1985

A Student scheduling system for a microcomputer

Michael Mahaney

Rochester Institute of Technology

School of Computer Science and Technology

A Student Scheduling System for a Microcomputer

by

Michael C. Mahaney Jr.

A thesis, submitted to

The Faculty of the School of Computer Science and Technology,

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science.

Approved by:

Henry A. Etlinger

Henry A. Etlinger (Advisor)

Chris Comte

Chris Comte

Andrew Kitchen

Andrew Kitchen

May 1, 1985

Title of Thesis: A Student Scheduling System for a Microcomputer

Date: May 20, 1985

ACKNOWLEDGMENTS

I would like to express gratitude to the members of my thesis committee, Chris Comte and Andrew Kitchen for their suggestions and criticisms. I am especially indebted to Hank Etlinger, chairperson of my committee, for help in reviewing and revising this paper.

Most of all I would like to thank my family, who provided support and understanding during the past four years.

1. PRELIMINARY INFORMATION

1.2. ABSTRACT

A STUDENT SCHEDULING SYSTEM FOR A MICROCOMPUTER

This student scheduling system was written for use on a microcomputer to give the local high school more control over the scheduling process. Scheduling systems are used by high schools to schedule their students, balance classes, and print student schedules and class lists. A scheduling system must be flexible to provide for the generation of many different types of schedules. Most scheduling programs currently used are shared by high schools through the use of BOCES (Board of Cooperative Educational Services). Schools send their schedules to the BOCES regional computer centers to be run on mainframe computers.

This scheduling system was written using Apple Pascal for use on an Apple IIE microcomputer. The system was developed to do student scheduling for a high school of less than 1200 students. The system will section students and provide the school with student schedules, class lists as well as many scheduling tools which are useful in the development of the schedule.

1.3. KEY WORDS AND PHRASES

Balancing (Class size) - Making the number of students in each section (or class) of a course the same, or as nearly so, as possible.

Block Scheduling - Placement of students with common subject selections into a group. The group then could be assigned to a team of teachers for large segments of instructional time.

Code Number - A three-digit number identifying each course offered in a school.

Computer Scheduling - A general term describing the use of a computer to assist in student scheduling. The range of assistance extends from the preparation of simple reports or printouts to the complete processing of both the master schedule and student schedules.

Conflict - A condition in which two or more courses selected by a student are offered at the same time.

Conflict Matrix - A list of all courses that would conflict with a given course. The number of students who would have this conflict is also listed.

Conflict Rate - The percentage of students who have one or more conflicts.

Conventional Schedule - A traditional scheduling method based on a weekly cycle of class periods of equal length , usually meeting the same time each day.

Course Tallies - Enrollment figures for each course offered in the curriculum.

Doubleton - A course which has sufficient enrollment to require two sections.

Flexible Modular Schedule - A type of schedule which divides the school day into 15 to 30 abbreviated periods (10 to 20 minutes each) in order to provide for a greater variety of instructional experiences.

Manual Scheduling - The completion of the scheduling process by hand, without computerized data processing assistance.

Master Schedule - A comprehensive listing of the class offerings that assigns every section of every course to specific teachers in designated rooms at various times of the day.

Mosaic Scheduling - A loading technique used to schedule student courses individually rather that in block fashion.

Scheduling - The process by which the curriculum, the students, and the teachers are brought together in an organized manner.

Section - A class of students of a predetermined number range that study a particular course together in the same period.

Simulation Run - A computerized testing of the efficiency of the master schedule in which problems or possible solutions are identified.

Singleton - A course which has only one section.

Student Requests - A list of courses that a student would like to schedule.

Turnaround Time - The elapsed time between the school submitting information to the computer center and the delivery of the resulting computer printout to the school.

Tripleton - A course that has sufficient enrollment to require three distinct sections.

1.4. COMPUTING REVIEW SUBJECT CODE

Not applicable.

1.5. TABLE OF CONTENTS

10. Program Listings

2. PROPOSAL

A copy of the thesis proposal is on file.

3. INTRODUCTION AND BACKGROUND

3.1 PROBLEM STATEMENT

Every high school faces the problem each year of scheduling their students and teachers. The most widely used scheduling format is one in which the computer generates data from which the master schedule is built. The master schedule is a listing that assigns every section of every course to a specific teacher at a designated time of the day. The data that the computer generates includes a course tallies list and the conflict matrix. The course tallies list is a list of enrollment figures for each course in the curriculum. The conflict matrix is a list of all courses that would conflict with a given course. The number of students who would have each conflict are also listed.

After the schedule is built the computer tests the efficiency of the master schedule through a simulation run in which the students are loaded into the schedule. The computer then prints partial schedules, conflicts and course tallies so that the scheduler can make adjustments in the schedule. A partial schedule is a student schedule for all courses which where successfully scheduled along with a list of unscheduled courses. Finally student and teacher schedules, class lists and other helpful reports are printed.

The computer usually is a mainframe located at BOCES (Board of Cooperative Educational Services) and all the schools in a given region would use the same scheduling services. This form of centralization causes some problems for local schools.

Problems encountered by schools include:

1.) BOCES allows a school only three runs of a schedule with a two to three week turnaround time for each run. As a result the scheduling process starts in May and the final run is usually in early August.

2.) After the first run, if the scheduler finds a problem with a course placement he/she corrects it. The scheduler must then wait for two to three weeks to find out if his/her solution works. The scheduler only has two chances or runs to resolve all the problems in a schedule.

3.) Any student who enters after the final run (first week in August) has to be hand scheduled and all class lists must be updated.

4.) Any student who changes a course during the first week in August or the first few weeks of school has an incorrect schedule, resulting in class lists which are often inaccurate.

SOLUTION

This thesis will solve the problems listed above by developing a student scheduling system for a microcomputer. Since most schools have a microcomputer, the control of the scheduling process could be held in the local school.

Difficulties previously encountered that could be solved by this system are:

1.) Eliminating the two to three week turnaround time between runs of the schedule.

2.) The scheduler would not be limited to only three runs which would greatly increase his/her efficiency and accuracy.

3.) The first run could be made before the end of school in June so guidance counselors would have time to contact students regarding conflicts in the students' schedules.

4.) The final run of schedules could be made during the week before school begins to ensure more accurate student schedules and class lists.

THE STUDENT SCHEDULING SYSTEM

This student scheduling system will allow the school to enter all required data using a microcomputer. The computer will then be able to generate the information necessary to build the schedule. Finally the schedule and final reports will be printed.

1.) Data entered by the user.

A.) Student information

1.) Name, grade, address.

2.) Courses requested.

B.) Schedule information

1.) Courses offered.

2.) Time and room information.

C.) Teacher information

1.) Courses and sections offered.

2.) Room or rooms used.

2.) Data generated by the system

A.) Information used to help build the master schedule.

1.) A conflict matrix which is used to determine how many students are taking a given combination of courses.

2.) A list of the number of students registered for each course by grade levels.

B.) After scheduling the students, the system will provide information used to help resolve conflicts.

1.) A list of students unable to schedule all requested courses.

2.) A partial schedule for each student whose schedule contained a conflict, listing any course in which a student was unable to enroll.

3.) A listing of all classes and the number of students enrolled in each. This list would be used to consider balance problems.

C.) Changes can then be made to both student course selections and course offerings. The schedule can then be rerun until the results are acceptable.

D.) After arriving at an acceptable schedule the system will print the following information:

1.) Individual student schedules for every student.

2.) Class lists.

3.) Master list of classes with enrollments.

3.2 PREVIOUS WORK

3.2.1 TYPES OF SCHEDULES

The schedule is an essential component of the school program. Scheduling combines all of a school's essential resources - faculty and staff, curriculum, space and facilities, and students - into an integrated and efficient learning environment. A successful master schedule holds curricular objectives, student course requests, and faculty strengths and preferences in appropriate balance. Some of the different types of schedules used by schools include block schedules, mosaic schedules, a combination of block and mosaic schedules, and modular or flexible modular schedules.

BLOCK SCHEDULES

Students can be scheduled in groups or individually. Block scheduling is often used when students are taking a similar set of courses. When a block schedule is used students are scheduled in groups. Block scheduling is often preferred in junior high schools in conjunction with team teaching. All students might be taking english, math, social studies, science and reading courses. Students would first be divided into groups and then the students in each group would follow the same schedule.

MOSAIC SCHEDULING

At times the block structure method is impractical, such as when large groups of students do not take similar subjects. In

high schools where required courses are offered along with a wide range of electives, a mosaic method is used. Mosaic scheduling methods are concerned with individual sections of courses. In this method the scheduler places one course section at a time into the master schedule, continually building upon the previous sections until the total mosaic is completed.

FLEXIBLE MODULAR SCHEDULING

A flexible modular schedule divides the school day into fifteen to thirty abbreviated periods which range from ten to twenty minutes each. These schedules attempt to provide a greater variety of instructional experience by the use of varying time blocks. Modular scheduling permits many alternatives that were impossible under traditional scheduling. To provide for all the individual differences required in the flexible modular schedule, computer generation of the schedule is required.

3.2.2 CONFLICTS IN SCHEDULING

No matter which type of schedule is used, conflicts can arise. Conflicts occur as a result of problems scheduling rooms, programs, and courses. The most common type of conflict arises when two of a student's preferred courses are scheduled in the same period. If a single section course (singleton) is placed in a particular period along with another singleton, the student will be unable to schedule one of them.

Conflicts do not arise so much from individual student preferences as from the difficulty of constructing a schedule that will accommodate the seemingly infinite combination of

different courses. The scheduler's goal for each scheduling year is the building of an efficient schedule that is as conflict-free as possible. This would result in only a small number of students needing to change their course requests.

In an attempt to limit potential conflicts the scheduler uses a conflict matrix (an array of all courses in the curriculum paired against each other). The matrix would usually include singletons, doubletons and some tripletons. The scheduler would read the matrix to anticipate potential conflicts. Although different implications arise when comparing two-section courses (doubletons) as opposed to singletons, the matrix maintains a high degree of usefulness.

According to Richard Dempsey and Henry Traverso[1] a number of elements affect the scheduler's ability to construct a schedule with a minimum of conflicts. Some of these are:

1) High density within the schedule (e.g. six required courses in a six - period schedule).

2) An excessive number of singletons.

3) Non-graded courses (e.g. those open to students in several grade levels).

4) A large number of restrictive factors (e.g. part time faculty).

5) Constraints in the teacher contract (e.g. length of teachers' day).

6) Many teachers assigned to more than one department.

7) Many double-period subjects (e.g. science labs).

1 Richard Dempsey and Henry Traverso, Scheduling the Secondary School (NASSP, 1983), p. 51.

As more of these factors are present, the more difficult it will be to build a conflict-reduced schedule.

Scheduling techniques have changed frequently and dramatically since the early 1960's, in keeping with changing national priorities. The increase in school enrollments and proliferation of curricular offerings in the 1960's spurred the development of computerized scheduling techniques. The number of courses offered, and the individualization of instruction resulted in a surge in the variety and flexibility of scheduling models available to schools. By the early 1970's, a growing awareness of accountability prompted educators to revitalize conventional scheduling.

3.2.3 MANUAL VS COMPUTER SCHEDULING

Many procedures employed by manual and computerized scheduling are the same. Steps requiring analysis and evaluation must be performed by the scheduler regardless of the scheduling method used. They include:

1) Determining student needs.
2) Reviewing the curriculum.
3) Formulating the program of studies.
4) Preparing registration materials.
5) Setting the calendar for registration.
6) Interpreting the tallies.
7) Identifying staffing needs.
8) Utilizing a conflict matrix.
9) Building the master schedule.

The computer is useful in preparation of the conflict matrix and providing course tallies. A conflict matrix can be prepared by hand but it is a very time consuming process. Computers are generally used in the "load" phase where the computer does not generate the master schedule but does assign students to classes. The computer also provides valuable reports and partial schedules to assist in refining the master schedule.

Junior high schools which use block scheduling techniques can manually produce their schedules while most high schools using mosaic schedules use computers to assist in the scheduling of students. There are two types of scheduling programs: student sectioning programs, and integrated class scheduling and student sectioning programs.

STUDENT SECTIONING PROGRAMS

Programs for assigning students to course sections in a pre-determined master schedule make up the majority of operational school scheduling computer programs.

One of the first operational models of this type was developed at Purdue University in 1956. I.B.M. followed this development with a program called CLASS (Class Load and Student Scheduling) based on the work at Purdue. Several other student sectioning programs have been written which are all basically quite similar. A large number of private and public organizations provide student sectioning services to schools.

Most of these models take as inputs the master class schedule, the student course requests, and class size parameters. Using various types of heuristics, students are sectioned, student schedules are printed out, and names of students for whom

the technique cannot find a no-conflict schedule are printed out. The scheduler makes some adjustments and the process is repeated.

INTEGRATED CLASS SCHEDULING AND STUDENT SECTIONING PROGRAMS

A number of research groups have written computer programs which integrate the two jobs of building the master class schedule and sectioning the students. Two models are GASP (Generalized Academic Simulation Program), developed at M.I.T., and SSSS (Stanford School Scheduling System), developed at Stanford University.

Input to these programs includes rooms and instructors available, desired time pattern for the school day and each course, and the student course requests. Using this data, a master schedule is developed and the students are sectioned. This procedure is most commonly performed by the scheduler who utilizes the conflict matrix to develop the master schedule. The main difference between these integrated class scheduling programs and programs that only "section" students is that the master schedule is computer generated.

High schools began to use these programs in the 1960's with the introduction of flexible modular scheduling. The varying time blocks and individualization provided for in a flexible modular schedule made it almost impossible to generate these schedules by hand. As a result schools started to use the services of computer companies to computer generate their schedule.

3.2.4 RESEARCH FINDINGS

COMPUTERIZED SCHEDULING

Lynne M. Durward in "Computerized Scheduling In Vancouver Schools" examined computerized scheduling in sixteen Vancouver secondary schools. Fifteen of these schools used the Honeywell Scheduling Program (a sectioning program) and one used Columbia Computing Services Limited (an integrated class and sectioning program). Durward listed the main advantages of computer scheduling as saving of secretarial time, more complete and accurate student lists and better balancing of class size.[2] Schools using the Honeywell Scheduling Program had a median conflict rate after the final scheduling run of 4.7%. The school using a flexible modular system of scheduling (the Columbia Service) had a 62% conflict rate. Durward found the 62% rate to be somewhat misleading because most of these were resolved by teachers agreeing to hold classes during their free modules to accommodate students.[3]

The report's main criticisms of the Honeywell System were that the turnaround time for simulation runs was too long and that the run dates were too early and inflexible. While she found no significant relationship between the number of simulation runs and the conflict rate it appeared that schools with extra simulation runs were operating smoothly earlier.

2 Lynne M. Durwood, "Computerized Scheduling in Vancouver Schools. A Research Report." (April 1973), p.3.

3 Ibid. , p. 10.

FLEXIBLE MODULAR SCHEDULING

Clayton Braddock in his 1967 article "Changing Times are Changing Schools" examined flexible modular scheduling. Braddock[4] listed some of the advantages of modular scheduling as:

1) More efficient use of time.

2) More opportunities for independent study and research.

3) Closer contact with teachers and other students as well as smaller groups.

4) Highly personalized class schedules for each student.

5) Greater opportunities for teachers to teach in depth.

6) More decisions by students about their own school work.

At Trezevant High School in Memphis, Tennessee, Braddock found that as much as 40% of a student's time was unscheduled.[5] Some provisions had to be made for students who were unable to handle the unscheduled time. Braddock found that many people felt that a modular schedule did not always accomplish its objectives. Dr. Frank Brown, a Superintendent of Schools in Florida, called modular scheduling a "gold plated fad." Brown felt modular scheduling took an excessive amount of administrative time and involved higher costs.

In 1965, Robert Oakford and Dwight Allen began a three year development program to determine the desirability of modular scheduling. In their report Oakford and Allen investigated the impact of the Stanford School Scheduling System (SSSS) on

4 Clayton Braddock, "Changing Times are Changing Schools," Southern Education Report , V3N3(Oct. 1967) p. 4.
5 Ibid. , p. 1.

scheduling eighteen secondary schools. They found that in many project schools the technology for flexibility is present, but teachers and administrators were not geared to implement the full potential.

Oakford and Allen found that even in the schools where staffs were deeply committed to the new educational objectives, 99 percent of the possible alternatives permitted by modular scheduling had yet to be tried. The authors stated that "only people can make a modular schedule, a flexible schedule." [6]

SECTIONING VS. INTEGRATED PROGRAMS

Project Pass (Project in Automated School Scheduling) was sponsored by the Western New York School Study Council during the summer of 1965. The goal was to provide in-service education for school personnel contemplating the use of automated approaches to school scheduling. The two techniques utilized were CLASS and GASP. Two pilot schools were selected and these schools were scheduled with both processes.

Analysis of the data showed CLASS and GASP could not be compared in terms of their ultimate objectives. CLASS is a "sectioning" technique, where as GASP develops a master schedule.

Dr. Leonard Chaffee and Dr. Robert Heller in their report on Project Pass [7] also concluded it would be financially impractical for schools with conventional programs to adopt GASP. GASP is for schools contemplating innovative instructional

6 Oakford and Allen, "Flexibility for Vocational Education Through Computer Scheduling. A Final Report," p. 11.

7 Chafee and Heller, "Analysis of Selected Factors Relative to Automated School Scheduling Process," (April 1973) p. 23.

programs which increase the number of variables to be considered in the development of the master schedule. Since it was originally designed for university scheduling it assumed more flexibility than was found in pilot schools. Many of the alternatives it presented would not have been so awkward in a higher education setting. As a result the GASP scheduler still had to build a significant portion of the schedule manually and then lock it into GASP.

SCHEDULING PROBLEMS

Robert Harding, in The Problem of Generating Class Schedules for Schools, examined the practical and theoretical approaches to generating class schedules. He found that no method was entirely satisfactory in terms of both the quality of output and the cost of the process. Harding found existing techniques suffered from one or a combination of the following weaknesses:

> 1) The process takes too long, either in total time and/or in computer time.
> 2) It costs too much.
> 3) It requires the attention of a high-level administrator for too long a time period.
> 4) It requires too much computer core capacity when compared with what is commonly available.
> 5) The schedules generated are of poor quality in terms of the following criteria:
> a) The number of students unable to schedule their requested courses.
> b) The number of teachers reassigned to areas in which they are less well qualified, to accomplish the scheduling process.
> c) Utilization of facilities.
> d) Schedule balance.
> 6) The analytical model is logically sound , but when applied to a real problem, the combinatorial properties so swell the problem that the method becomes computationally infeasible and hence not useful.
> 7) The approach does not accurately reflect the true conditions in the schools and therefore cannot generate a useful schedule of high quality.[8]

8 Robert Harding, Problem of Generating Schedules for Schools (Ann Arbor, Michigan, 1969), p. 8.

3.3 THEORETICAL AND CONCEPTUAL DEVELOPMENT

The scheduling system proposed in this thesis will be able to schedule the students at Sodus Central School. The parameters will be left general so that this system could be used to schedule other schools. This project is broken down into three phases which correspond to normal schedule development.

3.3.1 PHASE 1

DESCRIPTION

Guidance counselors meet with each student to determine the student's course requests. The scheduler works with the department chairpersons and the principal to determine the course offerings. Two files are developed: one which contains course offerings and the other the student requests.

In this phase a program must be written which uses the student request file and course offering file as inputs. The output of this phase is a conflict matrix and the course tallies list. The normal procedure is that the conflict matrix and the course tallies list are printed and they are then used by the scheduler to determine the final course offering and to develop the master schedule.

This project will give the scheduler some added flexibilities. Course tallies are used to make decisions which might involve dropping a course from the offering. It would be possible to run the conflict matrix at a later time after the guidance department updates student requests. This would result

in a conflict matrix which is more accurate.

CURRENT PROBLEMS

1) Course tallies and conflict matrix are run at the same time. The course tally sheet is then used to make decisions about dropping courses with insufficient enrollment. Some students have to pick up new courses to replace courses dropped and as a result the conflict matrix is not accurate.

2) The conflict matrix is run once in May and by the second simulation run in July the data is too old to be of use in working out problems.

3) The scheduler must individually tally the number of students conflicting with a given course.

SOLUTIONS

1) The course tallies list can be regenerated after the course decisions have been made and the conflict matrix can then be run to provide more accurate information.

2) A conflict matrix could be regenerated whenever changes have been made to student course requests.

3) A conflict matrix could individually generate and tally all conflicts with a given course.

3.3.2 PHASE 2

DESCRIPTION

In phase 2 the Student Course File and the Course Schedule File become inputs for the sectioning program. The sectioning program loads students into the courses while attempting to maintain a balance in the class size of different sections of the

same course. If students are unable to schedule they are added to the conflict list and a partial schedule is printed. At the end of this simulation run, the school results are printed.

Guidance counselors use the partial schedules to resolve direct conflicts by having the students make a choice between the courses which conflicted. The scheduler uses the school results to examine problems with class balance and make other decisions about changing the times of course offerings. After changes have been made to the Student Request File and the Course Schedule File there will be another simulation run.

This process is continued until the scheduler is satisfied with the results and is ready for a final run. This thesis will give the scheduler some added flexibilities. Under present conditions BOCES allows only two simulation runs before the final run. Since the scheduler will have control of this process he/she can take more runs in attempting to provide a better schedule.

CURRENT PROBLEMS

1) BOCES only permits two trial simulation runs in the general package. A school must pay for extra runs.

2) Turnaround time is about two weeks for each run.

3) As a result of having only two trial runs, the first run is usually held until after June exams in order to include failures.

4) After the first trial run the scheduler might have to move some courses to different periods. He is unable to determine the possible effect the moves will have on each other. It is possible for a change to cause more problems

than were present originally.

5) It is not possible to resolve all balance problems.

SOLUTIONS

1) By having the capability of running the schedule on the school's equipment the scheduler can take as many trial runs as necessary to obtain a satisfactory schedule.

2) Turnaround time will be about one day instead of two weeks.

