURN
740  REM *****************************************************
750  REM  SUBROUTINE TO READ TEACHER-INDEX"
760  REM *****************************************************
770  PRINT D$;"OPEN TEACHER-INDEX,D2"
780  PRINT D$;"READ TEACHER-INDEX"
790  INPUT NTES: INPUT TP: INPUT OBAL
800  IF NTEA = 0 THEN  GOTO 850
810  FOR I = 1  TO NTEA
820  FOR J = 1  TO 4
830  INPUT TINDX(I,J)
840  NEXT J: NEXT I
850  PRINT D$;"CLOSE TEACHER-INDEX"
860  RETURN
870  REM *****************************************************
880  REM  SUBROUTINE TO READ CLASS-INDEX"
890  REM *****************************************************
900  PRINT D$;"OPEN CLASS-INDEX"
910  PRINT D$;"READ CLASS-INDEX"
920  INPUT NCLASS: INPUT CP: INPUT OCBAL
930  IF NCLASS = 0 THEN  GOTO 980
```

```
940  FOR I = 1  TO NCLASS
950  FOR J = 1  TO 4
960  INPUT CINDX(I,J)
970  NEXT J: NEXT I
980  PRINT D$;"CLOSE CLASS-INDEX"
990  RETURN
1000  REM ************************************************
1010  REM  SUBROUTINE TO READ STUDENT-INDEX"
1020  REM ************************************************
1030  PRINT D$;"OPEN STUDENT-INDEX"
1040  PRINT D$;"READ STUDENT-INDEX"
1050  INPUT NST: INPUT SP: INPUT OBAL
1060  IF NSTU = 0 THEN  GOTO 1110
1070  FOR I = 1  TO NSTU
1080  FOR J = 1  TO 4
1090  INPUT SNDX(I,J)
1100  NEXT J: NEXT I
1110  PRINT D$;"CLOSE ST-INDEX"
1120  RETURN
1130  REM ************************************************
1140  REM  SUBROUTINE TO FIND TEACHER RECORD"
1150  REM ************************************************
1160  PT = TP
1170  IF PT = 0 THEN  GOTO 1230
1180  IF NUMB < TINDX(PT,2)  THEN IK = 1
1190  IF NUMB > TINDX(PT,2)  THEN IK = 3
1200  IF NUMB = TINDX(PT,2)  THEN  GOTO 1260
1210  PT = TINDX(PT,IK)
1220  GOTO 1170
1230  REM  TEACHER IS NOT IN DATA BASE
1240  FTEAC = 1
1250  RETURN
1260  REC = TINDX(PT,4)
1270  RETURN
1280  REM ************************************************
1290  REM  SUBROUTINE TO READ TEACHER SCHEDULE
1300  REM ************************************************
1310  PRINT D$;"OPEN TEACHER-SCHEDULE,L31"
1320  FOR J = 1  TO 7
1330  BYT = 3 + (J - 1) * 4
1340  PRINT D$;"READ TEACHER-SCHEDULE,R";REC;",B";BYT
1350  INPUT SCHD(J)
1360  NEXT J
1370  PRINT D$;"CLOSE TEACHER-SCHEDULE"
1380  RETURN
1382  REM
1385  REM
```

```
1387  REM
1390  REM **************************************************
1400  REM  SUBROUTINE TO FIND CLASS RECORD
1410  REM  ************************************************
1420  PT = CP
1430  IF PT = 0 THEN  GOTO 1490
1440  IF NUMB < CINDX(PT,2)  THEN IK = 1
1450  IF NUMB > CINDX(PT,2)  THEN IK = 3
1460  IF NUMB = CINDX(PT,2)  THEN 1510
1470  PT = CINDX(PT,IK)
1480  GOTO 1430
1490  REM  CLASS IS NOT IN DATA BASE
1500  FLAG = 1
1510  REC = CINDX(PT,4)
1520  RETURN
1530  REM **************************************************
1540  REM  SUBROTINE TO READ CLASS-LIST
1550  REM **************************************************
1560  PRINT D$;"OPEN CLASS-INFO,L225"
1570  PRINT D$;"READ CLASS-INFO,R";REC;",B0"
1580  INPUT INFO(1)
1590  PRINT D$;"READ CLASS-INFO,R";REC;",B4"
1600  INPUT INFO(2)
1610  PRINT D$;"READ CLASS-INFO,R";REC;",B9"
1620  INPUT INFO(4)
1630  PRINT D$;"READ CLASS-INFO,R";REC;",B7"
1635  INPUT INFO(3)
1640  PRINT D$;"READ CLASS-INFO,R";REC;",B12"
1650  INPUT INFO(5)
1660  IF INFO(5)  = 0 THEN  1740
1670  FOR I = 1  TO INFO(5)
1680  BYT = 15 + (I - 1)  * 6
1690  PRINT D$;"READ CLASS-INFO,R";REC;",B";BYT
1700  INPUT LST(I)
1710  NEXT I
1720  GOTO 1740
1730  FLAG = 1
1740  PRINT D$;"CLOSE CLASS-INFO"
1750  RETURN
1760  REM **************************************************
1770  REM  SUBROUTINE TO FIND STU-SCH RECORDS
1780  REM **************************************************
1790  FOR I = 1  TO INFO(5)
1800  PT = SP
1810  IF PT = 0 THEN  GOTO 1890
1820  IF LST(I)  < SNDX(PT,2)  THEN IK = 1
1830  IF LST(I)  > SNDX(PT,2)  THEN IK = 3
```

```
1840  IF LST(I) = SNDX(PT,2) THEN 1870
1850  PT = SNDX(PT,IK)
1860  GOTO 1810
1870  SREC(I) = SNDX(PT,4)
1880  GOTO 1900
1890  SREC(I) =  - 9
1900  NEXT I
1910  RETURN
1920  REM  ************************************************
1930  REM  SUBROUTINE TO INPUT AND PRINT GRADES
1940  REM  ************************************************
1950  HOME : PRINT : PRINT
1960  PRINT "CLASS  TEACHER  PERIOD  RM#  CSIZE"
1970  PRINT
1980  FOR I = 1 TO 5
1990  PRINT INFO(I);"        ";
2000  NEXT I
2010  PRINT : PRINT
2020  IF INFO(5) = 0 THEN  GOTO 2320
2030  PRINT "ENTER GRADE,COMMENT"
2040  PRINT
2050  PRINT "STUDENT  GRADE,COMMENT"
2060  PRINT
2070  FOR J = 1 TO INFO(5)
2080  PRINT LST(J);"      ";
2090  INPUT GDE(J),COM(J)
2100  PRINT
2110  NEXT J
2120  PRINT D$;"OPEN STUDENT-SCHEDULE,L104"
2130  REM  WRITE TO STUDENT-SCHEDULE FILE
2140  FOR I = 1 TO INFO(5)
2150  PT = SP
2160  IF PT = 0 THEN  GOTO 2230
2170  IF LST(I) < SNDX(PT,2) THEN IK = 1
2180  IF LST(I) = SNDX(PT,2) THEN  GOTO 2220
2190  IF LST(I) > SNDX(PT,2) THEN IK = 3
2200  PT = SNDX(PT,IK)
2210  GOTO 2160
2220  SREC(I) = SNDX(PT,4)
2230  NEXT I
2240  FOR I = 1 TO INFO(5)
2250  BYT = B1 + (INFO(2)-1) * 14
2260  PRINT D$;"WRITE STUDENT-SCHEDULE,R";SREC(I);",B";BYT"
2270  PRINT GDE(I)
2280  BYT = B2 + (INFO(2) - 1) * 14
2290  PRINT D$;"WRITE STUDENT-SCHEDULE,R";SREC(I);",B";BYT"
2300  PRINT COM(I)
```

```
2310  NEXT I
2320  PRINT D$;"CLOSE STUDENT-SCHEDULE"
2330  INPUT "PRESS RETURN FOR THE NEXT CLASS";ZZ$
2340  RETURN
```

APPENDIX G

PROGRAM PRINT GRADES