3) The first trial run could be held before school ended so guidance counselors would have time to personally contact each student with a direct conflict before they leave for summer vacation.

4) The scheduler does not have to solve all problems during a single trial run. A better schedule can result by isolating the results of a single change and having extra trial runs. This is especially true when trying to resolve balance problems.

5) If a trial run does not accomplish its objective the scheduler can back up and start from the last good run.

3.3.3 PHASE 3

DESCRIPTION

In phase 3 the same inputs are used as in phase 2 but now all student schedules and teacher class lists are printed with the school results.

CURRENT PROBLEMS

1) With only two trial runs in phase 2 the scheduler often makes some additional changes before the final run. If these changes cause any conflicts then the conflicts must be

hand scheduled and all class lists updated.

2) The final run is during the first two weeks in August, so any student entering after this must be hand scheduled and class lists updated.

3) Second semester class lists are inaccurate when January arrives.

4) Study halls must be manually divided after the final run.

5) Double gym classes for students who failed must be hand scheduled after the final run which results in many class lists being inaccurate.

SOLUTIONS

1) The final run can be the same as the last trial run so there are no untested results in the final run.

2) This thesis allows for the development of extensions after the final run which would permit additions and deletions while updating the class lists.

3.3.4. SCHEDULING CALENDAR

Scheduling is a year-long project for most schools. A typical calendar for a secondary school is:

November - January

Review the curriculum and make decisions on course offerings, type of schedule, and length of school day for the following year.

February - April

Guidance counselors meet with students to prepare the students' course requests.

May

Course offerings and student requests are used as input to the computer program to produce the conflict matrix and course tallies. Registration figures are adjusted for cancelled and merged courses and staffing needs are formulated.

June (first three weeks)

Construction of master schedule by the scheduler.

June (last week)

Change student requests to reflect results of the June exams.

July

Two trial or simulation runs of the schedule. During these runs student requests are matched with course offerings. The results are reviewed to determine whether the master schedule and/or student course selections must be modified. Students are contacted regarding conflicts.

August (first two weeks)

The final run of the schedule which produces all student schedules and class lists.

August (last two weeks)

Register new students and update class lists. Hand schedule students unassigned time (study halls) and make faculty study hall coverage assignments. Some classes must be hand balanced and lists updated.

September

Guidance counselors continue to enroll new students and add and delete courses from student's schedules.

4. FUNCTIONAL SPECIFICATION

4.1. FUNCTIONS PERFORMED

The student scheduling system requires the user to input student and course information which is stored in system files. This data is used to develop the course tallies list and the conflict matrix. After allowing for updating and deleting of information, the system sections students and prints their schedules. The functions performed include:

1. Develop student information files.
    A. Student course request file.
    B. Student general information file.
2. Update student information file.
3. Develop course file.
4. Update course file.
5. Develop course tallies list.

    The course tallies list summarizes the student request information. The list will include the course number, course name, total course enrollment and the course enrollment by grade level.

6. Develop conflict matrix.

    The conflict matrix gives the total number of students registered for any given pair of courses.

7. Section students.

A. Develop student schedule.

1. Get a student request record.

2. Get course records for the student's requests.

3. Prioritize the course requests.

Courses are first prioritized according to the number of sections of a given course that are being offered. Each section is then prioritized according to the number of seats remaining.

4. Schedule the student's courses.

B. Print student schedule for any student with a conflict. If a student is unable to schedule all of his courses his student ID is added to the conflict list file.

C. Print school results.

After each run a list of course numbers and their enrollments is printed along with the scheduling results by grade level. The number of students assigned to study hall is also printed.

D. Store student schedule.

The student's schedule is stored in the schedule file (during the final run).

8. Print results.

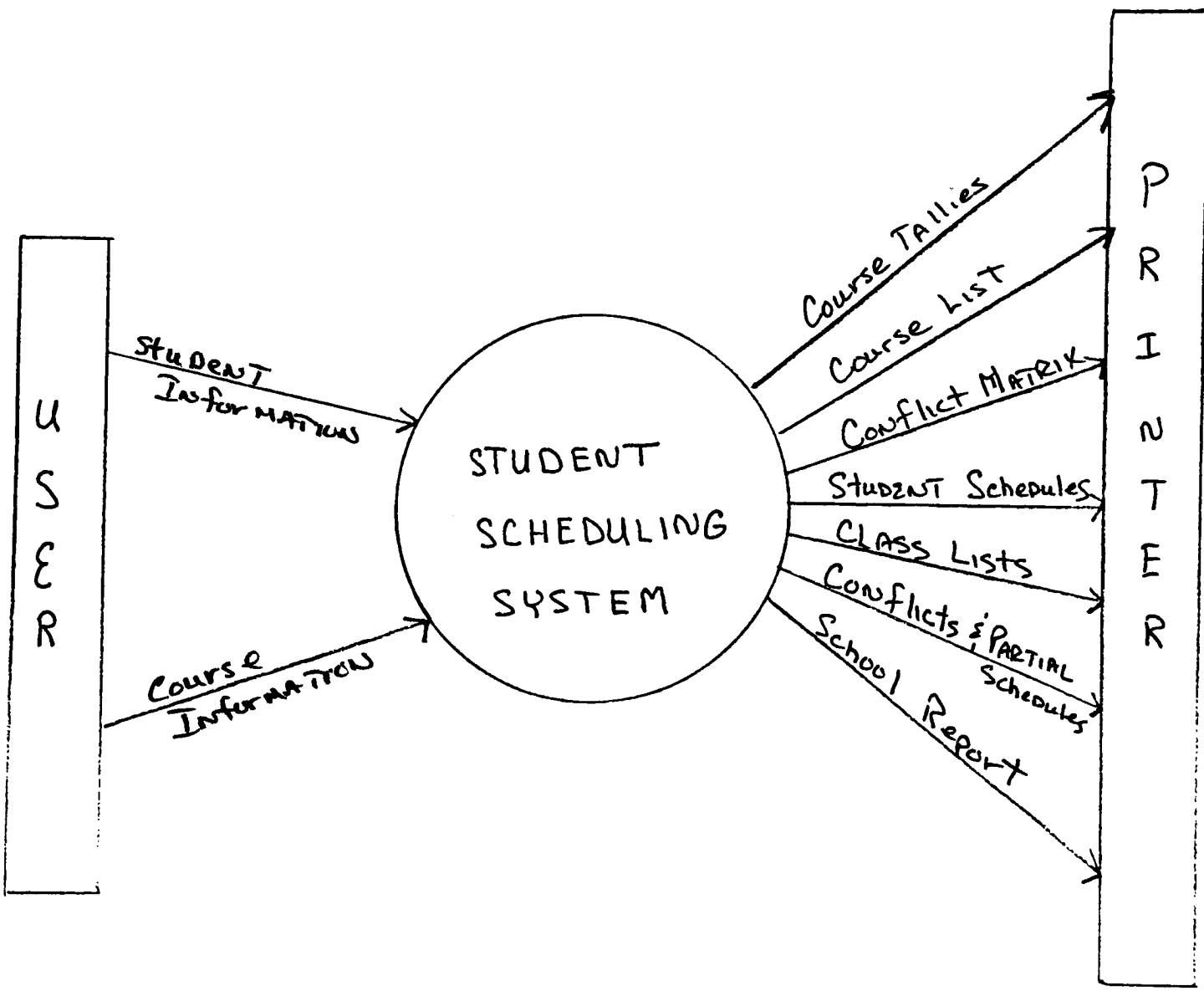
A. Print schedules of all students after the final run.

B. Print class lists from student schedule records after the final run.

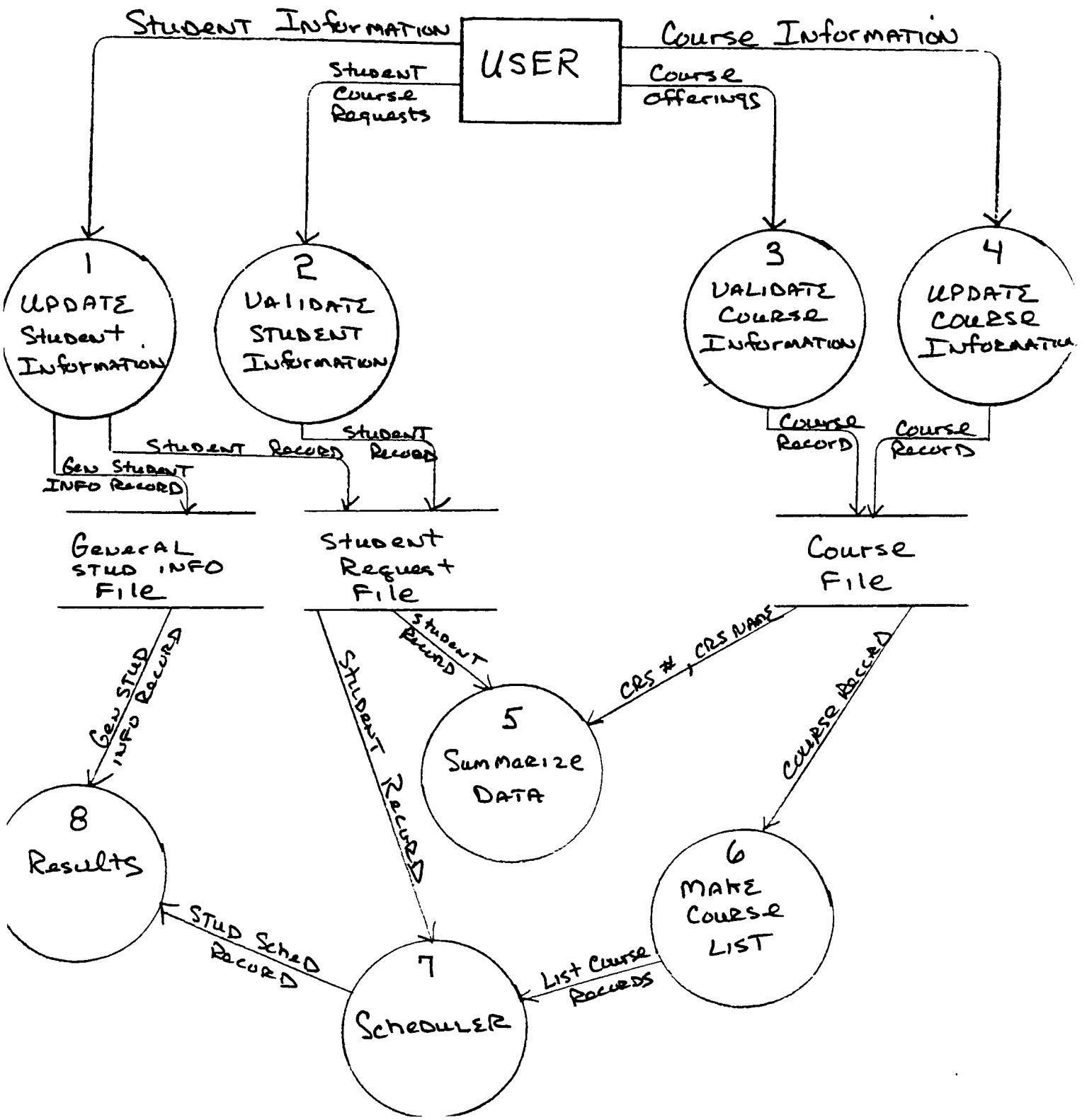
4.2. SYSTEM DATA FLOW CHART

CONTEXT DIAGRAM

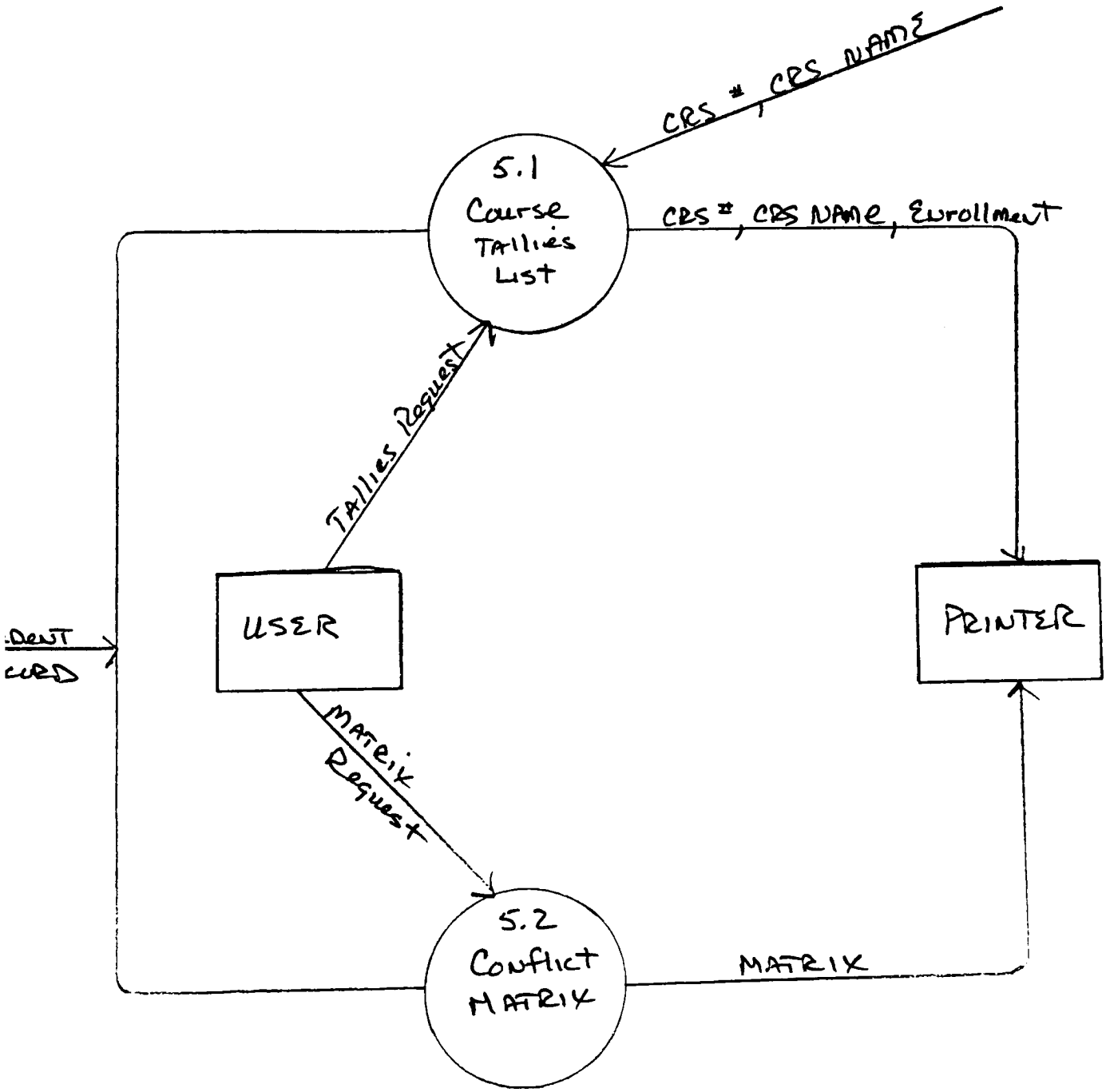
LEVEL 0



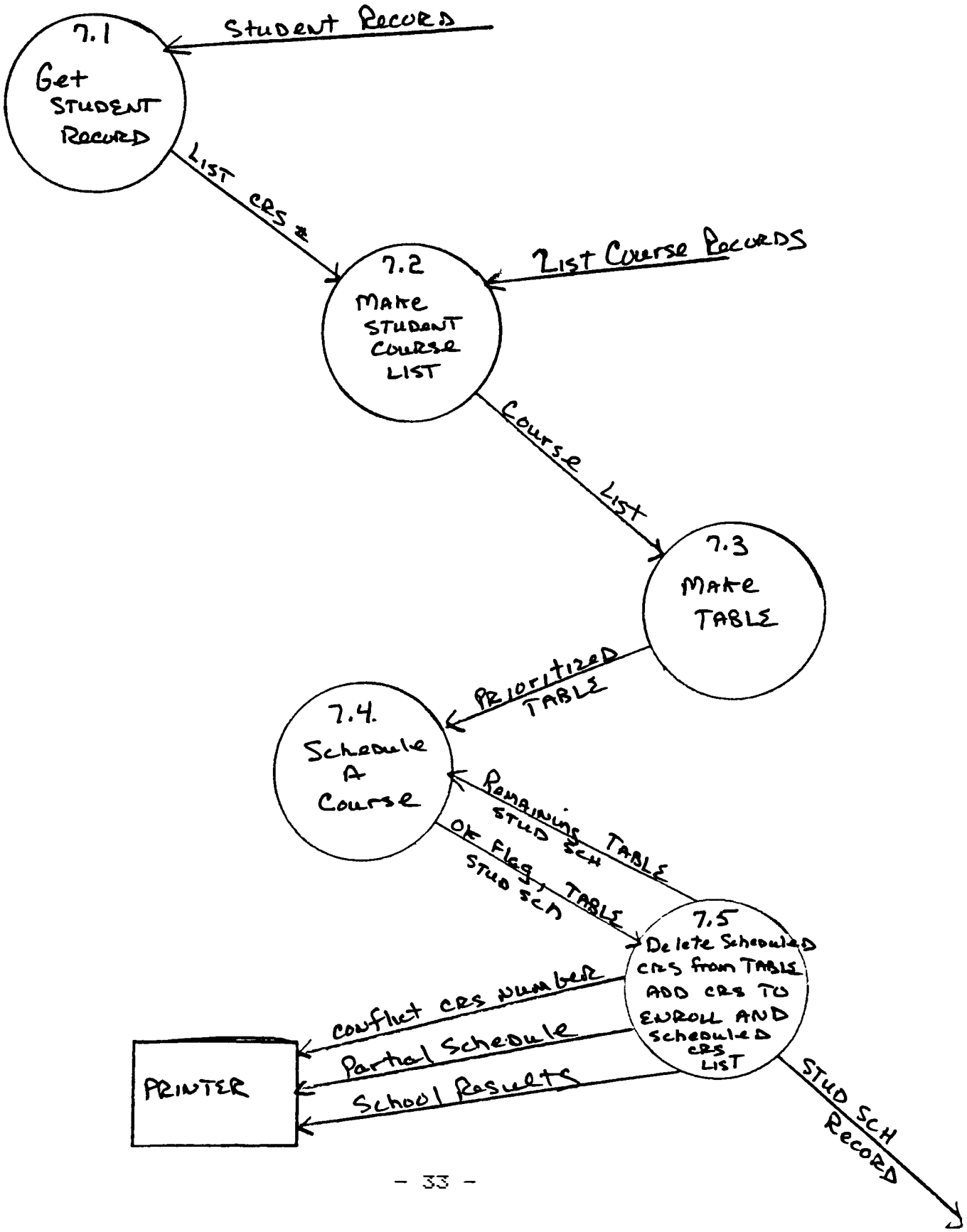
LEVEL 1


STUDENT INFORMATION
USER
COURSE INFORMATION
STUDENT COURSE REQUESTS
COURSE OFFERINGS
1 UPDATE STUDENT INFORMATION
2 VALIDATE STUDENT INFORMATION
3 VALIDATE COURSE INFORMATION
4 UPDATE COURSE INFORMATION
GEN STUDENT INFO RECORD
STUDENT RECORD
STUDENT RECORD
COURSE RECORD
COURSE RECORD
GENERAL STUD INFO FILE
STUDENT REQUEST FILE
COURSE FILE
GEN STUD INFO RECORD
STUDENT RECORD
STUDENT RECORD
CRS #, CRS NAME
COURSE RECORD
5 SUMMARIZE DATA
8 RESULTS
6 MAKE COURSE LIST
STUD SCHED RECORD
7 SCHEDULER
LIST COURSE RECORDS

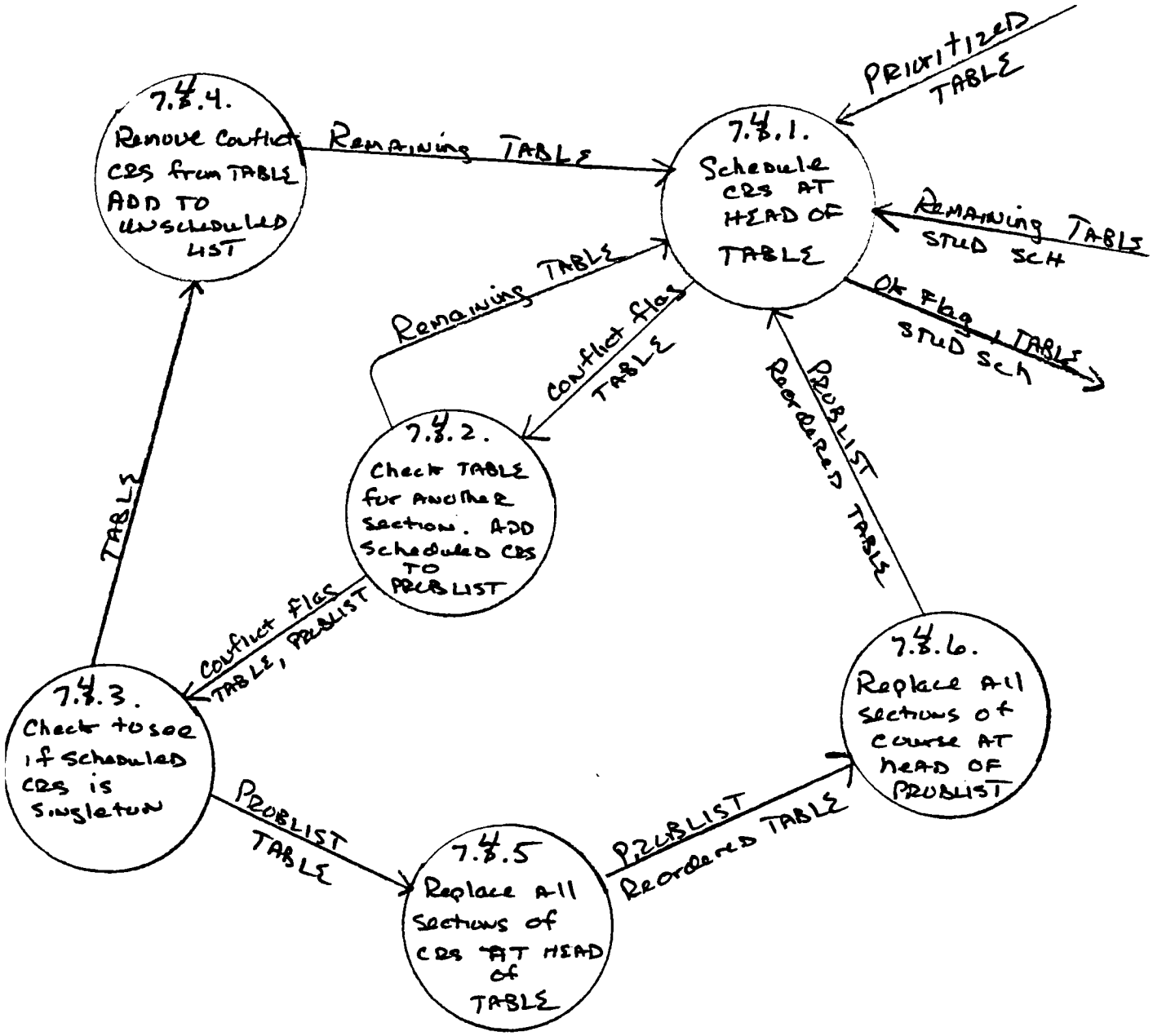
EXPLOSION 5


CRS #, CRS NAME
5.1
Course
Tallies
List
CRS #, CRS NAME, Enrollment
Tallies Request
STUDENT
CARD
USER
PRINTER
MATRIX
Request
5.2
Conflict
MATRIX
MATRIX