```
100  REM
110  REM  PROGRAM PRINT-GRADES
120  REM
130  REM
140  D$ =  CHR$ (4)
150  DIM SNDX(500,4)
160  DIM CINDX(150,4)
170  DIM SCHD(7)
180  DIM TCH(7)
190  DIM GMT(7),GF(7)
200  DIM CMT(7),CF(7)
210  DIM CREC(7)
220  DIM STACK(500,2)
230  DIM ODR(500)
240  FLAG = 0
250  GOSUB 730
260  GOSUB 880
270  IF FLAG = 1 THEN  GOTO 580
280  HOME : PRINT : PRINT
290  PRINT "THIS PROGRAM PRINTS THE: "
300  PRINT : PRINT : PRINT
310  PRINT "1......MID-SEMESTER GRADES."
320  PRINT : PRINT : PRINT
330  PRINT "2.....FINIAL GRADES."
340  PRINT : PRINT : PRINT
350  PRINT "3.....TO END PROGRAM"
360  PRINT : PRINT : PRINT
370  INPUT "ENTER THE NUMBER OF YOUR CHOICE: ";ANS
380  IF ANS = 3 THEN  GOTO 630
390  IF ANS <  > 1 AND ANS <  > 2 THEN  GOTO 370
400  GOSUB 650
410  HOME : PRINT : PRINT : PRINT
420  PRINT "DO YOU WANT GRADES FOR:  "
430  PRINT : PRINT
440  PRINT "1.....ONE STUDENT"
450  PRINT : PRINT
460  PRINT "2.....ALL STUDENTS"
470  PRINT : PRINT
```

```
475  INPUT "ENTER THE NUMBER OF YOUR CHOICE:  ";BANS
480  IF BANS = 1 THEN  GOSUB 1030
490  IF BANS = 2 THEN  GOSUB 1200
500  IF BANS <  > 1 AND BANS <  > 2 THEN  GOTO 420
510  HOME : PRINT : PRINT
520  PRINT "DO YOU WANT TO RETRIEVE MORE"
530  PRINT : PRINT "     STUDENT'S GRADES  (Y/N)"
540  PRINT : PRINT : INPUT C$
550  IF C$ = "Y" THEN  GOTO 410
560  IF C$ = "N" THEN  GOTO 280
570  GOTO 510
580  REM  EXIT PROGRAM AND RUN MENU PROGRAM
590  HOME : PRINT : PRINT
600  PRINT "THE CLASS INDEX HAS NOT BEEN BUILT"
610  PRINT:PRINT:PRINT " USE BUILD PROGRAM TO ENTER CLASSES"
620  PRINT : PRINT : INPUT "PRESS RETURN TO CONTINUE  ";ZZ$
630  REM  EXIT PROGRAM AND RUN MENU
640  PRINT D$;"RUN MENU,D1"
650  REM *************************************************
660  REM  SUBROUTINE SET BYTE-POINTERS
670  REM *************************************************
680  B1 = 10
690  B2 = 13
700  B3 = 15
710  B4 = 18
720  RETURN
730  REM *************************************************
740  REM  SUBROUTINE READ STUDENT-INDEX
750  REM *************************************************
760  PRINT D$;"OPEN STUDENT-INDEX,D2"
770  PRINT D$;"READ STUDENT-INDEX"
780  INPUT NST: INPUT SP: INPUT OBAL
790  IF NST = 0 THEN  GOTO 850
800  FOR I = 1 TO NST
810  FOR J = 1 TO 4
820  INPUT SNDX(I,J)
830  NEXT J: NEXT I
840  GOTO 860
850  FLAG = 1
860  PRINT D$;"CLOSE STUDENT-INDEX"
870  RETURN
880  REM *************************************************
890  REM  SUBROUTINE READ CLASS-INDEX"
900  REM *************************************************
910  PRINT D$;"OPEN CLASS-INDEX"
920  PRINT D$;"READ CLASS-INDEX"
```

```
930  INPUT NCLASS: INPUT CP: INPUT OCBAL
940  IF NCLASS = 0 THEN  GOTO 1000
950  FOR I = 1 TO NCLASS
960  FOR J = 1 TO 4
970  INPUT CINDX(I,J)
980  NEXT J: NEXT I
990  GOTO 1010
1000  FLAG = 1
1010  PRINT D$;"CLOSE CLASS-INDEX"
1020  RETURN
1030  REM **************************************************
1040  REM  SUBROUTINE PRINT-GRADES (ONE STUDENT)
1050  REM **************************************************
1060  HOME : PRINT
1070  INPUT "ENTER THE STUDENT NUMBER  ";NUMB
1080  FLAG = 0
1090  GOSUB 1310
1100  IF FLAG = 1 THEN  GOTO 1150
1110  GOSUB 1450
1120  IF ANS = 1 THEN  GOSUB 2340
1130  IF ANS = 2 THEN  GOSUB 2480
1140  GOTO 1190
1150  HOME : PRINT : PRINT
1160  PRINT " STUDENT  ";NUMB;"  IS NOT IN THE DATA BASE"
1170  PRINT : PRINT : PRINT
1180  INPUT "PRESS RETURN TO CONTINUE F ";ZZ$
1190  RETURN
1200  REM **************************************************
1210  REM  SUBROUTINE PRINT-GRADES (ALL STUDENTS)
1220  REM **************************************************
1230  GOSUB 1950
1240  FOR JJ = 1 TO NST
1250  REC = SNDX(ODR(JJ) ,4)
1260  GOSUB 1450
1270  IF ANS = 1 THEN  GOSUB 2340
1280  IF ANS = 2 THEN  GOSUB 2480
1290  NEXT JJ
1300  RETURN
1310  REM **************************************************
1320  REM  SUBROUTINE READ STUDENT-RECORD
1330  REM **************************************************
1340  PT = SP
1350  IF PT = 0 THEN  GOTO 1430
1360  IF NUMB < SNDX(PT,2) THEN IK = 1
1370  IF NUMB > SNDX(PT,2) THEN IK = 3
1380  IF NUMB = SNDX(PT,2) THEN  GOTO 1410
1390  PT = SNDX(PT,IK)
```

```
1400  GOTO 1350
1410  REC = SNDX(PT,4)
1420  GOTO 1440
1430  FLAG = 1
1440  RETURN
1450  REM ************************************************
1460  REM  SUBROUTINE READ-GRADES
1470  REM ************************************************
1480  PRINT D$;"OPEN STUDENT-SCHEDULE,L104"
1490  PRINT D$;"READ STUDENT-SCHEDULE,R";REC;",B0"
1500  INPUT NUMB
1510  FOR I = 1 TO 7
1520  BYT = 6 + (I - 1) * 14
1530  PRINT D$;"READ STUDENT-SCHEDULE,R";REC;",B";BYT
1540  INPUT SCHD(I)
1550  NEXT I
1560  PRINT D$;"CLOSE STUDENT-SCHEDULE"
1570  GOSUB 1720
1580  PRINT D$;"OPEN STUDENT-SCHEDULE,L104"
1590  FOR I = 1 TO 7
1595  BYT = B1 +(I-1)*14
1600  PRINT D$;"READ STUDENT-SCHEDULE,R";REC;",B";BYT
1610  INPUT GMT(I)
1615  BYT = B2 + (I-1) * 14
1620  PRINT D$;"READ STUDENT-SCHEDULE,R";REC;",B";BYT
1630  INPUT CMT(I)
1640  IF ANS = 1 THEN  GOTO 1690
1645  BYT = B3 + (I-1) * 14
1650  PRINT D$;"READ STUDENT-SCHEDULE,R";REC;",B";BYT
1660  INPUT GF(I)
1665  BYT = B4 + (I-1) * 14
1670  PRINT D$;"READ STUDENT-SCHEDULE,R";REC;",B";BYT
1680  INPUT CF(I)
1690  NEXT I
1700  PRINT D$;"CLOSE STUDENT-SCHEDULE"
1710  RETURN
1720  REM ************************************************
1730  REM  SUBROUTINE READ CLASS-DATA
1740  REM ************************************************
1750  PRINT D$;"OPEN CLASS-INFO,L225"
1760  FOR I = 1 TO 7
1770  CNUMB = SCHD(I)
1780  PT = CP
1790  IF PT = 0 THEN  GOTO 1910
1800  IF CNUMB < CINDX(PT,2) THEN IK = 1
1810  IF CNUMB > CINDX(PT,2) THEN IK = 3
1820  IF CNUMB = CINDX(PT,2) THEN  GOTO 1850
```