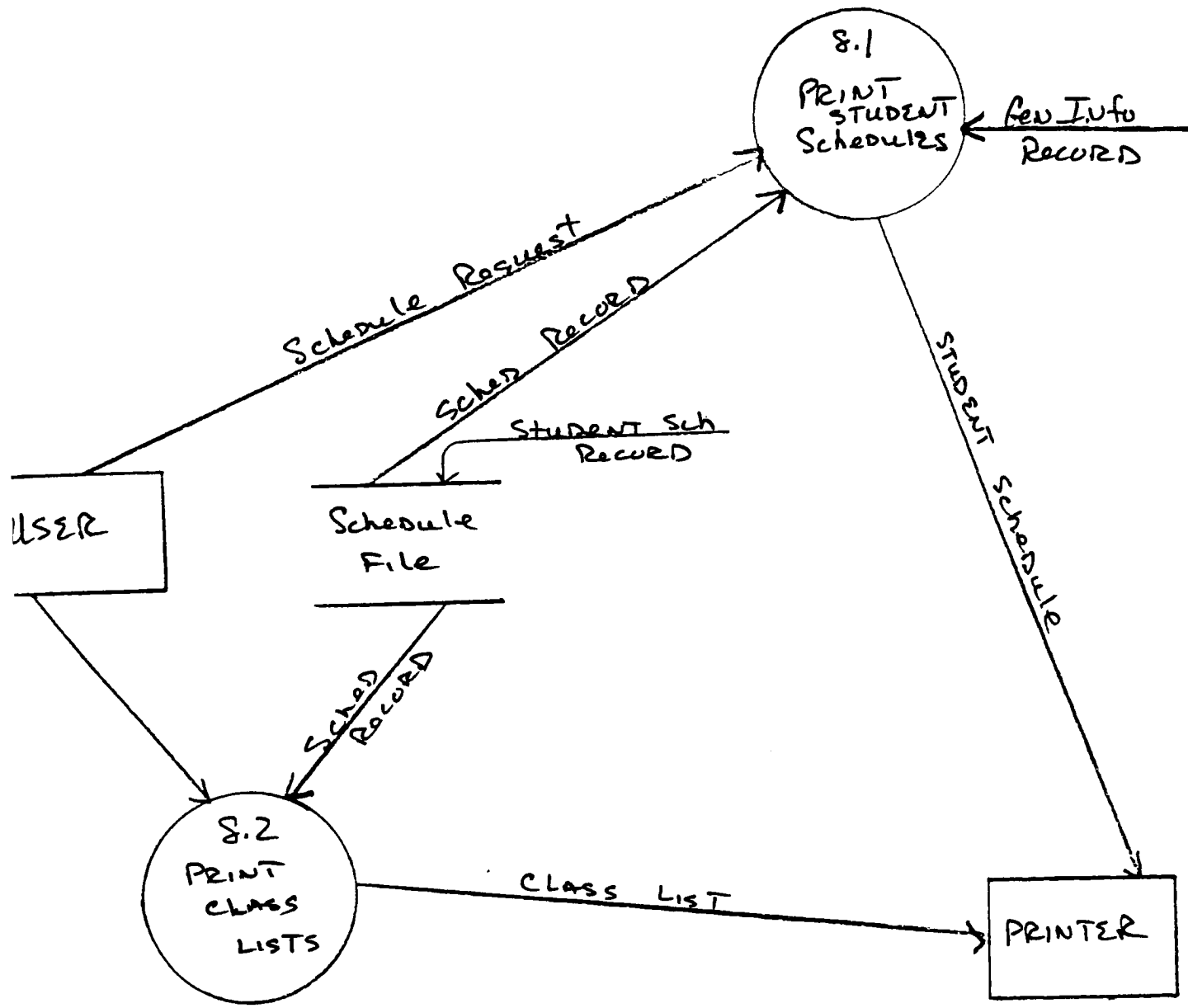
EXPLOSION 7


7.1
Get STUDENT RECORD
STUDENT RECORD
LIST CRS #
7.2
MAKE STUDENT COURSE LIST
LIST COURSE RECORDS
COURSE LIST
7.3
MAKE TABLE
PRIORITIZED TABLE
7.4
Schedule A Course
REMAINING STUD SCH TABLE
OK FLAG, STUD SCH TABLE
7.5
Delete Scheduled CRS from TABLE
ADD CRS TO ENROLL AND SCHEDULED CRS LIST
conflict CRS NUMBER
Partial Schedule
School Results
PRINTER
STUD SCH RECORD

EXPLOSION 7.4



EXPLOSION 8


8.1
PRINT STUDENT SCHEDULES
GEN INFO RECORD
SCHEDULE REQUEST
SCHED RECORD
STUDENT SCH RECORD
STUDENT SCHEDULE
USER
SCHEDULE FILE
SCHED RECORD
8.2
PRINT CLASS LISTS
CLASS LIST
PRINTER

4.3. LIMITATIONS AND RESTRICTIONS

The system will not generate the master schedule. The master schedule information must be an input.

4.4. USER INPUTS

Student and course records developed from user inputs are stored in system files. The student information is entered once but stored in two separate records, the student record and the general student information record. The system scheduler develops schedule and class list records which are also stored in system files. The inputs to the system include:

1. Student information
   A. Student record (entered by user)
      - Student name
      - Student ID
      - Grade
      - Course requests
   B. General student information record (entered by user)
      - Student name
      - Student ID
      - Address
      - Phone number
      - Homeroom number
      - Birthdate
2. Course record (entered by user)
   - Course number
   - Course name
   - Section number
   - Maxsize
   - Period number
   - PD number (scheduling period)
   - Days
   - Semester
   - Room number
   - Teacher number

3. Schedule record (generated by scheduler (DFD EXPLOSION 7))
   - Student ID
   - Student name
   - Grade
   - Scheduled courses
4. Course requests Array [1 to 9] (from user)
   - Course numbers

4.5. USER OUTPUTS

1. Course tallies list - This list is printed using information from the student request file and the course file (DFD EXPLOSION 5) . The list contains:

   - Course number

   - Course name

   - Total enrollment

   - Enrollment by grade level

2. Conflict Matrix - This matrix is a printed listing of the enrollment for courses which could conflict with a requested course (DFD EXPLOSION 5). The matrix contains:

   - Course for which matrix is requested.

   - List of all courses which will conflict with that course.

   - Enrollment for each given pair of course numbers.

3. Student Schedule - The schedule is stored in the schedule file for printing at a later time. It is developed using information from the course file and the student request file (DFD EXPLOSION 7). The stored schedule contains:

   - Student name

   - Student ID

   - Student grade

   - Schedule course records

When printing the schedule (DFD EXPLOSION 8) the following information is added from the general student information file:

- Address
- Phone number
- Homeroom number
- Birthdate

4. School results report - This report is generated after each run of the schedule using information stored in the course file (DFD EXPLOSION 7). The report contains:

- Course number
- Course name
- Section number
- Maxsize
- Period number
- PD number
- Days
- Semester
- Room number
- Teacher number
- Enrollment

5. Partial Schedules - The schedules of students who were unable to schedule all of their courses are printed along with the course numbers of their unscheduled courses (DFD EXPLOSION 7).

6. Class lists - These lists are only printed after the final run. They are printed from information stored in the schedule file (DFD EXPLOSION 8). The lists contain:

- Course name
- Course number
- Section
- Period number
- Days
- Teacher number
- Student list

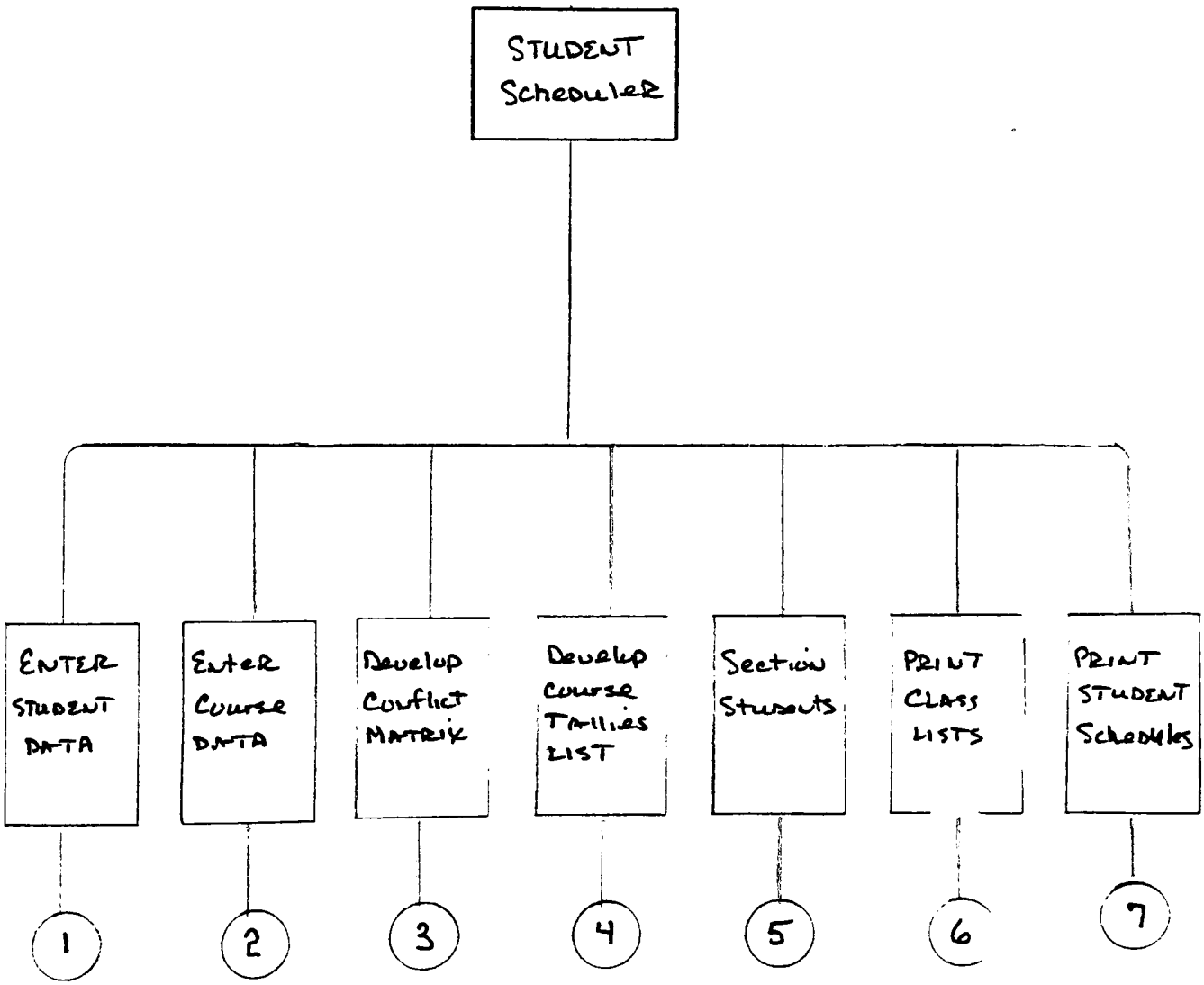
4.6. SYSTEM FILES

The student request and general information files, and the course file are inputed by the user. The system generates the schedule file during a run of the schedule. The temporary class record file is a temporary file that is generated in order to print class lists. The following files are developed and maintained by the student scheduling system:

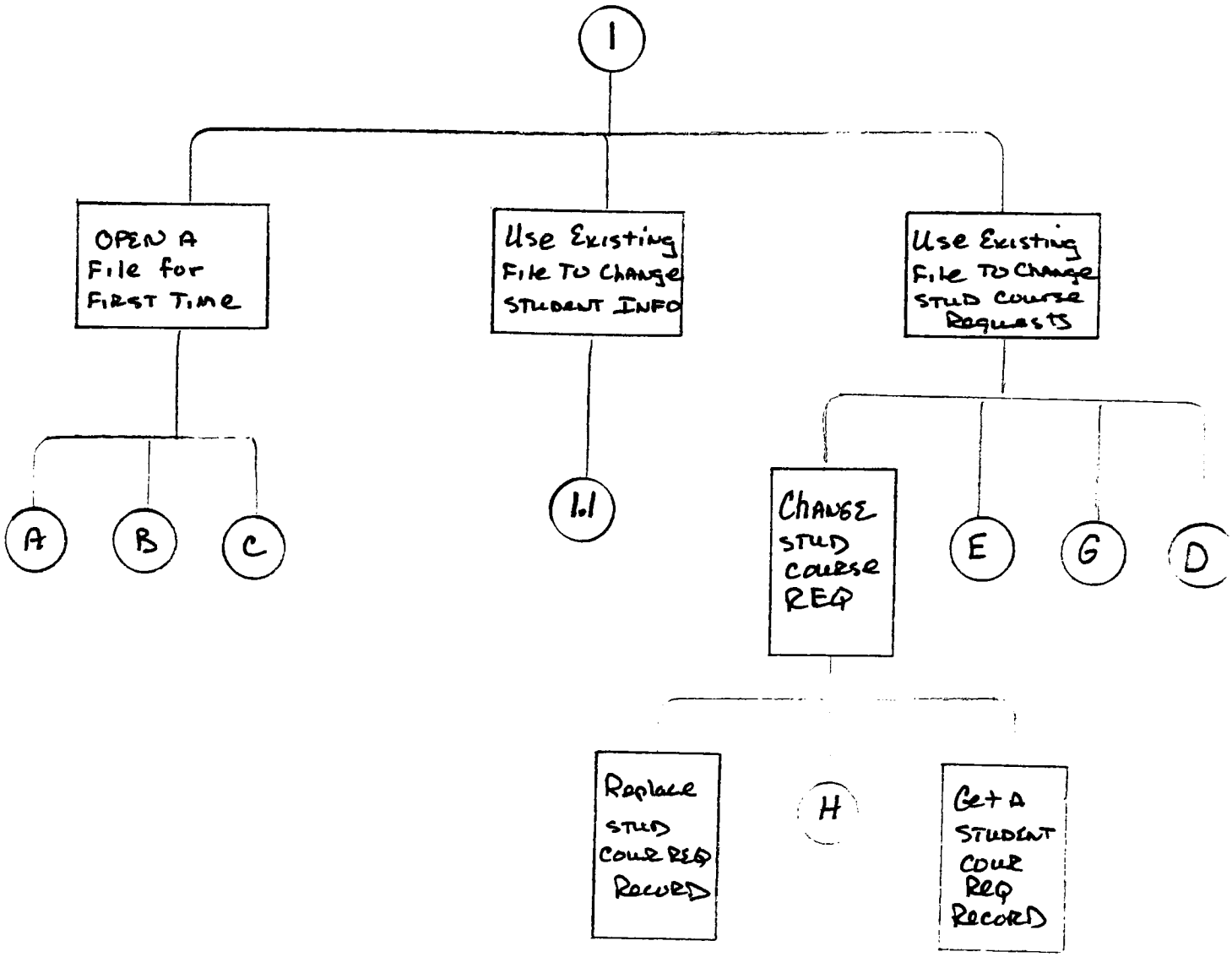
1. Student request file
   - Student request records
2. Student information file
   - General student information records
3. Course file
   - Course records
4. Schedule file
   - Schedule records
5. Temporary class record file
   - Schedule records