```
1830  PT = CINDX(PT,IK)
1840  GOTO 1790
1850  CREC = CINDX(PT,4)
1860  PRINT D$;"READ CLASS-INFO,R";CREC;",B4"
1870  INPUT TCH(I)
1880  PRINT D$;"READ CLASS-INFO,R";CREC;",B9"
1890  INPUT RM(I)
1900  GOTO 1920
1910  TCH(I) =  - 1:RM(I) =  - 1
1920  NEXT I
1930  PRINT D$;"CLOSE CLASS-INFO"
1940  RETURN
1950  REM *********************************************
1960  REM  SUBROUTINE IN-ORDER
1970  REM *********************************************
1980  PT = SP
1990  IF SNDX(PT,1) <  > 0 THEN  GOTO 2110
2000  LN = LN + 1
2010  ODR(LN) = PT
2020  D = 1
2030  GOSUB 2170
2040  D = 0
2050  PT = SNDX(PT,3)
2060  IF PT <  > 0 THEN  GOTO 1990
2070  GOSUB 2240
2080  IF D = 3 THEN  GOTO 2160
2090  IF D = 1 THEN  GOTO 2070
2100  GOTO 2000
2110  D = 2
2120  GOSUB 2170
2130  D = 0
2140  PT = SNDX(PT,1)
2150  GOTO 1990
2160  RETURN
2170  REM *********************************************
2180  REM  SUBROUTINE PUSH-STACK
2190  REM *********************************************
2200  LST = LST + 1
2210  STACK(LST,1) = PT
2220  STACK(LST,2) = D
2230  RETURN
2240  REM *********************************************
2250  REM  SUBROUTINE PUSH-STACK
2260  REM *********************************************
2270  IF LST = 0 THEN  GOTO 2320
2280  PT = STACK(LST,1)
2290  D = STACK(LST,2)
```

```
2300  LST = LST - 1
2310  GOTO 2330
2320  D = 3
2330  RETURN
2335  REM
2340  REM *********************************************
2350  REM  SUBROUTINE PRINT MID-SEMESTER GRADES
2360  REM *********************************************
2370  HOME : PRINT
2380  PRINT "MID-SEMSESTER GRADES FOR STUDENT: ";NUMB
2390  PRINT : PRINT
2400  PRINT "PER";TAB( 8);"CLASS";TAB( 16);"TEA"; TAB( 22);
2405  PRINT "RM"; TAB( 27);"GR"; TAB( 32);"COM"
2410  FOR I = 1 TO 7
2420  PRINT
2430  PRINT  TAB( 1)I; TAB( 9)SCHD(I); TAB( 16)TCH(I);
2435  PRINT TAB( 22)RM(I); TAB( 27)GMT(I); TAB( 32)CMT(I)
2440  NEXT I
2450  PRINT : PRINT
2460  INPUT "PRESS RETURN TO CONTINUE:  ";ZZ$
2470  RETURN
2480  REM *********************************************
2490  REM  SUBROUTINE TO PRINT FINIAL-GRADES
2500  REM *********************************************
2510  HOME : PRINT
2520  PRINT "FINIAL GRADES FOR STUDENT:  ";NUMB
2530  PRINT : PRINT
2540  PRINT  TAB( 22)"MID"; TAB( 26)"MID"; TAB( 33)"FIN";
2545  PRINT TAB( 38)"FIN"
2550  PRINT "PER"; TAB( 6)"CLASS"; TAB( 13)"TEA"; TAB( 18)
2555  PRINT "RM";TAB( 22)"GR"; TAB( 26)"COM"; TAB( 33)"GR";
2558  PRINT TAB( 38)"COM"
2560  FOR I = 1 TO 7
2570  PRINT
2580  PRINT  TAB( 2)I; TAB( 7)SCHD(I); TAB( 13)TCH(I);
2585  PRINT TAB( 18)RM(I); TAB( 22)GMT(I); TAB( 27)CMT(I);
2587  PRINT TAB( 33)GF(I); TAB( 39)CF(I)
2590  NEXT I
2600  PRINT : PRINT
2610  INPUT "PRESS RETURN TO CONTINUE:  ";ZZ$
2620  RETURN
```

APPENDIX H

PROGRAM RETRIEVE CLASS LISTS LISTING

```
100  REM  PROGRAM TO RETREIVE CLASS LIST
110  REM
120  DIM CINDX(500,4)
130  DIM TINDX(50,4)
140  DIM INFO(5)
150  DIM LST(35)
160  DIM STACK(150,2)
170  DIM ODR(150)
180  D$ =  CHR$ (4)
190  GOSUB 440
200  GOSUB 540
210  HOME : PRINT : PRINT
220  PRINT "THIS PROGRAM WILL FIND CLASS LIST FOR:"
230  PRINT : PRINT
240  PRINT "1.....CLASS LIST (ONE CLASS)
250  PRINT : PRINT
260  PRINT "2.....CLASS LISTS (ONE TEACHER)"
270  PRINT : PRINT
280  PRINT "3.....CLASS LISTS (ALL CLASSES)"
290  PRINT : PRINT
300  PRINT "4.....TO EXIT PROGRAM"
310  PRINT : PRINT
320  PRINT : PRINT
330  INPUT "THE NUMBER OF YOUR CHOICE";ANS
340  IF ANS = 1 THEN  GOSUB 670
350  IF ANS = 2 THEN  GOSUB 820
360  IF ANS = 3 THEN  GOSUB 1160
370  IF ANS = 4 THEN  GOTO 390
380  GOTO 210
390  REM  EXIT PROGRAM AND RUN MENU
400  PRINT D$;"RUN MENU,D1"
410  REM *************************************************
420  REM  SUBROUTINE READ TEACHER INDEX
430  REM *************************************************
440  PRINT D$;"OPEN TEACHER-INDEX,D2"
450  PRINT D$;"READ TEACHER-INDEX"
460  INPUT NTEA: INPUT S1P: INPUT KOBAL
470  IF NTEA = 0 THEN  GOTO 520
```

```
480  FOR I = 1 TO NTEA
490  FOR J = 1 TO 4
500  INPUT TINDX(I,J)
510  NEXT J: NEXT I
520  PRINT D$;"CLOSE TEACHER-INDEX"
530  RETURN
540  REM **************************************************
550  REM  SUBROUTINE TO READ CLASS-INDEX
560  REM **************************************************
570  PRINT D$;"OPEN CLASS-INDEX"
580  PRINT D$;"READ CLASS-INDEX"
590  INPUT NCLASS: INPUT S2P: INPUT OCBAL
600  IF NCLASS = 0 THEN  GOTO 650
610  FOR I = 1 TO NCLASS
620  FOR J = 1 TO 4
630  INPUT CINDX(I,J)
640  NEXT J: NEXT I
650  PRINT D$;"CLOSE CLASS-INDEX"
660  RETURN
670  REM **************************************************
680  REM  SUBROUTINE FOR CLASS
690  REM **************************************************
700  HOME : PRINT : PRINT
710  INPUT "CLASS NUMBER:   ";NUMB
720  PT = S2P
730  GOSUB 1860
740  IF FLAG = 1 THEN  GOTO 780
750  GOSUB 1470
760  GOSUB 1710
770  GOTO 810
780  PRINT : PRINT : PRINT "CLASS IS NOT IN THE DATA BASE"
790  FLAG = 0
800  PRINT:PRINT
850  INPUT "PRESS THE RETURN KEY TO CONTINUE:";ZZ$
810  RETURN
820  REM **************************************************
830  REM  TEACHER SUBROUTINE
840  REM **************************************************
850  HOME : PRINT : PRINT
860  PRINT "ENTER TEACHER NUMBER"
870  PRINT : PRINT : INPUT NUMB
880  PT = S1P
890  IF PT = 0 THEN  GOTO 1110
900  IF NUMB < TINDX(PT,2) THEN IK = 1
910  IF NUMB > TINDX(PT,2) THEN IK = 3
920  IF NUMB = TINDX(PT,2) THEN 950
```