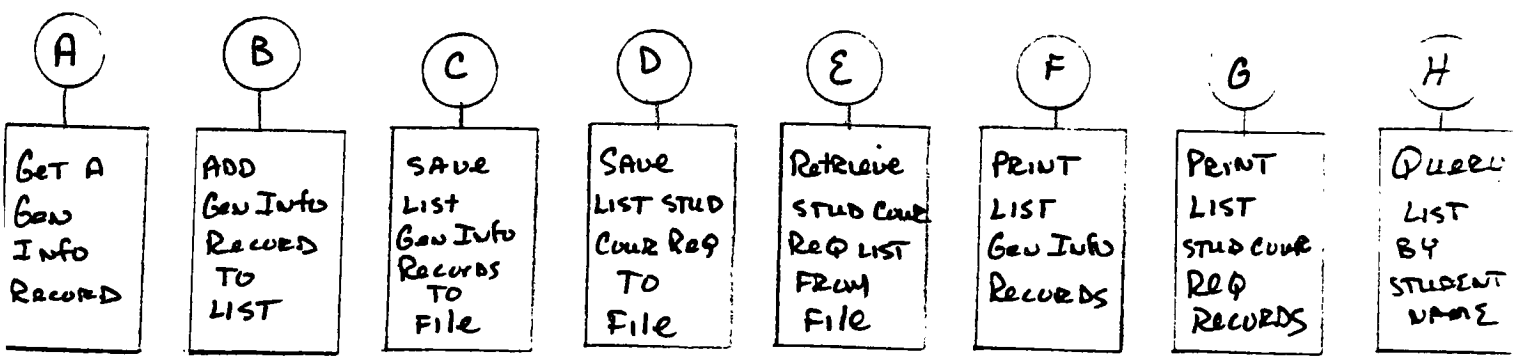
5. SYSTEM SPECIFICATION

5.1. SYSTEM ORGANIZATION CHART

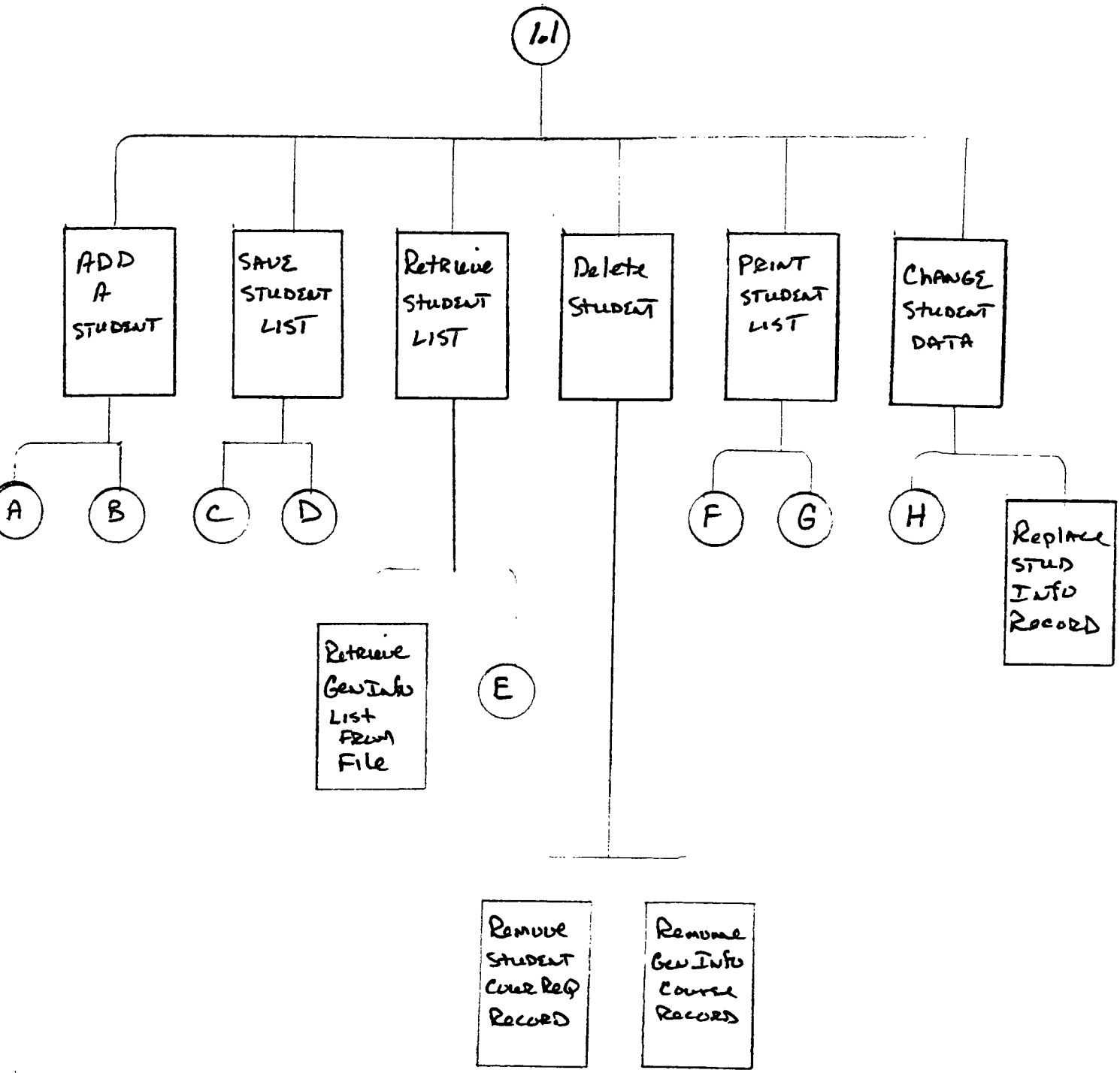


EXPLOSION 1

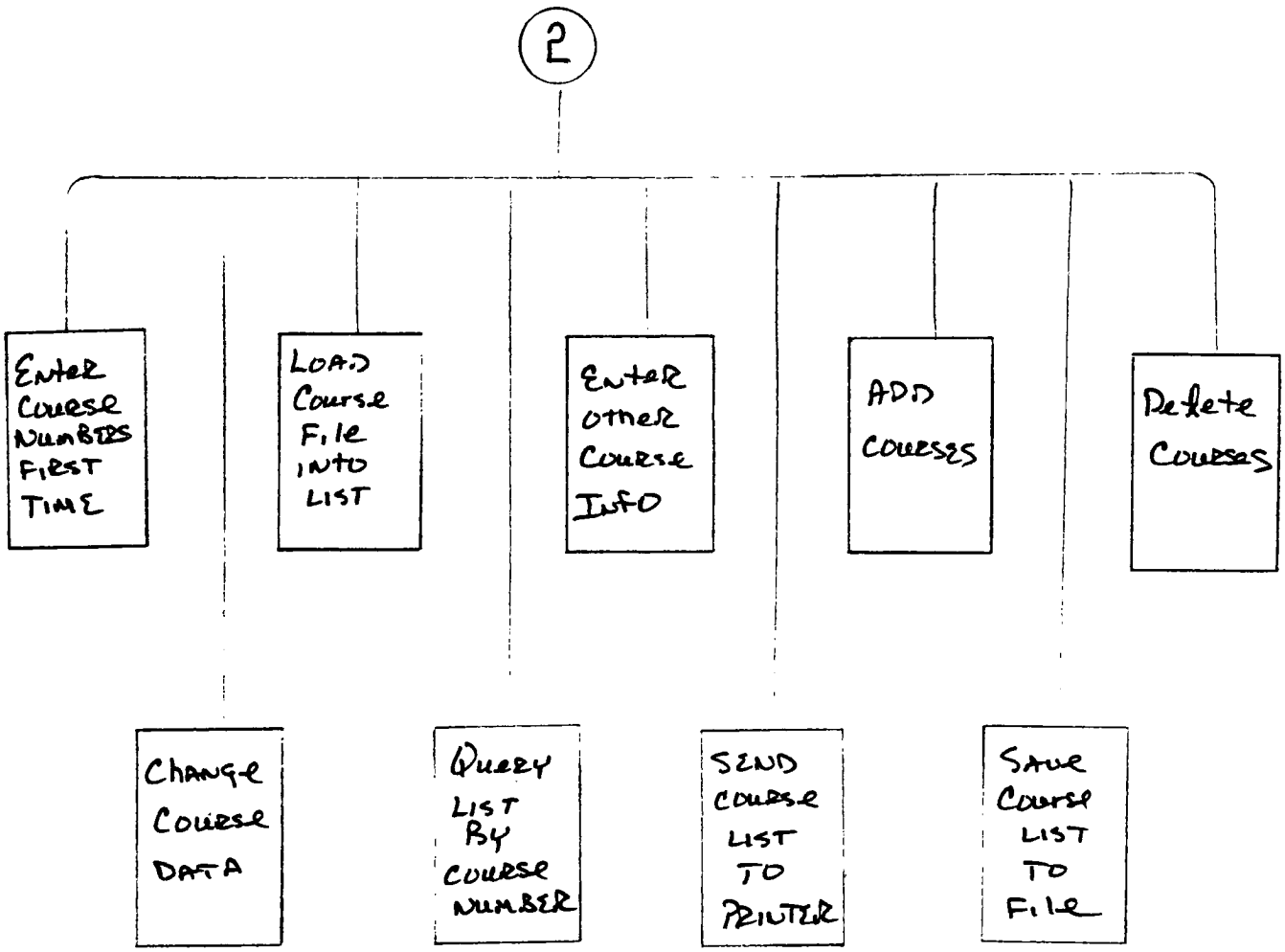

1
OPEN A File for FIRST TIME
Use Existing File TO Change STUDENT INFO
Use Existing File TO Change STUD Course Requests
A
B
C
1.1
CHANGE STUD COURSE REQ
E
G
D
Replace STUD COUR REQ RECORD
H
Get A STUDENT COUR REQ RECORD


A
GET A GEN INFO RECORD
B
ADD GEN INFO RECORD TO LIST
C
SAVE LIST GEN INFO RECORDS TO File
D
SAVE LIST STUD COUR REQ TO File
E
RETRIEVE STUD COUR REQ LIST FROM File
F
PRINT LIST GEN INFO RECORDS
G
PRINT LIST STUD COUR REQ RECORDS
H
QUERY LIST BY STUDENT NAME

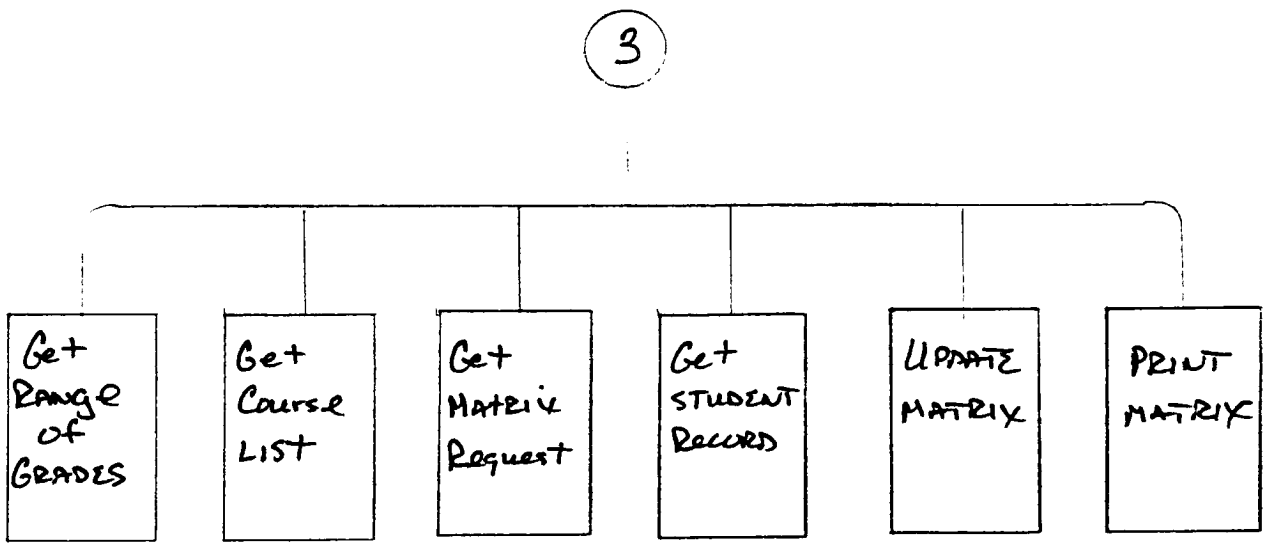
EXPLOSION 1.1



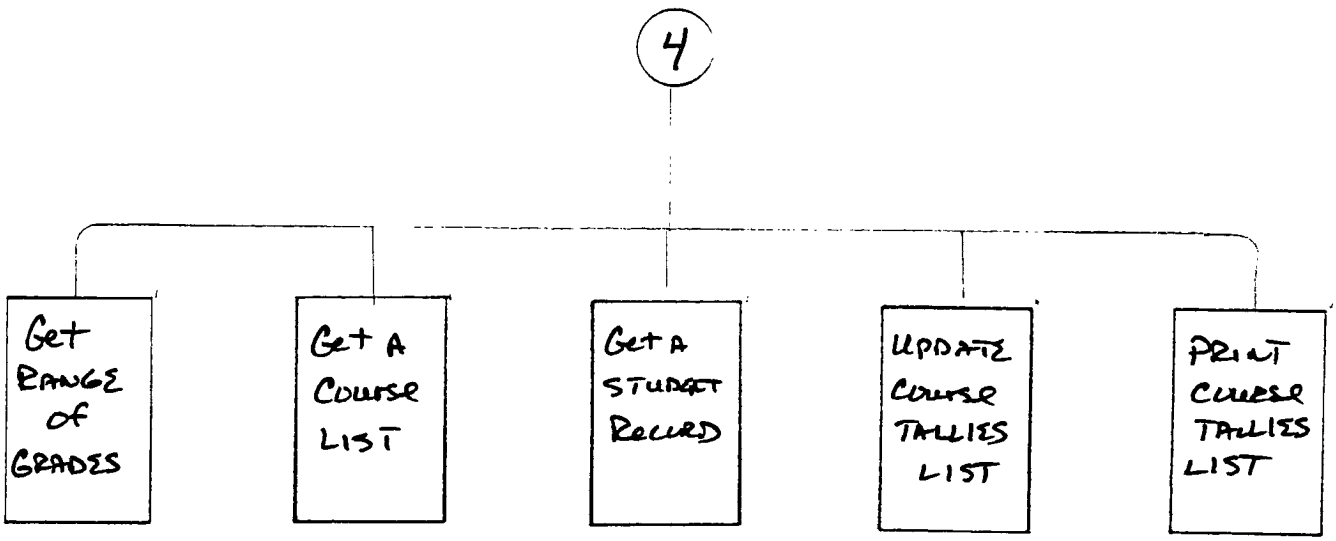
EXPLOSION 2


2
Enter Course Numbers First Time
Load Course File into List
Enter other Course Info
Add Courses
Delete Courses
Change Course Data
Query List By Course Number
Send Course List to Printer
Save Course List to File

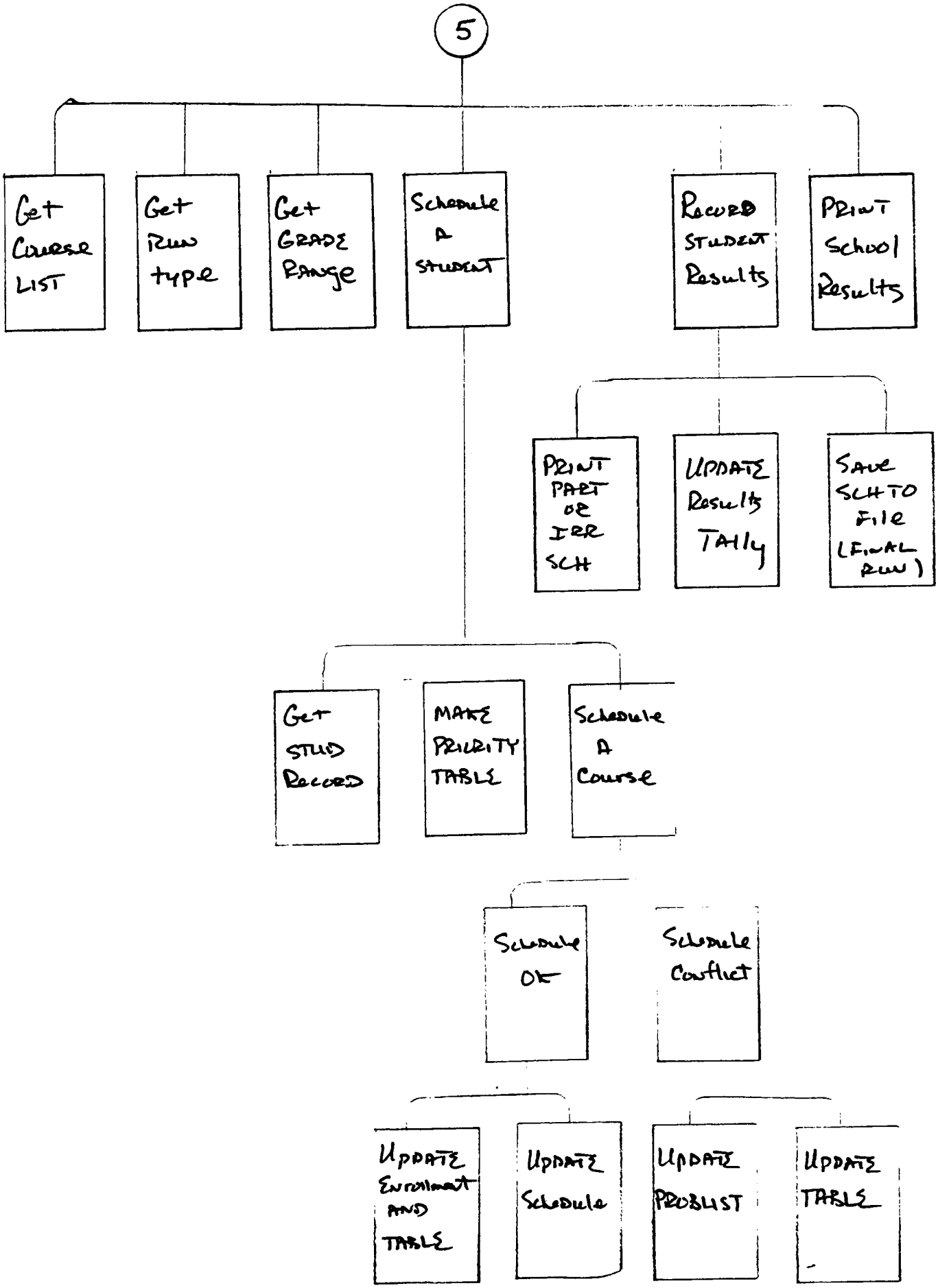
EXPLOSION 3


3
Get Range of Grades
Get Course List
Get Matrix Request
Get Student Record
Update Matrix
Print Matrix

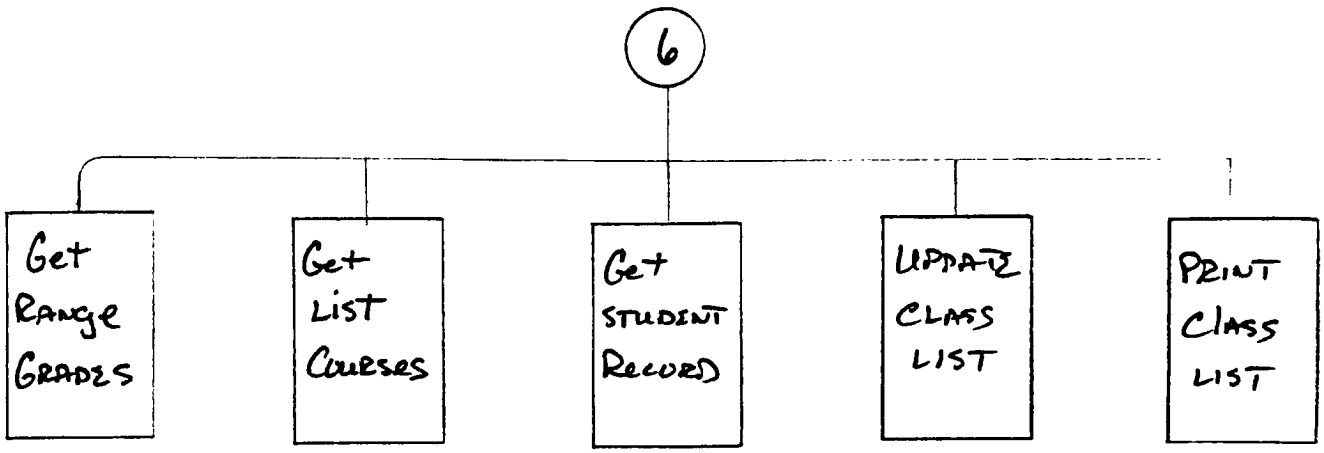
EXPLOSION 4

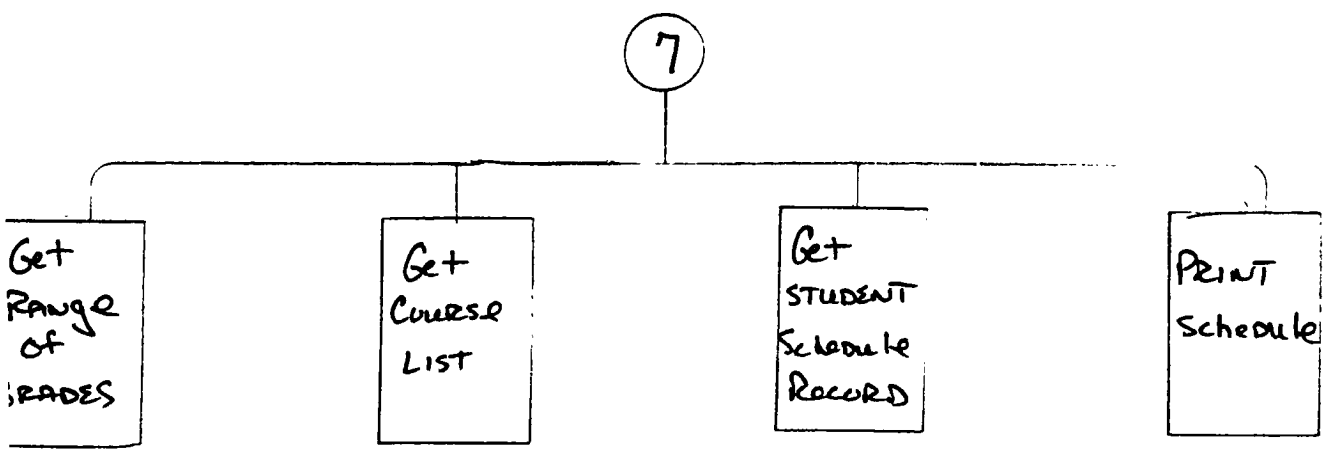

4
Get RANGE of GRADES
Get A COURSE LIST
Get A STUDENT RECORD
UPDATE COURSE TALLIES LIST
PRINT COURSE TALLIES LIST

EXPLOSION 5


5
Get Course List
Get Rpt type
Get GRADE Range
Schedule A Student
Record Student Results
Print School Results
PRINT PART OF IRR SCH
UPDATE Results TAlly
SAVE SCH TO File (FINAL RUN)
Get STUD Record
MAKE PRIORITY TABLE
Schedule A Course
Schedule OK
Schedule Conflict
UPDATE Enrollment AND TABLE
UPDATE Schedule
UPDATE PROBLIST
UPDATE TABLE

EXPLOSION 6 AND 7





5.2. EQUIPMENT CONFIGURATION

This scheduling system requires an Apple IIE with four disk drives and 64K main memory with an extended eighty column card.

5.3. IMPLEMENTATION TOOLS

This student scheduling system is written in Apple Pascal 1.2.

5.4. OVERVIEW OF THE PROGRAMMING SYSTEM

The student scheduling system presented in this thesis consists of seven programs written in Apple Pascal. The following sections present an overview of each program.

5.4.1. Enstuddata (see program listing 10.1.)

This program allows the user to create and update all student information. Entstuddata uses Unit One (see program listing 10.2.) which contains subroutines used by options one and two. The user is allowed to select one of four options:

Option 1

This option is used the first time the user wants to enter student information for each grade. The user enters general information for a grade and the information is stored in a student request record (used for student course requests) and a general information record (general student information). At this time the course requests are unknown so they are not entered. The

student request records and general information records are placed in separate linked lists in alphabetical order. The user is then given the opportunity to save the two lists to files on a disk. One file will contain student request records (ST:STREC(grade)) and the other will contain general information records (ST:GINFO(grade)) for the same grade.

Option 2

This option allows the user to select from nine options in order to use an existing file to enter and change student information. The grade is used as the key to locate the correct files. The records are then placed into separate linked lists. As information is changed for any student, both the student request record and the general information record are updated. It is also possible to add or delete students. Any student who is added is placed in both lists in alphabetical order. Options are provided to print either the student course request list (see appendix 9.1.2.) or the general information list (see Appendix 9.1.1). After updating student information both lists can be saved to files on disk.

Option 3

The user can use option 3 to select from six options in order to use an existing student request file to enter and change course records. This option is used after the student request records have been created by options one or two. The records for a selected grade are read from the student request file for that grade and placed in a linked list in alphabetical order. The user then is allowed to add up to 10 course requests (course numbers) per student. The course requests are sorted and stored in an

array in the course request record. The user may query the course request list by student name or send the list to the printer.

Option 4

Exit the program.

5.4.2 Entcourdata (see program listing 10.5.)

This program allows the user to select from ten options to enter and update all course information. The first option allows the user to enter course information (course name and number) for the first time. The course number and name are read into a course record which is then inserted into a linked list ordered by course number. This option allows the user to develop the list of courses to be offered before other specific course information is known. After the Coursefile (CR:COURSEFILE) has been developed, it may be retrieved to add, delete and and change courses or course information. The course file contains course records which are placed in a linked list in order of course number. The user may query the list by course number or send the list to the printer (see Appendix 9.4.). The updated course list may be saved to disk.

5.4.3. Cmatrix (see program listing 10.3.)

This program produces the conflict matrix (see Appendix 9.2.). The user enters the range of grades that the matrix will cover. The matrix is a list of all courses which would conflict with the course requested by the user. The list would also

display the number of students who would have this conflict.

The program uses two parallel arrays to tally the number of students in conflict with the course entered by the user. One array which contains course names is initialized to all blanks while the other array which contains tallies (integers) is initialized all to zero. Course records are read from the coursefile (CR:COURSEFILE). The course name is placed into the course name array using the course number as the index.

The user enters the number of the course for which the matrix is to be constructed. The student request records, with file names which match the range of grade numbers entered by the user, are read. If the student's course request array contains the same course number as requested for the matrix then the course number is used as an index to add one to the tally in the tally array for each course that the student is requesting. When this is completed for all students in the requested grade range the matrix for the requested course is complete.