```
930  PT = TINDX(PT,IK)
940  GOTO 890
950  REC = TINDX(PT,4)
960  PRINT D$;"OPEN TEACHER-SCHEDULE,L31"
970  FOR I = 1 TO 7
980  BYT = 3 + (I - 1) * 4
990  PRINT D$;"READ TEACHER-SCHEDULE,R";REC;",B";BYT
1000  INPUT SCHD(I)
1010  NEXT I
1020  FOR JJ = 1 TO 7
1030  PT = S2P
1040  NUMB = SCHD(JJ)
1050  IF NUMB = 999 THEN  GOTO 1090
1060  GOSUB 1860
1070  GOSUB 1470
1080  GOSUB 1710
1090  NEXT JJ
1100  GOTO 1150
1110  PRINT : PRINT : PRINT
1120  PRINT " TEACHER IS NOT IN THE DATA BASE"
1130  PRINT : PRINT
1140  INPUT "PRESS THE RETURN KEY TO CONTINUE:     ";ZZ$
1150  RETURN
1160  REM **********************************************
1170  REM  PROCESS  ALL TEACERS-ALL CLASSES
1180  REM **********************************************
1190  IF NTEA = 0 THEN  GOTO 1390
1200  GOSUB 1990
1210  FOR IJ = 1 TO NTEA
1220  PRINT D$;"OPEN TEACHER-SCHEDULE,L31"
1230  FOR IK = 1 TO 7
1240  BYT = 3 + (IK - 1)  * 4
1250  PRINT D$;"READ TEACHER-SCHEDULE,R";ODR(IJ);",B";BYT
1260  INPUT SCHD(IK)
1270  NEXT IK
1280  PT = S2P
1290  FOR J = 1 TO 7
1300  NUMB = SCHD(J)
1310  IF NUMB = 999 THEN  GOTO 1350
1320  GOSUB 1860
1330  GOSUB 1470
1340  GOSUB 1710
1350  NEXT J
1360  NEXT IJ
1370  PRINT D$;"CLOSE TEACHER-SCHEDULE"
1380  GOTO 1450
1390  HOME : PRINT : PRINT : PRINT
```

```
1400  PRINT "THE TEACHER INDEX HAS NOT BEEN BIULD"
1410  PRINT : PRINT : PRINT
1420  PRINT "BUILD TEACHER-INDEX USING BUILD PROGRAM.
1430  PRINT : PRINT : PRINT
1440  INPUT "PRnESS RETURN TO CONTINUE ";ZZ$
1450  RETURN
1460  D$ =  CHR$ (4)
1470  REM *************************************************
1480  REM  SUBROUTINE TO READ CLASS RECORD
1490  REM  ************************************************
1500  PRINT D$;"OPEN CLASS-INFO,L225"
1510  PRINT D$;"READ CLASS-INFO,R";REC;",B0"
1520  INPUT INFO(1)
1530  PRINT D$;"READ CLASS-INFO,R";REC;",B4"
1540  INPUT INFO(2)
1550  PRINT D$;"READ CLASS-INFO,R";REC;",B7"
1560  INPUT INFO(3)
1570  PRINT D$;"READ CLASS-INFO,R";REC;",B9"
1580  INPUT INFO(4)
1590  PRINT D$;"READ CLASS-INFO,R";REC;",B12"
1600  INPUT INFO(5)
1610  IF INFO(5) = 0 THEN  GOTO 1670
1620  FOR I = 1 TO INFO(5)
1630  BYT = 15 + (I - 1)  * 6
1640  PRINT D$;"READ CLASS-INFO,R";REC;",B";BYT
1650  INPUT LST(I)
1660  NEXT I
1670  PRINT D$;"CLOSE"
1680  GOTO 1700
1690  FLAG = 1
1700  RETURN
1710  REM *************************************************
1720  REM  DISPLAY SUBROUTINE
1730  REM *************************************************
1740  HOME
1750  PRINT "CLASS"; TAB( 9)"TEACHER"; TAB( 19)"PERIOD";
1755  PRINT TAB( 28)"RM#"; TAB( 34)"CSIZE"
1760  PRINT : PRINT
1770  PRINT  TAB( 1)INFO(1); TAB( 11)INFO(2);
1773  PRINT TAB( 21)INFO(3);
1775  PRINT TAB( 28)INFO(4); TAB( 35)INFO(5)
1780  PRINT : PRINT
1790  IF INFO(5) = 0 THEN  GOTO 1840
1800  FOR I = 1 TO INFO(5)
1810  PRINT LST(I)
1820  NEXT I
1830  PRINT : PRINT
```