The conflict matrix is then printed by indexing through both arrays and printing only when the course name array is not blank and the tally array is greater than zero.

5.4.4 Tallylist (see program listing 10.4)

Tallylist prints the course tallies list (see Appendix 9.3.). This list shows the enrollment for each course in the curriculum by grade level.

A two-dimensional array is initialized to zero. This array is used as an array of counters which keeps track of course

tallies. One dimension is indexed by the course number and the other by grade. The user enters the range of grades. The grade is then used to get the correct student course request files (ST:STREC(grade)). The student's record is read from the file and every course the student requested is used as an index along with the student's grade, so one can be added to the tally. The sum of the rows provides the total enrollment for a course.

A course name array which is used to print the course names is filled by reading course records from the coursefile (CR:COURSEFILE). The course tallies list includes the course name, number, total enrollment and enrollment by grade level for each course offered.

5.4.5. Scheduler (see program listing 10.6.)

This program schedules students and prints the results of each scheduling run. Scheduler uses Unit Two (see program listing 10.7.) which contains sub-routines used by the Scheduler. Unit Two loads all course records from the coursefile into course enrollment records and sets the class enrollment to zero. The class enrollment records are loaded into a linked list in order of course, section and PD (secheduling period) number.

The user must also enter the type of scheduling run (trial or final) and the range of grades to be scheduled. During a final scheduling run the student schedule records are saved to a file (SCH:SCH(grade)).

A student record is read from the student request file and the course records of all sections of the course which a student

requested are found in the linked list of course enrollment records. These records are inserted into a linked list of course records in order of course, section and PD number for the student being scheduled. This linked list is then used to create a table which is a two-dimensional array of course number, course section, class balance (enrollment subtracted from max seats), and priority. The priority is the number of sections of any course. The table is then sorted first by priority then by the class balance.

The table which contains a prioritized list of all sections of a students requested courses is then used to schedule the student. The first course in the table is placed in the schedule (a two-dimensional array)indexed by PD numbers (1..13) and days of the week (M,T,W,R,F). The schedule array contains course numbers which are initialized to zero. To schedule a course the schedule array is checked for zero and if a zero is found then the course is scheduled in place of the zero. If a conflict exists and if another section of the same course exists in the table then the attempt is made to schedule that section. The course conflicting with the original section is then added to the problem list (an array of course numbers which resulted in conflicts). If another section of the same course does not exist then the conflict course is removed from the schedule and all sections of this conflict course are added to the head of the table. The original course that the program was trying to schedule is scheduled in place of the conflict course. The conflict course which is now at the head of the table is now scheduled and if this results in another conflict then the course

is removed from the table and declared a conflict. If only one conflict exists in a schedule then the student is considered partially scheduled but if a second conflict occurs the student is considered unscheduled.

Courses are taken from the head of the table and placed in the schedule until the table is empty. If a course is scheduled then one number is added to the enrollment for that section of the course.

When the student records of an entire grade have been scheduled then the scheduling results of the grade are printed and if it is the final run the schedule records are saved to a file. When all requested grades have been scheduled the final school results (see Appendix 9.5.) are printed.

5.4.6. Printschedules (see program listing 10.8.)

This program prints out the student schedules (see Appendix 9.6.) after the final run of the schedule has been completed. The course records are read from the course file and inserted into a linked list in order of course number, course section and PD number.

The user enters the range of grades for which schedules are to be printed. The student schedule records are read from the schedule file (SCH:SCH(grade)) and information is used from the course records list to print the students' schedules.

5.4.7. Printclasslists (see program listing 10.9.)

This program prints course class lists (see Appendix 9.6) using records from the schedule file. The course records are read from the course file and inserted into a linked list in order of course number, course section, and PD number.

The user enters the range of grades for which the class lists are to be generated. All courses for which a student is scheduled are copied from student schedule records in the schedule files. These records are transferred into temporary class records containing the student name, course, section, student ID and grade and are then stored in a temporary class record file.

A course class list is then developed for each course and section listed in the linked list of courses. The temporary class record file is read and every record which contains the given course and section is inserted into a linked list of temporary class records in alphabetical order. This list is then printed as the class list.

6. VERIFICATION AND VALIDATION

6.1. TEST PLAN

In this thesis a programming system to schedule students using a microcomputer was developed. The scheduling at Sodus High School is currently done using a mainframe computer at BOCES. To test this thesis programming system a schedule was developed using the same data that was used for scheduling Sodus High School with the BOCES computer.

6.2 TEST PROCEDURES

The schedule for the 1984-85 school year at Sodus High School was developed during the summer of 1984. This schedule was run on the BOCES computer for grades nine through twelve. In testing the student scheduling program written for this thesis the same data was used.

The test of this program was done in the following sequence:

1. Development of a list of student records (see Appendix 9.1).
2. Development of a conflict matrix (see Appendix 9.2).
3. Development of a course tallies list (see Appendix 9.3).
4. Development of a course list (see Appendix 9.4).
5. Two trial runs of the schedule (see Appendix 9.5).
6. A final run of the schedule (see Appendix 9.5).
7. Printing student schedules (see Appendix 9.6).
8. Printing course lists (see Appendix 9.7).

6.3 TEST RESULTS

The results of the trial and final runs of this project where very similar to the results of the BOCES run of the schedule. The main advantage of having a microcomputer scheduling program was to provide more local control over the scheduling process and this project does provide that control. Some advantages that local control provided were:

1. Turnaround time was reduced from about two weeks to one day. The thesis program takes approximately 3.5 hours to run.

2. It is possible to do more than two trial runs or to go back to a previous run if a run is unsatisfactory.

3. A conflict matrix can be generated before any run, not just the first run. This is very helpful in making changes in later runs of the schedule. Also, conflict matrices are much more useful because they are printed for each individual course.

4. The course tallies list will be more useful because it can be generated anytime. Using the BOCES system the course tallies list is generated after all students have signed up for their courses. Before the list is received from BOCES the tallies have already been counted by hand. In order to make decisions regarding staff assignment for the next year lists are made by secretaries of all students enrolled in a given course. This step could be eliminated by an extension to this project. The result would be a savings in time spent by secretaries in compiling and counting student course requests.

5. Since more than two trial runs are possible the first run of the schedule can occur before school is over in June. This will permit guidance counselors to contact students regarding schedule changes before they leave for the summer.

6. If a run was completed before the end of the school year it would be possible for the scheduler to consult with the department chairpersons and teachers regarding possible changes in their schedules before they leave for the summer vacation. This would allow for more staff input to the scheduling process.

7. By developing some of the extensions suggested in section 7.3.2., this student scheduling system would become even more useful to a school.

COMPARISON OF RESULTS BY RUN

	BOCES	THESIS PROJECT
RUN 1 TOTAL STUDENTS	499	496
STUDENTS FULLY SCHEDULED	352	356
STUDENTS PARTIALLY SCHEDULED	131	123
UNSCHEDULED STUDENTS	16	17
RUN 2 TOTAL STUDENTS	498	496
STUDENTS FULLY SCHEDULED	462	464
STUDENTS PARTIALLY SCHEDULED	35	30
UNSCHEDULED STUDENTS	1	2

FINAL RUN TOTAL STUDENTS	498	496
STUDENTS FULLY SCHEDULED	489	486
STUDENTS PARTIALLY SCHEDULED	9	10
UNSCHEDULED STUDENTS	0	0

7. CONCLUSIONS

7.1. PROBLEMS ENCOUNTERED AND SOLVED

In the development of this program, Apple Pascal was used with an Apple IIE microcomputer. Apple Pascal 1.1 has several bugs, one of which was that IORESULT did not always return a correct value. IORESULT is supposed to return a value from one to sixteen if an error occurs in an input/output operation and a zero if no error occurs. One of the bugs in Apple Pascal 1.1 was that a fourteen should be returned for an error in reading integer or real data. The IORESULT function on Apple Pascal 1.1 did not function correctly and a zero was returned.

Apple Pascal 1.2 corrected 43 bugs in Apple Pascal 1.1 but it did not correct the IORESULT bug. To get around the problem, it was discovered that when an integer was read and an alpha character was entered it left IORESULT as zero but it also left the integer variable a zero. A loop was then used to read until the variable was greater than zero.

While Apple Pascal 1.2 did not solve the IORESULT problem, it did provide some solutions to other problems that were encountered. This newer version of Apple Pascal required an extended 80-column card which provided another 64K of memory. In Apple Pascal 1.1 it was only possible to edit files which where less that 38 blocks (19K bytes). If the program exceeded 38

blocks then some subroutines could be compiled as units and stored in the system library. Apple Pascal 1.2 allowed the editing of files up to 58 blocks (29K bytes). This allowed for greater flexibility in program development.

One of the major problems which was encountered in writing this program for an Apple, was in careful use of main memory. It was very important to develop algorithms which did not result in stack overflow problems. One example of this occurred in the development of the conflict matrix. The conflict matrix printed by the BOCES program was a large matrix (100 by 100) of course tallies . When printed this matrix was eleven sheets of paper which where then taped together to form the giant conflict matrix. In trying to develop this matrix the Apple ran out of main memory so ways had to be found to break the problem down so the Apple could handle it. This matrix is used to find how many students are enrolled in any given pair of courses. In scheduling a course, a list is prepared of the courses and number of students that would conflict with the given course. The solution was to compare one course against all the others and print all the courses that would conflict with the given course and how many students would have the conflict. This turned out to be a better solution to the problem than the BOCES matrix.

Another area where insufficient main memory caused changes in algorithm development was in the Scheduler program. This program grew very large and used many arrays so the use of main memory became an important part of algorithm design. The initial scheduling algorithm used in the Scheduler did not find all possible scheduling combinations, which resulted in

scheduling results about 12% lower than the BOCES results. This problem was solved by using a problem list, an array which keep track of other scheduling possibilities. This enabled the Scheduler to find the other scheduling combinations if all courses where unable to be scheduled.

The original design planned to add each student's course selections to the appropriate class list after scheduling the student. The design was changed to write student schedule records to a file only after the final run and then use this file to print both class lists and student schedules. Besides saving some run time the improved design provides for some future extensions which would make the system even more useful.

7.2.DISCREPANCIES AND SHORTCOMINGS OF THE SYSTEM

This microcomputer version of the scheduling program takes considerable time to run. It requires approximately 3.5 hours for a trial run and about four hours to print the class lists after the final run. If another microcomputer such as the IBM XT, with a larger main memory was used, it might be possible to improve some of the scheduling algorithms and reduce run time.

7.3 LESSONS LEARNED

7.3.1. ALTERNATIVE APPROACHES FOR IMPROVED SYSTEM

This scheduling program could be modified to work more efficiently and run faster on a microcomputer with a larger main memory.

7.3.2. SUGGESTIONS OF FUTURE EXTENSIONS

Many extensions could be added which would greatly simplify the scheduling process. Some of these are:

1. An extension could be written to schedule all study halls that a student would have in his/her unscheduled time.

2. A report card and attendance extension could make this a complete system to replace the services now offered by BOCES and used by many districts. Local districts contract with BOCES for scheduling, attendance and report card services. The three services come as a package so the local district would have to replace all three services.

3. An extension to update all schedules and class lists as students add and drop courses would be of great value. Since the students' schedules are stored in the schedule file, an extension could be written to enable the user to change all courses in an individual student's schedule and rewrite the schedule to the schedule file. Once this schedule was available then the Printclasslists and Printschedule programs could be used to generate new class lists and schedules. This would allow a school to print completely accurate class lists and schedules for second semester (January). These second semester class lists and schedules are currently printed at schedule run-time in August and as a result they are very inaccurate by January.

4. Another extension could be written to allow the guidance counselors to generate a schedule for a single student. The student could be added to the appropriate class lists and the schedule could then be sent to a printer

so that the student could leave with a copy of the schedule. This would be especially useful during the first days of school to enroll new students.

5. It would be possible to write an extension to allow the school offices to have some on-line query capabilities of student schedules. The five offices at Sodus High School currently use handwritten copies of student schedules which are often inaccurate.

7.3.3 FUTURE DIRECTIONS IN SCHEDULING

Computers started to play a role in scheduling of high school students in the 1960's. There was a great deal of research into scheduling theory at that time and most schools began to do their scheduling using computers. Some companies provided scheduling services to schools but this was often very expensive. As a result schools began to share the use of computer services through BOCES. These services included payroll, scheduling, attendance and report cards.

In the 1960's and 1970's the climate of schools began to change and the curriculum began to expand to include more electives and free time for students. Many scheduling patterns such as flexible modular scheduling were developed using computers, which allowed schools to develop schedules to accommodate the many changes in schools. This lead to the development of open schools and other innovative approaches to education.

After the big push for change started to wear off, schools in the past few years have pulled back to more

traditional patterns and schedules. The literature shows a reduction in approaches to school scheduling in the late 1970's and early 1980's but in the past few years schools are starting to use microcomputers both in the classroom and in doing administrative tasks. This has lead to an increased interest in scheduling and in the next few years many programs will be written to do scheduling for schools on microcomputers.

8. BIBLIOGRAPHY

Braddock, Clayton, "Changing Times are Changing Schools," Southern Education Report , V3N3 (OCT 1967) 8P.
In this article the author examines some of the advantages and disadvantages of modular or flexible class scheduling.

Chaffee, Dr. Leonard M. and Dr. Robert W. Heller, "Analysis of Selected Factors Relative to Automated School Scheduling Process," State University of New York, Albany, ERIC ED 019 754, 1967.
This project (Project PASS) utilized the CLASS and GASP scheduling techniques to provide in-service education for school personnel.

Dempsey, Richard and Henry Traverso, "Scheduling the Secondary School," NASSP, 1983.
The authors provide technical information and a common sense approach to the scheduling process.

Durward, M. Lynne, "Computerized Scheduling in Vancouver Schools. A Research Report.," Vancouver Board of School Trustees (British Columbia), ERIC ED 088 216, April 1973.
This study examined computerized scheduling in Vancouver Secondary Schools and analyzed the results.

Harding, Robert Elton, Problem of Generating Class Schedules for Schools . Ann Arbor, Michigan: University Microfilms, Inc., 1969.
This dissertation reported research in scheduling theory and examined the use of analytical models to solve the scheduling problem.

Keill, James, "Project Admire (Assistance for Decision Making Through Information Retrieval in Education), Principal's Manual for Pupil Scheduling by Computer," Office of Education (DHEW), Washington, D.C., ERIC ED 022 243, 1967.
A step-by-step process is described for registering students in Junior and Senior High Schools by computer.

Oakford, Robert V. and Dwight W. Allen, "Flexibility for Vocational Education Through Computer Scheduling. Final Report," Office of Education (DHEW), Washington, D.C., ERIC ED 029 952, 1968.
An examination of the impact of the Stanford Scheduling System on 18 Secondary Schools.

Simair, Dennis J., "Computer Uses in School Administration: A pilot Project," British Journal of Educational Technology , NO2 Vol13 (May , 1982), pp114-128.
This report describes the implementation of a project which used microcomputers in administration.

Swaab, Alexander M., School Administrator's Guide to Flexible Modular Scheduling . West Nyack, New York: Parker Publishing Company, 1974.
This book progresses from an overall view of traditional schedules to the concept of Flexible Modular Scheduling.

9. APPENDICES

The following pages contain samples of the indicated reports.

9.1. LIST OF STUDENT RECORDS

9.1.1. GENERAL INFORMATION

GENERAL INFORMATION STUDENT RECORDS

STUDENT NAME	ST.NO.	HR	BIRTH	PHONE	SEX
ABARE ANN 123 MILL ST. SODUS 14551	21610	113	9/1/70	483-9999	F
ALLEN CRYSTAL 123 MILL ST. SODUS 14551	21465	113	9/1/70	483-9999	F
ALLEN MATTHEW 123 MILL ST. SODUS 14551	21465	113	9/1/70	483-9999	M
BARCLAY TINA 123 MILL ST. SODUS 14551	22532	113	9/1/70	483-9999	F
BAUMGARTNER MARK 123 MILL ST. SODUS 14551	22556	113	9/1/70	483-9999	M
BAUMGARTNER MICHAEL 123 MILL ST. SODUS 14551	22537	113	9/1/70	483-9999	M
BAURER JAMES 123 MILL ST. SODUS 14551	21471	113	9/1/70	483-9999	M
BECKER DONNA 123 MILL ST. SODUS 14551	22783	113	9/1/70	483-9999	F
BRANCH HEIDI 123 MILL ST. SODUS 14551	22807	113	9/1/70	483-9999	F
BRETT CHARITY 123 MILL ST. SODUS 14551	22557	113	9/1/70	483-9999	F
BROWN KRISTIN 123 MILL ST. SODUS 14551	21476	113	9/1/70	483-9999	F
BROWN TROY 123 MILL ST. SODUS 14551	20563	113	9/1/70	483-9999	M
BULLOCK JOHN 123 MILL ST. SODUS 14551	21477	113	9/1/70	483-9999	M
BURNAP KENDRA 123 MILL ST. SODUS 14551	22587	113	9/1/70	483-9999	F

9.1.2. STUDENT COURSE REQUEST LIST

STUDENT COURSE REQUEST LIST

STUDENT NAME	GR	ST.NO.
ABARE ANN	9	21610
701 505 462 308 205 105 6		
ALLEN CRYSTAL	9	21465
701 410 309 205 106 73 52 7		
ALLEN MATTHEW	9	21465
701 410 307 204 105 6		
BARCLAY TINA	9	22532
701 555 554 410 308 204 105 6		
BAUMGARTNER MARK	9	22556
701 462 410 307 204 105 6		
BAUMGARTNER MICHAEL	9	22537
701 410 307 255 203 105 6		
BAURER JAMES	9	21471
701 603 404 308 204 105 6		
BECKER DONNA	9	22783
701 655 654 410 308 204 105 6		
BRANCH HEIDI	9	22807
701 464 404 308 205 105 6		
BRETT CHARITY	9	22557
701 551 550 309 206 106 7		
BROWN KRISTIN	9	21476
701 462 404 308 205 105 6		
BROWN TROY	9	20563
701 505 503 309 206 106 73 52 7		
BULLOCH JOHN	9	21477
701 505 503 309 206 106 73 52 7		
BURNAP KENDRA	9	22587
701 654 464 404 307 204 105 6		
CASTIGLIONE JODY	9	22677
701 655 654 462 404 307 204 105 6		
CASTLE THOMAS	9	21481
701 404 308 204 105 6		
CLEVELAND GREGORY	9	22580
701 462 410 307 204 105 6		
COBB MICHAEL	9	22633
701 655 404 307 204 105 6		
CONROW DONNA	9	22578
701 551 550 307 206 106 73 52 7		

9.2. CONFLICT MATRIX

CONFLICT MATRIX FOR COURSE 100

CRS NAME	CODE	TALLY
ENG 12	1	8
ENG 11-2	2	1
ENG 11-3	3	1
AP ENGLISH	9	3
SS IND STD	100	11
AM STD 3	102	1
LAW	107	4
MATH 12-1	200	2
MATH 12-2	201	3
MATH CRS 3	202	2
COMP PRG 4	250	1
COMP-AWARE	255	5
ADV BIO	300	1
PHYSICS 2	301	4
BIOLOGY 3	306	2
SPAN 1	410	1
OFF PROCED	451	1
ACCT	453	1
BUS LAW	455	1
PER TYPING	462	1
MECH DWG	507	2
PHOTOG	604	3
YEARBOOK	605	1
CON BAND	654	1
CON CHOIR	655	1
PE 11-12	700	8
PE 9-10	701	2
PE BOCES	702	1
DR ED TH	750	6
DR ED TH	751	1
PSYCH	790	4

CONFLICT MATRIX FOR COURSE 901

CRS NAME	CODE	TALLY
ENG 12	1	40
ENG 11-2	2	3
ENG 11-3	3	12
WRIT LAB	52	3
WRIT 11-12	53	2
READING 10	72	2
READ 11-12	74	1
AM STD 2	101	3
AM STD 3	102	1
E CUL ST 3	104	1
LAW	107	2
AMER STD 3	110	14
MATH CRS 3	202	2
MATH CRS 1	204	1
COMP-AWARE	255	4
CHEM 2	303	1
SPAN 1	410	1
PER TYPING	462	2
WORD PROC	463	2
APP CERAM	500	2
BAS CERAM	501	3
APP WOOD	504	1
ARCH DWG	506	1
MECH DWG 2	510	2
BAS FOODS	553	3
CH CARE 1	555	2
STUDIO ART	603	1
PHOTOG	604	1
CON CHOIR	655	1
PE 11-12	700	27
PE 9-10	701	5
PE BOCES	702	5
DR ED TH	750	6
DR ED TH	751	20
HEALTH 11	770	12
PSYCH	790	6
BOCES PM	901	40