```
1840  INPUT "PRESS THE RETURN KEY TO CONTINUE:   ";ZZ$
1850  RETURN
1860  REM *************************************************
1870  REM  SUBROUTINE TO FIND RECORD NUMBER"
1880  REM *************************************************
1890  IF PT = 0 THEN  GOTO 1950
1900  IF NUMB < CINDX(PT,2) THEN IK = 1
1910  IF NUMB > CINDX(PT,2) THEN IK = 3
1920  IF NUMB = CINDX(PT,2) THEN 1970
1930  PT = CINDX(PT,IK)
1940  GOTO 1890
1950  REM  NUMB IS NOT IN INDEX
1960  FLAG = 1
1970  REC = CINDX(PT,4)
1980  RETURN
1990  REM *************************************************
2000  REM  SUBROUTINE IN ORDER
2010  REM *************************************************
2020  LST = 0:PT = S1P
2030  IF TINDX(PT,1) <  > 0 THEN  GOTO 2190
2040  PRINT PT
2050  IY = IY + 1
2060  ODR(IY) = TINDX(PT,4)
2070  PRINT ODR(IY),IY
2080  D = 1
2090  REM  PUST THE STACK
2100  GOSUB 2270
2110  D = 0
2120  PT = TINDX(PT,3)
2130  IF PT <  > 0 THEN  GOTO 2030
2140  REM  POOP THE STACK
2150  GOSUB 2340
2160  IF D = 3 THEN  GOTO 2240
2170  IF D = 1 THEN  GOTO 2140
2180  GOTO 2050
2190  D = 2
2200  GOSUB 2270
2210  D = 0
2220  PT = TINDX(PT,1)
2230  GOTO 2030
2240  FOR KK = 1 TO 7
2250  PRINT ODR(KK);
2260  NEXT KK
2270  REM ************************************************
2280  REM  SUBROUTINE PUSH
2290  REM ************************************************
2300  LST = LST + 1
```

```
2310  STACK(LST,1) = PT
2320  STACK(LST,2) = D
2330  RETURN
2340  REM **************************************************
2350  REM  SUBROUTINE POP
2360  REM **************************************************
2370  IF LST = 0 THEN  GOTO 2420
2380  PT = STACK(LST,1)
2390  D = STACK(LST,2)
2400  LST = LST - 1
2410  GOTO 2430
2420  D = 3
2430  RETURN
```

REFERENCES

(1) DATE C. J.: An Introduction to Data Base System. Allyn and Bacon, 1979.

(2) Riley M. J. : Management Information Systems. Holden-Day, 1981.

(3) Scott, George M.: A Data Base for Your Company? California Management Review: 13/1, 1976.

√(4) Hutt A. T. F.: A Relational Data Base: Management Sytem. John Wiley & Sons Ltd., 1979.

(5) Codd E. F.: A Relational Model of Data for Large Shared Data Banks. Communications of the ACM: 13/6, June 1970.

(6) Alfonso F. Cardenas: Data Base Management Systems: Allyn and Bacon, 1979.