9.3. COURSE TALLIES LIST

SODUS
STUDENT TALLY OF COURSES OFFERED

COURSE	CODE	TOTAL	GR 9	GR 10	GR 11	GR 12
ENG 12	1	91	0	0	0	91
ENG 11-2	2	78	0	2	68	8
ENG 11-3	3	57	0	3	37	17
ENG 10-2	4	87	3	83	1	0
ENG 10-3	5	57	13	40	4	0
ENG 9-2	6	100	98	2	0	0
ENG 9-3	7	46	46	0	0	0
AP ENGLISH	9	12	0	0	0	12
WRITING 12	50	1	0	0	0	1
WRITING 11	51	7	0	2	5	0
WRIT LAB	52	80	43	34	0	3
WRIT 11-12	53	11	0	1	8	2
READING 11	71	5	0	0	5	0
READING 10	72	25	2	21	0	2
READING 9	73	44	42	2	0	0
READ 11-12	74	10	0	1	8	1
SS IND STD	100	11	0	0	0	11
AM STD 2	101	75	0	0	69	6
AM STD 3	102	42	0	1	32	9
E CUL ST 2	103	85	2	81	2	0
E CUL ST 3	104	56	8	44	3	1
AFR-ASIA 2	105	100	100	0	0	0
AFR-ASIA 3	106	52	48	3	1	0
LAW	107	15	0	0	2	13
AMER STD 3	110	18	0	0	2	16
MATH 12-1	200	15	0	0	0	15
MATH 12-2	201	25	0	0	8	17
MATH CRS 3	202	52	0	8	36	8
MATH CRS 2	203	56	7	46	3	0
MATH CRS 1	204	103	72	27	3	1
MATH 9-3	205	47	36	10	0	1
GEN MATH	206	39	35	4	0	0
C PREP M	207	5	2	2	1	0
COMP PRG 4	250	6	0	0	1	5
COMP PROG	251	1	0	0	1	0
COMP PRG 2	252	12	0	2	6	4
COMP PRG 1	253	15	1	10	3	1
COMP-AWARE	255	48	1	14	11	22
ADV BIO	300	21	0	0	15	6
PHYSICS 2	301	20	0	0	0	20
CHEM 2	303	54	0	23	28	3
BIOLOGY 2	305	57	0	41	16	0
BIOLOGY 3	306	12	1	8	0	3
ER SCI R	307	39	39	0	0	0
SCI 9-2	308	64	61	3	0	0
SCI 9-3	309	61	54	5	1	1
FRENCH 3	402	12	0	2	10	0
FRENCH 2	403	24	0	23	1	0
FRENCH 1	404	36	31	3	1	1
SPAN 3	407	16	0	7	7	2
SPAN 2	408	18	1	16	1	0
SPAN 1	410	39	35	1	1	2
OFF PROCED	451	16	0	0	0	16
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

ACCT	453	27	1	13	7	6
BUS LAW	455	8	0	0	3	5
SHRT/TRANS	457	7	0	0	3	4
SHRT TH	458	14	0	8	2	4
BUS DYN	459	19	5	6	3	5
ADV KEYBD	460	13	0	9	3	1
PER TYPING	462	29	20	2	1	6
WORD PROC	463	15	0	0	6	9
KEY-COM	464	40	24	14	1	1
APP CERAM	500	25	3	12	5	5
BAS CERAM	501	34	3	14	7	10
APP METAL	502	17	1	14	2	0
BAS METAL	503	38	31	7	0	0
APP WOOD	504	20	1	18	0	1
BAS WOOD	505	45	36	7	0	2
ARCH DWG	506	4	0	0	2	2
MECH DWG	507	49	17	23	5	4
MECH DWG 2	510	5	0	0	3	2
ADV CLOTH	550	11	9	2	0	0
BAS SEWING	551	15	11	2	1	1
FAM MEAL	552	9	6	3	0	0
BAS FOODS	553	13	7	2	0	4
CH CARE 2	554	14	9	4	0	1
CH CARE 1	555	16	10	4	0	2
HOME EC 1	556	3	1	1	0	1
ART IND ST	600	2	0	0	1	1
DWP/PTNG	601	10	1	4	5	0
STUDIO ART	603	30	23	6	0	1
PHOTOG	604	37	0	10	6	21
YEARBOOK	605	10	2	0	0	8
CON BAND	654	57	16	15	21	5
CON CHOIR	655	15	7	3	2	3
HB CHOIR	656	14	6	3	3	2
PE 11-12	700	156	0	2	77	77
PE 9-10	701	294	155	109	15	15
PE BOCES	702	40	0	15	17	8
DR ED TH	750	73	0	0	21	52
DR ED TH	751	31	0	0	9	22
HEALTH 11	770	101	0	6	77	18
HEALTH 11	772	38	0	13	23	2
PSYCH	790	25	0	1	4	20
BOCES AM	900	52	0	17	35	0
BOCES PM	901	40	0	0	0	40
LUNCH 7B	999	0	0	0	0	0

9.4. COURSE LIST

COURSE LIST - SODUS

CODE	SEC	SEATS	SEM	PD	DAYS MET	DESCRIPTION	ROOM	TCH	PER
1	1	200	3	11	MTWRF	ENG 12	N	1	8
2	1	35	3	5	MTWRF	ENG 11-2	125	5	5
2	1	35	3	6	MTWRF	ENG 11-2	125	5	5
2	2	35	3	7	MTWRF	ENG 11-2	125	1	6
2	2	35	3	8	MTWRF	ENG 11-2	125	1	6
2	3	35	3	9	MTWRF	ENG 11-2	125	8	7
2	3	35	3	10	MTWRF	ENG 11-2	125	8	8
2	4	35	3	13	MTWRF	ENG 11-2	125	2	10
3	1	35	3	4	MTWRF	ENG 11-3	125	8	4
3	2	35	3	3	MTWRF	ENG 11-3	125	8	3
3	3	35	3	12	MTWRF	ENG 11-3	125	6	9
4	1	35	3	7	MTWRF	ENG 10-2	129	4	6
4	1	35	3	8	MTWRF	ENG 10-2	129	4	6
4	2	35	3	4	MTWRF	ENG 10-2	123	4	4
4	3	35	3	9	MTWRF	ENG 10-2	129	9	7
4	3	35	3	10	MTWRF	ENG 10-2	129	9	7
4	4	35	3	13	MTWRF	ENG 10-2	129	9	10
5	1	35	3	5	MTWRF	ENG 10-3	220	3	5
5	1	35	3	6	MTWRF	ENG 10-3	220	3	5
5	2	35	3	3	MTWRF	ENG 10-3	129	9	3
5	3	35	3	12	MTWRF	ENG 10-3	129	9	9
6	1	35	3	3	MTWRF	ENG 9-2	220	10	3
6	2	35	3	7	MTWRF	ENG 9-2	220	10	6
6	2	35	3	8	MTWRF	ENG 9-2	220	10	6
6	3	35	3	13	MTWRF	ENG 9-2	219	5	10
6	4	35	3	12	MTWRF	ENG 9-2	220	5	9
6	5	35	3	9	MTWRF	ENG 9-2	220	3	7
6	5	35	3	10	MTWRF	ENG 9-2	220	3	7
7	1	35	3	5	MTWRF	ENG 9-3	129	4	5
7	1	35	3	6	MTWRF	ENG 9-3	129	4	5
7	2	35	3	4	MTWRF	ENG 9-3	220	10	4
7	3	35	3	13	MTWRF	ENG 9-3	123	4	10
9	1	20	3	11	MTWRF	AP ENGLISH	129	9	8

9.5. RESULTS - TRIAL AND FINAL RUN

RUN - FINAL

RESULTS FOR GRADE 9

FULL SCHEDULES	- 157
PARTIAL SCHEDULES	- 0
IRR CONFLICTS	- 0
TOTAL	- 157

STUDY HALL COUNT

1ST SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	6	35	6	29	35
4	16	16	16	16	16
5	29	60	29	60	29
6	48	79	48	79	48
7	46	30	46	30	47
8	40	24	40	24	41
9	31	30	31	30	34
10	10	9	10	9	13
11	1	30	1	30	30
12	47	8	47	14	8
13	17	22	17	17	22

2ND SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	4	33	4	27	10
4	18	18	18	18	12
5	23	54	23	54	55
6	42	73	42	73	74
7	57	41	57	41	41
8	51	35	51	35	35
9	31	30	31	30	30
10	10	9	10	9	9
11	1	30	1	30	1
12	48	9	48	15	50
13	16	21	16	16	21

NORTIER STEPHANIE

PARTIAL SCHEDULE - COURSE 103 IS NOT SCHEDULED

1ST SEM SCHEDULE

1	0	0	0	0	0
2	654	0	654	0	654
3	453	453	453	453	453
4	4	4	4	4	4
5	999	999	999	999	999
6	0	0	0	0	0
7	403	403	403	403	403
8	403	403	403	403	403
9	203	203	203	203	203
10	203	203	203	203	203
11	305	305	305	305	305
12	700	305	700	305	0
13	0	0	0	0	0

2ND SEM SCHEDULE

1	0	0	0	0	0
2	654	0	654	0	654
3	453	453	453	453	453
4	4	4	4	4	4
5	999	999	999	999	999
6	0	0	0	0	0
7	403	403	403	403	403
8	403	403	403	403	403
9	203	203	203	203	203
10	203	203	203	203	203
11	305	305	305	305	305
12	700	305	700	305	700
13	0	0	0	0	0

RHINE DAVID

PARTIAL SCHEDULE - COURSE 403 IS NOT SCHEDULED

1ST SEM SCHEDULE

1	0	0	0	0	0
2	654	0	654	0	654
3	103	103	103	103	103
4	0	0	0	0	0
5	0	0	0	0	0
6	999	999	999	999	999
7	4	4	4	4	4
8	4	4	4	4	4
9	203	203	203	203	203
10	203	203	203	203	203
11	303	303	303	303	303
12	305	303	305	303	0
13	305	305	305	305	305

2ND SEM SCHEDULE

1	0	0	0	0	0
2	654	0	654	0	654
3	103	103	103	103	103
4	0	0	0	0	0
5	0	0	0	0	0
6	999	999	999	999	999
7	4	4	4	4	4
8	4	4	4	4	4
9	203	203	203	203	203
10	203	203	203	203	203
11	303	303	303	303	303
12	305	303	305	303	0
13	305	305	305	305	305

VANDE BUGART PATRIN

PARTIAL SCHEDULE - COURSE 303 IS NOT SCHEDULED

1ST SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	103	103	103	103	103
4	0	0	0	0	0
5	0	0	0	0	0
6	999	999	999	999	999
7	403	403	403	403	403
8	403	403	403	403	403
9	204	204	204	204	204
10	204	204	204	204	204
11	701	0	701	0	0
12	603	603	603	603	603
13	4	4	4	4	4

2ND SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	103	103	103	103	103
4	0	0	0	0	0
5	0	0	0	0	0
6	999	999	999	999	999
7	403	403	403	403	403
8	403	403	403	403	403
9	204	204	204	204	204
10	204	204	204	204	204
11	701	0	701	0	701
12	603	603	603	603	603
13	4	4	4	4	4

RUN - FINAL
RESULTS FOR GRADE 10

FULL SCHEDULES	- 124
PARTIAL SCHEDULES	- 3
IRR CONFLICTS	- 0
TOTAL	- 127

STUDY HALL COUNT

1ST SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	4	16	4	16	16
4	23	15	23	15	24
5	25	21	25	21	21
6	25	21	25	21	21
7	23	23	23	23	28
8	27	27	27	27	32
9	30	29	30	29	36
10	29	28	29	28	35
11	8	34	8	34	34
12	32	16	32	19	45
13	33	44	33	33	44

2ND SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	4	17	4	17	4
4	22	14	22	14	16
5	25	20	25	20	22
6	25	20	25	20	22
7	26	24	26	24	24
8	30	28	30	28	28
9	33	32	33	32	32
10	32	31	32	31	31
11	8	34	8	34	8
12	31	17	31	20	50
13	31	42	31	31	42

BURNAP KIRSTEN

PARTIAL SCHEDULE - COURSE 101 IS NOT SCHEDULED

1ST SEM SCHEDULE

1	0	0	0	0	0
2	654	0	654	0	654
3	253	253	253	253	253
4	303	700	303	700	0
5	303	303	303	303	303
6	303	303	303	303	303
7	2	2	2	2	2
8	2	2	2	2	2
9	999	999	999	999	999
10	0	0	0	0	0
11	0	0	0	0	0
12	0	770	0	770	770
13	202	202	202	202	202

2ND SEM SCHEDULE

1	0	0	0	0	0
2	654	0	654	0	654
3	253	253	253	253	253
4	303	700	303	700	700
5	303	303	303	303	303
6	303	303	303	303	303
7	2	2	2	2	2
8	2	2	2	2	2
9	999	999	999	999	999
10	0	0	0	0	0
11	0	0	0	0	0
12	0	0	0	0	0
13	202	202	202	202	202

CAMPBELL WILLIAM

PARTIAL SCHEDULE COURSE 303 IS NOT SCHEDULED

1ST SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	0	770	0	770	770
4	0	700	0	700	0
5	0	0	0	0	0
6	999	999	999	999	999
7	202	202	202	202	202
8	202	202	202	202	202
9	2	2	2	2	2
10	2	2	2	2	2
11	0	0	0	0	0
12	0	0	0	0	0
13	101	101	101	101	101

2ND SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	700	0	700	700
5	0	0	0	0	0
6	999	999	999	999	999
7	202	202	202	202	202
8	202	202	202	202	202
9	2	2	2	2	2
10	2	2	2	2	2
11	0	0	0	0	0
12	510	510	510	510	510
13	101	101	101	101	101

DEPAUL KENNETH

PARTIAL SCHEDULE - COURSE 999 IS NOT SCHEDULED

1ST SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	101	101	101	101	101
4	0	0	0	0	0
5	0	700	0	700	0
6	0	700	0	700	0
7	202	202	202	202	202
8	202	202	202	202	202
9	2	2	2	2	2
10	2	2	2	2	2
11	0	0	0	0	0
12	305	0	305	0	0
13	305	305	305	305	305

2ND SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	101	101	101	101	101
4	0	0	0	0	0
5	0	700	700	700	700
6	0	700	700	700	700
7	202	202	202	202	202
8	202	202	202	202	202
9	2	2	2	2	2
10	2	2	2	2	2
11	0	0	0	0	0
12	305	0	305	0	0
13	305	305	305	305	305

RUN - FINAL
RESULTS FOR GRADE 11

FULL SCHEDULES	- 106
PARTIAL SCHEDULES	- 3
IRR CONFLICTS	- 0
TOTAL	- 109

STUDY HALL COUNT

1ST SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	15	6	15	6	6
4	38	27	38	27	47
5	16	4	16	4	16
6	29	17	29	17	29
7	15	20	15	20	15
8	0	5	0	5	0
9	26	22	26	22	26
10	27	23	27	23	27
11	50	53	50	53	53
12	17	15	17	15	23
13	8	8	8	8	8

2ND SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	15	18	15	18	15
4	58	27	58	27	44
5	34	4	29	4	4
6	47	17	42	17	17
7	21	20	21	20	20
8	6	5	6	5	5
9	26	22	26	22	22
10	27	23	27	23	23
11	49	52	49	52	49
12	16	28	16	28	25
13	8	8	8	8	8

HUBER RICHARD

PARTIAL SCHEDULE - COURSE 900 IS NOT SCHEDULED

1ST SEM SCHEDULE

1	110	110	110	110	110
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	999	999	999	999	999
6	0	0	0	0	0
7	0	701	0	701	0
8	0	701	0	701	0
9	504	504	504	504	504
10	504	504	504	504	504
11	1	1	1	1	1
12	3	3	3	3	3
13	0	0	0	0	0

2ND SEM SCHEDULE

1	110	110	110	110	110
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	999	999	999	999	999
6	0	0	0	0	0
7	0	701	0	701	701
8	0	701	0	701	701
9	0	0	0	0	0
10	0	0	0	0	0
11	1	1	1	1	1
12	3	3	3	3	3
13	0	0	0	0	0

MOON KIM

PARTIAL SCHEDULE - COURSE 999 IS NOT SCHEDULED

1ST SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	457	457	457	457	457
4	0	0	0	0	0
5	0	700	0	700	0
6	0	700	0	700	0
7	463	463	463	463	463
8	463	463	463	463	463
9	2	2	2	2	2
10	2	2	2	2	2
11	1	1	1	1	1
12	303	0	303	0	0
13	303	303	303	303	303

2ND SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	457	457	457	457	457
4	0	0	0	0	0
5	0	700	750	700	700
6	0	700	750	700	700
7	463	463	463	463	463
8	463	463	463	463	463
9	2	2	2	2	2
10	2	2	2	2	2
11	1	1	1	1	1
12	303	0	303	0	0
13	303	303	303	303	303

O NEIL ALEXANDAR

PARTIAL SCHEDULE - COURSE 999 IS NOT SCHEDULED

1ST SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	300	300	300	300	300
4	0	300	0	300	0
5	0	700	0	700	0
6	0	700	0	700	0
7	551	551	551	551	551
8	551	551	551	551	551
9	0	0	0	0	0
10	0	0	0	0	0
11	9	9	9	9	9
12	0	0	0	0	0
13	107	107	107	107	107

2ND SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	300	300	300	300	300
4	0	300	0	300	0
5	0	700	750	700	700
6	0	700	750	700	700
7	255	255	255	255	255
8	255	255	255	255	255
9	604	604	604	604	604
10	604	604	604	604	604
11	9	9	9	9	9
12	0	0	0	0	0
13	107	107	107	107	107

SMITH PATRICIA

PARTIAL SCHEDULE - COURSE 999 IS NOT SCHEDULED

1ST SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	0	770	0	770	770
4	102	102	102	102	102
5	0	700	0	700	0
6	0	700	0	700	0
7	463	463	463	463	463
8	463	463	463	463	463
9	458	458	458	458	458
10	458	458	458	458	458
11	1	1	1	1	1
12	0	0	0	0	0
13	0	0	0	0	0

2ND SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	102	102	102	102	102
5	0	700	750	700	700
6	0	700	750	700	700
7	463	463	463	463	463
8	463	463	463	463	463
9	458	458	458	458	458
10	458	458	458	458	458
11	1	1	1	1	1
12	0	0	0	0	0
13	0	0	0	0	0

RUN - FINAL

RESULTS FOR GRADE 12

FULL SCHEDULES - 99
PARTIAL SCHEDULES - 4
IRR CONFLICTS - 0
TOTAL - 103

STUDY HALL COUNT

1ST SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	20	16	20	16	16
4	47	24	47	24	31
5	20	3	20	3	20
6	25	8	25	8	25
7	29	31	29	31	29
8	12	14	12	14	12
9	41	31	41	31	41
10	59	49	59	49	59
11	0	0	0	0	0
12	22	54	22	54	37
13	53	55	53	53	55

2ND SEM SCHEDULE

1	0	0	0	0	0
2	0	0	0	0	0
3	20	18	20	18	20
4	48	25	48	25	31
5	20	2	8	2	2
6	25	7	13	7	7
7	47	38	47	38	38
8	30	21	30	21	21
9	41	31	41	31	31
10	59	49	59	49	49
11	0	0	0	0	0
12	26	61	26	61	29
13	38	40	38	38	40

COURSES IN NUMERICAL ORDER ARE :

CODE	SEC	SEATS	SEM	PD	DAYS MET	DESCRIPTION	ROOM	TCH	PER	ENROL
1	1	200	3	11	MTWRF	ENG 12	N	1	8	91
2	1	25	3	5	MTWRF	ENG 11-2	125	5	5	16
2	1	25	3	6	MTWRF	ENG 11-2	125	5	5	16
2	2	25	3	7	MTWRF	ENG 11-2	125	1	6	22
2	2	25	3	8	MTWRF	ENG 11-2	125	1	6	22
2	3	25	3	9	MTWRF	ENG 11-2	125	8	7	24
2	3	25	3	10	MTWRF	ENG 11-2	125	8	8	24
2	4	25	3	13	MTWRF	ENG 11-2	125	2	10	16
3	1	25	3	4	MTWRF	ENG 11-3	125	8	4	10
3	2	25	3	3	MTWRF	ENG 11-3	125	8	3	26
3	3	25	3	12	MTWRF	ENG 11-3	125	6	9	21
4	1	25	3	7	MTWRF	ENG 10-2	129	4	6	16
4	1	25	3	8	MTWRF	ENG 10-2	129	4	6	16
4	2	25	3	4	MTWRF	ENG 10-2	123	4	4	22
4	3	25	3	4	MTWRF	ENG 10-2	129	9	4	22
4	4	25	3	13	MTWRF	ENG 10-2	129	9	10	26
5	1	25	3	5	MTWRF	ENG 10-3	220	3	5	15
5	1	25	3	6	MTWRF	ENG 10-3	220	3	5	15
5	2	25	3	3	MTWRF	ENG 10-3	129	9	3	20
5	4	25	3	12	MTWRF	ENG 10-3	129	9	9	21
6	1	25	3	3	MTWRF	ENG 9-2	220	10	3	20
6	2	25	3	7	MTWRF	ENG 9-2	220	10	6	11
6	2	25	3	8	MTWRF	ENG 9-2	220	10	6	11
6	3	25	3	13	MTWRF	ENG 9-2	219	5	10	21
6	4	25	3	12	MTWRF	ENG 9-2	220	5	9	21
6	5	25	3	9	MTWRF	ENG 9-2	220	3	7	5
6	5	25	3	10	MTWRF	ENG 9-2	220	3	7	5
6	6	25	3	13	MTWRF	ENG 9-2	220	10	3	22
7	1	25	3	5	MTWRF	ENG 9-3	129	4	5	14
7	1	25	3	6	MTWRF	ENG 9-3	129	4	5	14
7	2	25	3	4	MTWRF	ENG 9-3	220	10	4	15
7	3	25	3	13	MTWRF	ENG 9-3	123	4	10	17
9	1	20	3	11	MTWRF	AP ENGLISH	129	9	8	12
50	1	10	3	4	M W	WRITING 12	228	1	4	1
51	1	10	3	4	M W	WRITING 11	228	1	4	4
51	2	10	3	3	M W	WRITING 11	228	1	3	0
52	1	10	3	3	T F	WRIT LAB	228	3	3	9
52	2	9	3	4	T R	WRIT LAB	228	6	4	8
52	3	9	3	5	T R	WRIT LAB	228	6	5	3
52	3	9	3	6	T R	WRIT LAB	228	6	5	3
52	4	9	3	5	M W	WRIT LAB	228	6	5	7
52	4	9	3	6	M W	WRIT LAB	228	6	5	7
52	5	9	3	7	M W	WRIT LAB	228	6	6	6
52	5	9	3	8	M W	WRIT LAB	228	6	6	6
52	6	9	3	12	M W	WRIT LAB	222	1	9	9
52	7	9	3	9	M W	WRIT LAB	228	6	7	9
52	7	9	3	10	M W	WRIT LAB	228	6	7	9
52	8	9	3	12	T F	WRIT LAB	228	3	9	9
52	9	9	3	13	T F	WRIT LAB	228	6	10	9
53	1	10	3	1	T R	WRIT 11-12	228	1	1	10
71	1	10	3	4	T R	READING 11	209	15	4	4
72	1	20	3	5	T RF	READING 10	207	1	5	11
72	1	20	3	6	T RF	READING 10	207	1	5	11
72	2	20	3	13	M WR	READING 10	209	15	10	14
73	1	25	3	3	M WR	READING 9	207	17	3	15
73	2	22	3	4	M W F	READING 9	207	17	4	12
73	3	22	3	13	M WR	READING 9	207	17	10	13
74	1	15	3	1	M W F	READ 11-12	228	15	1	9

100	1	30	3	12	MTWRF	SS IND STD	222	24	9	11
101	1	35	3	3	MTWRF	AM STD 2	213	20	3	33
101	2	35	3	7	MTWRF	AM STD 2	219	24	6	10
101	2	35	3	8	MTWRF	AM STD 2	219	24	6	10
101	3	35	3	12	MTWRF	AM STD 2	213	20	9	18
101	4	35	3	13	MTWRF	AM STD 2	215	24	10	12
102	1	35	3	4	MTWRF	AM STD 3	213	20	4	20
102	2	35	3	5	MTWRF	AM STD 3	213	23	5	20
103	1	35	3	3	MTWRF	E CUL STD	219	26	3	20
103	2	35	3	5	MTWRF	E CUL ST 2	219	26	5	17
103	2	35	3	6	MTWRF	E CUL ST 2	219	26	5	17
103	3	35	3	9	MTWRF	E CUL ST 2	219	26	7	20
103	3	35	3	10	MTWRF	E CUL ST 2	219	26	7	20
103	4	35	3	11	MTWRF	E CUL ST 2	222	24	8	26
104	1	35	3	4	MTWRF	E CUL ST 3	222	24	4	19
104	2	35	3	11	MTWRF	E CUL ST 3	219	26	8	18
104	3	35	3	12	MTWRF	E CUL ST 3	219	26	9	17
105	1	35	3	7	MTWRF	AFR-ASIA 2	215	23	6	20
105	1	35	3	8	MTWRF	AFR-ASIA 2	215	23	6	20
105	2	35	3	4	MTWRF	AFR-ASIA 2	215	22	4	20
105	3	35	3	9	MTWRF	AFR-ASIA 2	215	22	7	21
105	3	35	3	10	MTWRF	AFR-ASIA 2	215	22	7	21
105	4	35	3	12	MTWRF	AFR-ASIA 2	215	23	9	21
105	5	35	3	11	MTWRF	AFR-ASIA 2	213	23	8	18
106	1	35	3	3	MTWRF	AFR-ASIA 3	215	22	3	19
106	2	35	3	5	MTWRF	AFR-ASIA 3	215	22	5	13
106	2	35	3	6	MTWRF	AFR-ASIA 3	215	22	5	13
106	3	35	3	11	MTWRF	AFR-ASIA 3	215	22	8	20
107	1	35	3	13	MTWRF	LAW	213	20	10	12
110	1	35	3	1	MTWRF	AMER STD 3	213	23	1	19
200	1	35	3	5	MTWRF	MATH 12-1	221	30	5	15
200	1	35	3	6	MTWRF	MATH 12-1	221	30	5	15
201	1	35	3	3	MTWRF	MATH 12-2	222	30	3	12
201	2	35	3	13	MTWRF	MATH 12-2	221	30	10	12
202	1	35	3	13	MTWRF	MATH CRS 3	224	32	10	23
202	2	35	3	7	MTWRF	MATH CRS 3	224	32	6	22
202	2	35	3	8	MTWRF	MATH CRS 3	224	32	6	22
203	1	35	3	3	MTWRF	MATH CRS 2	221	33	3	18
203	2	35	3	7	MTWRF	MATH CRS 2	221	36	6	18
203	2	35	3	8	MTWRF	MATH CRS 2	221	36	6	18
203	3	35	3	9	MTWRF	MATH CRS 2	221	33	7	19
203	3	35	3	10	MTWRF	MATH CRS 2	221	33	7	19
204	1	35	3	3	MTWRF	MATH CRS 1	224	32	3	23
204	2	35	3	4	MTWRF	MATH CRS 1	221	31	4	18
204	3	35	3	4	MTWRF	MATH CRS 1	224	32	4	18
204	4	35	3	9	MTWRF	MATH CRS 1	224	32	7	19
204	4	35	3	10	MTWRF	MATH CRS 1	224	32	7	19
204	5	35	3	12	MTWRF	MATH CRS 1	221	33	9	21
205	1	35	3	9	MTWRF	MATH 9-3	217	37	7	17
205	1	35	3	10	MTWRF	MATH 9-3	217	37	7	17
205	2	35	3	11	MTWRF	MATH 9-3	224	37	8	16
205	3	35	3	12	MTWRF	MATH 9-3	217	37	9	13
206	1	30	3	7	MTWRF	GEN MATH	217	37	6	18
206	1	30	3	8	MTWRF	GEN MATH	217	37	6	18
206	2	30	3	13	MTWRF	GEN MATH	222	35	10	19
207	1	20	3	12	MTWRF	C PREP M	116	35	9	5
250	1	20	3	4	MTWRF	COMP PRG 4	119	33	4	6
251	1	20	3	4	MTWRF	COMP PROG	119	33	4	1

252	1	20	3	12	MTWRF	COMP PRG 2	119	40	9
253	1	20	3	3	MTWRF	COMP PRG 1	119	40	3
255	1	14	2	7	MTWRF	COMP-AWARE	119	30	6
255	1	14	2	8	MTWRF	COMP-AWARE	113	30	6
255	2	14	1	9	MTWRF	COMP-AWARE	113	30	7
255	2	14	1	10	MTWRF	COMP-AWARE	113	30	7
255	3	14	1	13	MTWRF	COMP-AWARE	113	40	10
255	4	14	2	13	MTWRF	COMP-AWARE	113	40	10
300	1	30	3	3	MTWRF	ADV BIO	101	46	3
300	1	30	3	4	T R	ADV BIO	101	46	4
301	1	25	3	7	MTWRF	PHYSICS 2	108	45	6
301	1	25	3	8	MTWRF	PHYSICS 2	108	45	6
301	1	25	3	9	MTWRF	PHYSICS 2	108	45	7
303	1	25	3	11	MTWRF	CHEM 2	107	50	8
303	1	25	3	12	T R	CHEM 2	107	50	9
303	2	25	3	4	M W	CHEM 2	107	50	4
303	2	25	3	5	MTWRF	CHEM 2	107	50	5
303	2	25	3	6	MTWRF	CHEM 2	107	50	5
303	3	25	3	12	M W	CHEM 2	107	50	9
303	3	25	3	13	MTWRF	CHEM 2	107	50	10
305	1	30	3	5	MTWRF	BIOLOGY 2	101	46	5
305	1	30	3	6	MTWRF	BIOLOGY 2	101	46	5
305	1	30	3	7	MTWRF	BIOLOGY 2	101	46	6
305	2	30	3	11	MTWRF	BIOLOGY 2	101	46	8
305	2	30	3	12	T R	BIOLOGY 2	101	46	9
305	3	30	3	12	M W	BIOLOGY 2	101	46	9
305	3	30	3	13	MTWRF	BIOLOGY 2	101	46	10
306	1	25	3	3	MTWRF	BIOLOGY 3	107	50	3
307	1	30	3	4	MTWRF	ER SCI R	103	49	4
307	1	30	3	5	MTWRF	ER SCI R	103	49	5
307	2	30	3	10	MTWRF	ER SCI R	108	45	7
307	2	30	3	11	MTWRF	ER SCI R	108	45	8
308	1	30	3	3	MTWRF	SCI 9-2	103	49	3
308	2	30	3	11	MTWRF	SCI 9-2	103	49	8
308	3	30	3	13	MTWRF	SCI 9-2	103	49	10
309	1	30	3	4	MTWRF	SCI 9-3	108	45	4
309	2	30	3	7	MTWRF	SCI 9-3	103	84	6
309	2	30	3	8	MTWRF	SCI 9-3	104	84	6
309	3	30	3	9	MTWRF	SCI 9-3	103	49	7
309	3	30	3	10	MTWRF	SCI 9-3	103	49	7
402	1	30	3	5	MTWRF	FRENCH 3	227	55	5
402	1	30	3	6	MTWRF	FRENCH 3	227	55	5
403	1	30	3	7	MTWRF	FRENCH 2	227	55	6
403	1	30	3	8	MTWRF	FRENCH 2	227	55	6
404	1	30	3	13	MTWRF	FRENCH 1	227	55	10
404	2	30	3	11	MTWRF	FRENCH 1	227	55	8
407	1	30	3	13	MTWRF	SPAN 3	217	56	10
408	1	30	3	5	MTWRF	SPAN 2	217	56	5
408	1	30	3	6	MTWRF	SPAN 2	217	56	5
410	1	30	3	4	MTWRF	SPAN 1	227	56	4
410	2	30	3	9	MTWRF	SPAN 1	227	56	7
410	2	30	3	10	MTWRF	SPAN 1	227	56	7
410	3	30	3	11	MTWRF	SPAN 1	217	56	8
451	1	20	3	5	MTWRF	OFF PROCED	121	61	5
451	1	20	3	6	MTWRF	OFF PROCED	121	61	5
453	1	30	3	3	MTWRF	ACCT	123	62	3
455	1	30	3	7	MTWRF	BUS LAW	123	62	6
455	1	30	3	8	MTWRF	BUS LAW	123	62	6
457	1	30	3	3	MTWRF	SHRT/TRANS	121	60	3
458	1	30	3	9	MTWRF	SHRT TH	121	60	7
458	1	30	3	10	MTWRF	SHRT TH	121	60	7
459	1	30	3	12	MTWRF	BUS DYN	123	62	9
460	1	30	3	4	MTWRF	ADV KEYBD	121	61	4

462	1	30	1	7	MTWRF	PER TYPING	119	60	6	15
462	1	30	1	8	MTWRF	PER TYPING	119	60	6	15
462	2	30	2	5	MTWRF	PER TYPING	119	60	5	12
462	2	30	2	6	MTWRF	PER TYPING	119	60	5	12
463	1	30	3	7	MTWRF	WORD PROC	121	61	6	14
463	1	30	3	8	MTWRF	WORD PROC	121	61	6	14
464	1	30	3	13	MTWRF	KEY-COM	119	60	10	19
464	2	30	3	4	MTWRF	KEY-COM	119	60	4	19
500	1	20	2	7	MTWRF	APP CERAM	135	66	6	12
500	1	20	2	8	MTWRF	APP CERAM	135	66	6	12
500	2	20	2	13	MTWRF	APP CERAM	135	66	10	12
501	1	20	1	7	MTWRF	BAS CERAM	135	66	6	15
501	1	20	1	8	MTWRF	BAS CERAM	135	66	6	15
501	2	20	1	13	MTWRF	BAS CERAM	135	66	10	16
502	1	20	2	9	MTWRF	APP METAL	133	65	7	8
502	1	20	2	10	MTWRF	APP METAL	133	65	7	8
502	2	20	2	11	MTWRF	APP METAL	133	65	8	9
503	1	20	2	3	MTWRF	BAS METAL	133	65	3	13
503	2	20	2	4	MTWRF	BAS METAL	133	65	4	14
503	3	20	2	5	MTWRF	BAS METAL	133	65	5	11
503	3	20	2	6	MTWRF	BAS METAL	133	65	5	11
504	1	20	1	9	MTWRF	APP WOOD	135	67	7	9
504	1	20	1	10	MTWRF	APP WOOD	135	67	7	9
504	2	20	1	11	MTWRF	APP WOOD	135	67	8	8
505	1	20	1	3	MTWRF	BAS WOOD	135	67	3	11
505	2	25	1	4	MTWRF	BAS WOOD	135	67	4	16
505	3	25	1	5	MTWRF	BAS WOOD	135	67	5	15
505	3	25	1	6	MTWRF	BAS WOOD	135	67	5	15
506	1	25	1	12	MTWRF	ARCH DWG	131	66	9	4
507	1	25	3	3	MTWRF	MECH DWG	131	66	3	16
507	2	25	3	5	MTWRF	MECH DWG	131	66	5	12
507	2	25	3	6	MTWRF	MECH DWG	131	66	5	12
507	3	25	3	11	MTWRF	MECH DWG	131	66	8	16
510	1	25	2	12	MTWRF	MECH DWG 2	131	66	9	5
550	1	25	2	7	MTWRF	ADV CLOTH	H	70	6	11
550	1	25	2	8	MTWRF	ADV CLOTH	H	70	6	11
551	1	25	1	7	MTWRF	BAS SEWING	H	70	6	14
551	1	25	1	8	MTWRF	BAS SEWING	H	70	6	14
552	1	25	2	12	MTWRF	FAM MEAL	H	70	9	9
553	1	25	1	12	MTWRF	BAS FOODS	H	70	9	13
554	1	25	2	9	MTWRF	CH CARE 2	H	70	7	14
554	1	25	2	10	MTWRF	CH CARE 2	H	70	7	14
555	1	25	1	9	MTWRF	CH CARE 1	H	70	7	15
555	1	25	1	10	MTWRF	CH CARE 1	H	70	7	15
556	1	25	3	3	MTWRF	HOME EC 1	H	70	3	2
600	1	25	3	5	MTWRF	ART IND ST	223	52	5	2
600	1	25	3	6	MTWRF	ART IND ST	223	52	5	2
601	1	25	3	5	MTWRF	DWG/PTNG	223	52	5	7
601	1	25	3	6	MTWRF	DWP/PTNG	223	52	5	7
603	1	25	3	3	MTWRF	STUDIO ART	223	52	3	14
603	2	25	3	12	MTWRF	STUDIO ART	223	52	9	16
604	1	25	1	7	MTWRF	PHOTOG	223	52	6	9
604	1	25	1	8	MTWRF	PHOTOG	223	52	6	9
604	2	25	2	9	MTWRF	PHOTOG	223	52	7	10
604	2	25	2	10	MTWRF	PHOTOG	223	52	7	10
604	3	25	1	13	MTWRF	PHOTOG	223	52	10	8
604	4	25	2	13	MTWRF	PHOTOG	223	52	10	8
605	1	15	1	3	T RF	YEARBOOK	209	6	3	10
605	1	15	2	3	T R	YEARBOOK	223	6	3	10
654	1	75	3	2	M W F	CON BAND	AUD	75	2	57
655	1	35	3	1	MT RF	CON CHOIR	AUD	76	1	15
656	1	30	3	2	T	HB CHOIR	AUD	74	2	14

700	1	80	1	4	T R	PE 11-12	GYM	80	4	61
700	1	80	2	4	T RF	PE 11-12	GYM	80	4	61
700	2	80	1	5	T R	PE 11-12	GYM	80	5	45
700	2	80	2	5	T RF	PE 11-12	GYM	80	5	45
700	2	80	1	6	T R	PE 11-12	GYM	80	5	45
700	2	80	2	6	T RF	PE 11-12	GYM	80	5	45
700	3	80	1	12	M W	PE 11-12	GYM	80	9	70
700	3	80	2	12	M W F	PE 11-12	GYM	80	9	70
701	1	80	1	3	M W	PE 9-10	GYM	80	3	47
701	1	80	2	3	M W F	PE 9-10	GYM	80	3	47
701	2	80	1	5	M W F	PE 9-10	GYM	80	5	34
701	2	80	2	5	M W	PE 9-10	GYM	80	5	34
701	2	80	1	6	M W F	PE 9-10	GYM	80	5	34
701	2	80	2	6	M W	PE 9-10	GYM	80	5	34
701	3	80	1	7	T R	PE 9-10	GYM	80	6	34
701	3	80	2	7	T RF	PE 9-10	GYM	80	6	34
701	3	80	1	8	T R	PE 9-10	GYM	80	6	34
701	3	80	2	8	T RF	PE 9-10	GYM	80	6	34
701	4	80	1	9	T R	PE 9-10	GYM	80	7	26
701	4	80	2	9	T RF	PE 9-10	GYM	80	7	26
701	4	80	1	10	T R	PE 9-10	GYM	80	7	26
701	4	80	2	10	T RF	PE 9-10	GYM	80	7	26
701	5	90	1	11	M W	PE 9-10	GYM	80	8	58
701	5	90	2	11	M W F	PE 9-10	GYM	80	8	58
701	6	90	1	12	T RF	PE 9-10	GYM	80	9	66
701	6	90	2	12	T R	PE 9-10	GYM	80	9	66
702	1	80	1	1	T R	PE BOCES	GYM	80	1	22
702	1	80	2	1	T RF	PE BOCES	GYM	80	1	22
702	2	80	1	2	M W	PE BOCES	GYM	80	2	21
702	2	80	2	2	M W F	PE BOCES	GYM	80	2	21
750	1	35	1	4	F	DR ED TH	107	91	4	24
750	2	35	2	5	W	DR ED TH	108	91	5	17
750	2	35	2	6	W	DR ED TH	108	91	5	17
750	3	35	1	12	F	DR ED TH	107	91	9	23
751	1	35	2	1	W	DR ED TH	108	91	1	35
770	1	35	1	3	T RF	HEALTH 11	108	87	3	15
770	2	35	1	4	M W F	HEALTH 11	206	87	4	20
770	3	35	1	5	M W F	HEALTH 11	108	87	5	20
770	3	35	1	6	M W F	HEALTH 11	108	87	5	20
770	4	35	1	7	M W F	HEALTH 11	107	87	6	19
770	4	35	1	8	M W F	HEALTH 11	107	87	6	19
770	5	35	1	12	T RF	HEALTH 11	108	87	9	19
772	1	35	1	2	T RF	HEALTH 11	108	87	2	21
772	2	35	1	1	M W F	HEALTH 11	108	87	1	22
790	1	35	2	13	MTWRF	PSYCH	213	20	10	22
900	1	60	3	2	MTWRF	BOCES AM	BOC	99	2	56
900	1	60	3	3	MTWRF	BOCES AM	BOC	99	3	56
900	1	60	3	4	MTWRF	BOCES AM	BOC	99	4	56
900	1	60	3	5	MTWRF	BOCES AM	BOC	99	5	56
900	1	60	3	6	MTWRF	BOCES AM	BOC	99	5	56
901	1	60	3	8	MTWRF	BOCES PM	BOC	99	7	38
901	1	60	3	9	MTWRF	BOCES PM	BOC	99	7	38
901	1	60	3	10	MTWRF	BOCES PM	BOC	99	8	38
901	1	60	3	11	MTWRF	BOCES PM	BOC	99	8	38
901	1	60	3	12	MTWRF	BOCES PM	BOC	99	9	38
901	1	60	3	13	MTWRF	BOCES PM	BOC	99	10	38
999	1	300	3	5	MTWRF	LUNCH 5A	CAF	99	5	66
999	2	300	3	6	MTWRF	LUNCH 5B	CAF	99	5	67
999	3	300	3	7	MTWRF	LUNCH 6A	CAF	99	6	79
999	4	300	3	8	MTWRF	LUNCH 6B	CAF	99	6	80
999	5	300	3	9	MTWRF	LUNCH 7A	CAF	99	7	97
999	6	300	3	10	MTWRF	LUNCH 7B	CAF	99	7	98

9.6. STUDENT SCHEDULES

STUDENT NAME: ABARE ANN
ID : 21610
GRADE: 9

PERIOD	DAYS	COURSE	ROOM	SEM	TEACHER	CODE
3	M W F	PE 9-10	GYM	2	80	701
3	M W	PE 9-10	GYM	1	80	701
5	MTWRF	PER TYPING	119	2	60	462
5	MTWRF	BAS WOOD	135	1	67	505
6	MTWRF	ENG 9-2	220	3	10	6
7	MTWRF	LUNCH 7B	CAF	3	99	999
8	MTWRF	AFR-ASIA 2	213	3	23	105
9	MTWRF	MATH 9-3	217	3	37	205
10	MTWRF	SCI 9-2	103	3	49	308

STUDENT NAME: ALLEN CRYSTAL
ID : 21465
GRADE: 9

PERIOD	DAYS	COURSE	ROOM	SEM	TEACHER	CODE
3	M WR	READING 9	207	3	17	73
3	T F	WRIT LAB	228	3	3	52
4	MTWRF	SPAN 1	227	3	56	410
5	MTWRF	LUNCH 5B	CAF	3	99	999
6	MTWRF	SCI 9-3	103	3	84	309
7	MTWRF	MATH 9-3	217	3	37	205
8	MTWRF	AFR-ASIA 3	215	3	22	106
9	T R	PE 9-10	GYM	2	80	701
9	T RF	PE 9-10	GYM	1	80	701
10	MTWRF	ENG 9-3	123	3	4	7

STUDENT NAME: ALLEN MATTHEW
ID : 21465
GRADE: 9

PERIOD	DAYS	COURSE	ROOM	SEM	TEACHER	CODE
3	MTWRF	MATH CRS 1	224	3	32	204
4	MTWRF	SPAN 1	227	3	56	410
6	T RF	PE 9-10	GYM	2	80	701
6	T R	PE 9-10	GYM	1	80	701
7	MTWRF	LUNCH 7A	CAF	3	99	999
7	MTWRF	ER SCI R	108	3	45	307
8	MTWRF	ER SCI R	108	3	45	307
9	MTWRF	AFR-ASIA 2	215	3	23	105
10	MTWRF	ENG 9-2	220	3	3	6

9.7. COURSE CLASS LISTS

CLASS LIST

COURSE : SS IND STD
NUMBER : 100 SEC 1
PERIOD : 9
DAYS : MTWRF
SEM : 3
ROOM : 222
TEACHER : 24

ID#	GRADE	STUDENT NAME
19771	12	COLASURDO CHRISTOPH
19502	12	GALEK DAWN
18778	12	HAYNES JERRY
19566	12	MILLON DENISE
19854	12	PATCHETT THOMAS
19485	12	PETERSEN PAUL
19574	12	PIAZZA JOHN
19805	12	SHULTZ MARY
19713	12	SMITH SCOTT
19820	12	VALENCE CHARLES
19635	12	WOOD J MICHAEL

10. PROGRAM LISTINGS

10.1. ENTSTUDDATA

```
PROGRAM ENTSTUDDATA;

    (*************************************************************************)
    (*                                                                       *)
    (*  THIS PROGRAM ALLOWS THE USER TO CREATE, AND UPDATE ALL STUDENT       *)
    (*  INFORMATION.                                                         *)
    (*                                                                       *)
    (*  FILES USED                                                           *)
    (*    - ST:CRSREC(GRADE)                                                 *)
    (*                                                                       *)
    (*  UNITS USED                                                           *)
    (*    - UNIT ONE (IN SYSTEM.LIBRARY)                                     *)
    (*        THIS UNIT CONTAINS SOME SUBROUTINES USED BY OPTIONS ONE        *)
    (*        AND TWO OF THE PROGRAM.                                        *)
    (*                                                                       *)
    (*  OPTIONS                                                              *)
    (*                                                                       *)
    (*    1. OPEN A NEW FILE FOR THE FIRST TIME.                             *)
    (*                                                                       *)
    (*    2. USE AN EXISTING FILE TO ENTER AND CHANGE STUDENT INFORMATION    *)
    (*       1. RETRIEVE A STUDENT LIST FROM A FILE.                         *)
    (*       2. SAVE A STUDENT LIST TO A FILE.                               *)
    (*       3. ADD A STUDENT.                                               *)
    (*       4. DELETE A STUDENT.                                            *)
    (*       5. CHANGE STUDENT DATA.                                         *)
    (*       6. QUERY LIST BY STUDENT NAME.                                  *)
    (*       7. SEND LIST OF STUDENT COURSE REQUEST RECORDS TO PRINTER.      *)
    (*       8. SEND LIST OF GENERAL INFORMATION RECORDS TO PRINTER.         *)
    (*       9. RETURN TO MAIN PROGRAM.                                      *)
    (*                                                                       *)
    (*    3. USE AN EXISTING FILE TO ENTER AND CHANGE COURSE RECORDS.        *)
    (*       1. RETRIEVE A STUDENT LIST FROM A FILE.                         *)
    (*       2. SAVE A STUDENT LIST TO A FILE.                               *)
    (*       3. CHANGE OR ADD STUDENT COURSE REQUESTS.                       *)
    (*       4. QUERY LIST BY STUDENT NAME.                                  *)
    (*       5. SEND LIST OF STUDENT COURSE REQUEST TO PRINTER.              *)
    (*       6. RETURN TO MAIN MENU.                                         *)
    (*                                                                       *)
    (*    4. EXIT PROGRAM.                                                   *)
    (*                                                                       *)
    (*************************************************************************)


USES ONE;

PROCEDURE OPTION1;
    (* ALLOWS USER TO OPEN A FILE FOR THE FIRST TIME AND ENTER STUDENT
       INFORMATION *)

VAR FLAG  : INTEGER;                    (* REWRITE FILE IF 1 *)
    CH    : CHAR;                       (* - OR N ANSWER     *)

BEGIN
  PAGE(OUTPUT);
  HEAD1 := NIL;
  HEAD2 := NIL;
```

```
  FINDGRADE(GR);
  REPEAT
  NEW(STNODE);
  NEW(GENNODE);
  SETUP(GR);
  GOTOXY(0,1);
  WRITELN('WHEN YOU WISH TO EXIT ENTER STOP');
  GETDATA(GR,STNODE,GENNODE);
  UNTIL STNODE^.STUDNAM = 'STOP';
  PAGE(OUTPUT);
  GOTOXY(0,5);
  WRITELN('DO YOU WISH TO SAVE THIS TO A FILE?');
  WRITELN('ENTER ( Y OR N ) :       ');
  GOTOXY(20,6);
  READLN(CH);
  IF CH = 'Y'
    THEN
      BEGIN
        FLAG := 1;
        OPENFILE(GR,FLAG);
        FILEIT(HEAD1,HEAD2);
      END;
  PAGE(OUTPUT);
  GOTOXY(0,5);
  WRITELN('DO YOU WISH TO ENTER NEW INFORMATION FOR A DIFFERENT GRADE');
  WRITELN('ENTER ( Y OR N ) :       ');
  GOTOXY(20,6);
  READLN(CH);
  IF CH = 'Y'
    THEN OPTION1;
END;

PROCEDURE MENUOP2(VAR NUMBER : INTEGER);
   (* MENU FOR OPTION #2 *)

BEGIN
PAGE(OUTPUT);
GOTOXY(30,1);
WRITELN('STUDENT LIST OPTIONS');
GOTOXY(12,3);
WRITELN('1. RETRIEVE A STUDENT LIST FROM A FILE.');
GOTOXY(12,5);
WRITELN('2. SAVE A STUDENT LIST TO A FILE.');
GOTOXY(12,7);
WRITELN('3. ADD A STUDENT.');
GOTOXY(12,9);
WRITELN('4. DELETE A STUDENT.');
GOTOXY(12,11);
WRITELN('5. CHANGE STUDENT DATA');
GOTOXY(12,13);
WRITELN('6. QUERY LIST BY STUDENT NAME.');
GOTOXY(12,15);
WRITELN('7. SEND LIST OF STUDENT COURSE REQUEST RECORDS TO PRINTER.');
GOTOXY(12,17);
WRITELN('8. SEND LIST OF GENERAL INFORMATION RECORDS TO PRINTER.');
GOTOXY(12,19);
WRITELN('9. RETURN TO MAIN MENU.');
GOTOXY(15,21);
WRITELN('ENTER A NUMBER FROM 1 TO 9');
GOTOXY(15,22);
WRITELN('WHICH OPTION :         ');
GOTOXY(32,22);
READLN(NUMBER);

WHILE (NUMBER < 1 ) OR (NUMBER > 9) DO
  BEGIN
```

```
    GOTOXY(10,23);
    WRITELN('YOU MUST SELECT A NUMBER FROM 1 TO 9');
    GOTOXY(32,22);
    READLN(NUMBER);
  END
END;

PROCEDURE FETCHIT;
   (* GETS STUDENT COURSE RECORDS AND GENERAL INFORMATION RECORDS AND
      STORES THEM IN LINKED LISTS *)

VAR FLAG : INTEGER;                          (* RESET FILE IF 0 *)

BEGIN
  FLAG := 0;
  HEAD1 := NIL;
  HEAD2 := NIL;
  PAGE(OUTPUT);
  FINDGRADE(GR);
  OPENFILE(GR,FLAG);
  WHILE NOT EOF (STREC) DO
    BEGIN
      NEW(STNODE);
      STNODE^ := STREC^;
      IF HEAD1 = NIL
        THEN HEAD1 := STNODE
        ELSE INSERTST(HEAD1,STNODE);
      GET(STREC)
    END;
  CLOSE(STREC);
  WHILE NOT EOF (GENREC) DO
    BEGIN
      NEW(GENNODE);
      GENNODE^ := GENREC^;
      IF HEAD2 = NIL
        THEN HEAD2 := GENNODE
        ELSE INSERTGEN(HEAD2,GENNODE);
      GET(GENREC)
    END;
  CLOSE(GENREC);


END;

PROCEDURE ADDST;
   (* ADD A STUDENT RECORD TO THE LIST *)

BEGIN
  NEW(STNODE);
  NEW(GENNODE);
  PAGE(OUTPUT);
  FINDGRADE(GR);
  SETUP(GR);
  GETDATA(GR,STNODE,GENNODE)
END;

PROCEDURE FINDSTNODE(VAR LOOKAHEAD,CHASER,ITEM :POINT1; VAR TEST : BOOLEAN);
   (* LOCATES A STUDENT COURSE REQUEST RECORD *)

BEGIN

  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.STUDNAM > LOOKAHEAD^.STUDNAM) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK1
    END;
```

```
  IF (LOOKAHEAD <> NIL) AND (ITEM^.STUDNAM = LOOKAHEAD^.STUDNAM)
    THEN ITEM^ := LOOKAHEAD^;

  IF (LOOKAHEAD = NIL) OR (ITEM^.STUDNAM < LOOKAHEAD^.STUDNAM)
    THEN
      BEGIN
        GOTOXY(10,12);
        WRITELN(ITEM^.STUDNAM,' IS NOT IN THE STUDENT REQUEST LIST');
        TEST := TRUE;
        WAIT;
      END
END;

PROCEDURE FINDGENNODE(VAR LOOKAHEAD,CHASER,ITEM :POINT2; VAR TEST : BOOLEAN);
   (* LOCATES A GEN INFO RECORD IN THE LIST *)

BEGIN

  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.STUDNAM > LOOKAHEAD^.STUDNAM) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK2
    END;

  IF (LOOKAHEAD <> NIL) AND (ITEM^.STUDNAM = LOOKAHEAD^.STUDNAM)
    THEN ITEM^ := LOOKAHEAD^;

  IF (LOOKAHEAD = NIL) OR (ITEM^.STUDNAM < LOOKAHEAD^.STUDNAM)
    THEN
      BEGIN
        GOTOXY(10,12);
        WRITELN(ITEM^.STUDNAM,' IS NOT IN THE GENERAL INFORMATION LIST');
        TEST := TRUE;
        WAIT;
      END
END;

PROCEDURE WRINFO(FRONT1 :POINT1; FRONT2 : POINT2);
   (* WRITES STUDENT INFORMATION TO A SET UP SCREEN *)

BEGIN
  PAGE(OUTPUT);
  SETUP(GR);
  GOTOXY(17,6);
  WRITELN(FRONT1^.STUDNAM);
  GOTOXY(17,7);
  WRITELN(FRONT1^.STUDID);
  GOTOXY(17,8);
  WRITELN(FRONT1^.GRADE);
  GOTOXY(17,9);
  WRITELN(FRONT2^.STREET);
  GOTOXY(17,10);
  WRITELN(FRONT2^.TOWN);
  GOTOXY(17,11);
  WRITELN(FRONT2^.ZIP);
  GOTOXY(17,12);
  WRITELN(FRONT2^.BIRTH);
  GOTOXY(17,13);
  WRITELN(FRONT2^.SEX);
  GOTOXY(17,14);
  WRITELN(FRONT2^.HR);
  GOTOXY(17,15);
  WRITELN(FRONT2^.PHONE);

END;
```

```
PROCEDURE DELETEST;
   (* DELETES A STUDENT REQUEST RECORD AND STUDENT INFO RECORD
      FROM THE LIST *)

VAR  FLAG1,                                  (* CRS REQ NODE FOUND *)
     FLAG2      : BOOLEAN;                   (* GEN INFO NODE FOUND *)
     LOOK1,                                  (* CRS REQ - LOOKAHEAD *)
     CHAS1      : POINT1;                    (*         - CHASER    *)
     LOOK2,                                  (* GEN INFO - LOOKAHEAD *)
     CHAS2      : POINT2;                    (*          - CHASER    *)
     ANS        : CHAR;                      (* Y OR N - ANSWER      *)

BEGIN
  NEW(STNODE);
  NEW(GENNODE);
  PAGE(OUTPUT);
  FLAG1 := FALSE;
  FLAG2 := FALSE;
  LOOK1 := HEAD1;
  LOOK2 := HEAD2;
  CHAS1 := HEAD1;
  CHAS2 := HEAD2;
  GOTOXY(0,1);
  WRITELN('WHAT STUDENT WOULD YOU LIKE TO DELETE?');
  WRITELN('ENTER THE LAST NAME FIRST');
  WRITELN('STUDENT NAME :                          ');
  GOTOXY(15,3);
  READLN(STNODE^.STUDNAM);
  GENNODE^.STUDNAM := STNODE^.STUDNAM;
  FINDSTNODE(LOOK1,CHAS1,STNODE,FLAG1);
  FINDGENNODE(LOOK2,CHAS2,GENNODE,FLAG2);
  IF (FLAG1 = FALSE) AND (FLAG2 = FALSE)
    THEN
      BEGIN                                  (* STUDENT IS IN LIST *)
        WRINFO(STNODE,GENNODE);
        GOTOXY(10,23);
        WRITE('DO YOU WANT TO MAKE THE CHANGE IN THIS RECORD? (Y OR N) ');
        READLN(ANS);
        IF ANS = 'Y'
          THEN
            BEGIN                            (* DELETE STUDENT *)
              IF HEAD1^.STUDNAM = STNODE^.STUDNAM
                THEN HEAD1 := LOOK1^.LINK1
                ELSE
                  BEGIN
                    CHAS1^.LINK1 := LOOK1^.LINK1;
                    LOOK1^.LINK1 := NIL;
                  END;
               IF HEAD2^.STUDNAM = GENNODE^.STUDNAM
                 THEN HEAD2 := LOOK2^.LINK2
                 ELSE
                   BEGIN
                     CHAS2^.LINK2 := LOOK2^.LINK2;
                     LOOK2^.LINK2 := NIL;
                   END;
              PAGE(OUTPUT);
              GOTOXY(10,5);
              WRITELN('STUDENT ');
              WRITE( STNODE^.STUDNAM);
              WRITELN(' HAS BEEN DELETED');
              STNODE := NIL;
              GENNODE := NIL;
              WAIT;
            END;
      END;
END;
```

```
PROCEDURE CHANGEDATA;
   (* ALLOWS USER TO MAKE CHANGES IS STUDENT DATA *)

VAR  FLAG1,                                  (* CRS REQ NODE FOUND *)
     FLAG2      : BOOLEAN;                   (* GEN INFO NODE FOUND *)
     SSTNODE,                                (* CRS REQ - NEW NODE  *)
     LOOK1,                                  (*         - LOOKAHEAD *)
     CHAS1      : POINT1;                    (*         - CHASER    *)
     GGENNODE,                               (* GEN INFO - NEW NODE *)
     LOOK2,                                  (*          - LOOKAHEAD*)
     CHAS2      : POINT2;                    (*          - CHASER   *)
     ANS        : CHAR;                      (* Y OR N - ANSWER     *)

BEGIN
  NEW(STNODE);
  NEW(GENNODE);
  PAGE(OUTPUT);
  FLAG1 := FALSE;
  FLAG2 := FALSE;
  LOOK1 := HEAD1;
  LOOK2 := HEAD2;
  CHAS1 := HEAD1;
  CHAS2 := HEAD2;
  GOTOXY(0,1);
  WRITELN('WHAT STUDENT INFORMATION WOULD YOU LIKE TO CHANGE?');
  WRITELN('ENTER THE LAST NAME FIRST');
  WRITELN('STUDENT NAME :                          ');
  GOTOXY(15,3);
  READLN(STNODE^.STUDNAM);
  GENNODE^.STUDNAM := STNODE^.STUDNAM;
  FINDSTNODE(LOOK1,CHAS1,STNODE,FLAG1);
  FINDGENNODE(LOOK2,CHAS2,GENNODE,FLAG2);
  IF (FLAG1 = FALSE) AND (FLAG2 = FALSE)
    THEN
      BEGIN
        WRINFO(STNODE,GENNODE);
        GOTOXY(45,2);
        WRITELN('TO MAKE CHANGES');
        GOTOXY(45,3);
        WRITELN('REENTER ALL INFORMATION');
        NEW(SSTNODE);
        NEW(GGENNODE);
        GOTOXY(40,6);
        READLN(SSTNODE^.STUDNAM);
        GOTOXY(40,7);
        READLN(SSTNODE^.STUDID);
        GOTOXY(40,8);
        READLN(SSTNODE^.GRADE);
        GOTOXY(40,9);
        READLN(GGENNODE^.STREET);
        GOTOXY(40,10);
        READLN(GGENNODE^.TOWN);
        GOTOXY(40,11);
        READLN(GGENNODE^.ZIP);
        GOTOXY(40,12);
        READLN(GGENNODE^.BIRTH);
        GOTOXY(40,13);
        READLN(GGENNODE^.SEX);
        GOTOXY(40,14);
        READLN(GGENNODE^.HR);
        GOTOXY(40,15);
        READLN(GGENNODE^.PHONE);
        GOTOXY(10,23);
        WRITE('DO YOU WANT TO MAKE THE CHANGE IN THIS RECORD? (Y OR N) ');
        READLN(ANS);
        IF ANS = 'Y'
```

```
            THEN
              BEGIN
                 GGENNODE^.STUDNAM := SSTNODE^.STUDNAM;
                 GGENNODE^.STUDID  := SSTNODE^.STUDID;
                 SSTNODE^.LINK1 := NIL;
                 GGENNODE^.LINK2 := NIL;
                 IF HEAD1^.STUDNAM = STNODE^.STUDNAM
                   THEN HEAD1 := LOOK1^.LINK1
                   ELSE
                     BEGIN
                       CHAS1^.LINK1 := LOOK1^.LINK1;
                       LOOK1^.LINK1 := NIL;
                     END;
                  INSERTST(HEAD1,SSTNODE);
                  IF HEAD2^.STUDNAM = GENNODE^.STUDNAM
                    THEN HEAD2 := LOOK2^.LINK2
                    ELSE
                      BEGIN
                        CHAS2^.LINK2 := LOOK2^.LINK2;
                        LOOK2^.LINK2 := NIL;
                      END;
                  INSERTGEN(HEAD2,GGENNODE);
                  WRINFO(SSTNODE,GGENNODE);
                  GOTOXY(10,2);
                  WRITELN('THE CHANGE HAS BEEN MADE');
                  WAIT;
               END;
        END;


END;

PROCEDURE QUERY;
    (* DISPLAYS INFORMATION FOR ANY STUDENT REQUESTED *)

VAR  FLAG1,                                  (* CRS REQ NODE FOUND    *)
     FLAG2      : BOOLEAN;                   (* GEN INFO NODE FOUND   *)
     LOOK1,                                  (* CRS REQ - LOOKAHEAD   *)
     CHAS1      : POINT1;                    (*           - CHASER    *)
     LOOK2,                                  (* GEN INFO - LOOKAHEAD  *)
     CHAS2      : POINT2;                    (*            - CHASER   *)

BEGIN
  NEW(STNODE);
  NEW(GENNODE);
  PAGE(OUTPUT);
  FLAG1 := FALSE;
  FLAG2 := FALSE;
  LOOK1 := HEAD1;
  LOOK2 := HEAD2;
  CHAS1 := HEAD1;
  CHAS2 := HEAD2;
  GOTOXY(0,1);
  WRITELN('WHAT STUDENT WOULD YOU LIKE TO FIND?');
  WRITELN('ENTER THE LAST NAME FIRST');
  WRITELN( STUDENT NAME :                    ');
  GOTOXY(15,3);
  READLN(STNODE^.STUDNAM);
  GENNODE^.STUDNAM := STNODE^.STUDNAM;
  FINDSTNODE(LOOK1,CHAS1,STNODE,FLAG1);
  FINDGENNODE(LOOK2,CHAS2,GENNODE,FLAG2);
  IF (FLAG1 = FALSE) AND (FLAG2 = FALSE)
    THEN
```

```
      BEGIN
        WRINFO(STNODE,GENNODE);
        WAIT;
      END;

END;

PROCEDURE SAVEIT;
   (* SAVES STUDENT REQEST LIST AND GEN INFO LIST TO FILES *)

VAR FLAG    : INTEGER;                    (* REWRITE FILE IF 1 *)

BEGIN
  FLAG := 1;
  OPENFILE(GR,FLAG);
  FILEIT(HEAD1,HEAD2);
  PAGE(OUTPUT);
  GOTOXY(10,5);
  WRITELN('FILE SAVED ON DISK');
  WAIT;
END;

PROCEDURE OPTION2;
   (* SELECTIONS FOR OPTION #2 *)

VAR OPT    :  INTEGER;                    (* OPTION NUMBER *)

BEGIN

REPEAT
  PAGE(OUTPUT);
  MENUOP2(OPT);
  CASE OPT OF
    1   : FETCHIT;
    2   : SAVEIT;
    3   : ADDST;
    4   : DELETEST;
    5   : CHANGEDATA;
    6   : QUERY;
    7   : PRINT1LIST(HEAD1);
    8   : PRINT2LIST(HEAD2);
    9   : PAGE(OUTPUT);
  END;
UNTIL OPT  = 9;

END;

PROCEDURE FETCHCRS;
   (* LOADS STUDENT REQUEST RECORDS INTO LINKED LIST *)

BEGIN
  HEAD1 := NIL;
  PAGE(OUTPUT);
  FINDGRADE(GR);
  RESET(STREC,CONCAT('ST:STREC',GR));
  WHILE NOT EOF (STREC) DO
    BEGIN
      NEW(STNODE);
      STNODE^ := STREC^;
      IF HEAD1 = NIL
        THEN HEAD1 := STNODE
        ELSE INSERTST(HEAD1,STNODE);
      GET(STREC)
    END;
  CLOSE(STREC);
END;
```

```
PROCEDURE MENUOP3(VAR NUMBER : INTEGER);
   (* DISPLAYS MENU FOR OPTION #3 *)

BEGIN
PAGE(OUTPUT);
GOTOXY(30,1);
WRITELN('STUDENT LIST OPTIONS');
GOTOXY(12,3);
WRITELN('1. RETRIEVE A STUDENT LIST FROM A FILE.');
GOTOXY(12,5);
WRITELN('2. SAVE A STUDENT LIST TO A FILE.');
GOTOXY(12,7);
WRITELN('3. CHANGE OR ADD STUDENT COURSE REQUESTS.');
GOTOXY(12,9);
WRITELN('4. QUERY LIST BY STUDENT NAME.');
GOTOXY(12,11);
WRITELN('5. SEND LIST OF STUDENT COURSE REQUEST RECORDS TO PRINTER');
GOTOXY(12,13);
WRITELN('6. RETURN TO MAIN MENU.');
GOTOXY(15,21);
WRITELN('ENTER A NUMBER FROM 1 TO 6');
GOTOXY(15,22);
WRITELN('WHICH OPTION :          ');
GOTOXY(32,22);
READLN(NUMBER);

WHILE (NUMBER < 1 ) OR (NUMBER > 6) DO
  BEGIN
    GOTOXY(10,23);
    WRITELN('YOU MUST SELECT A NUMBER FROM 1 TO 6');
    GOTOXY(32,22);
    READLN(NUMBER);
  END
END;


PROCEDURE FILECRS(FRONT : POINT1);
   (* STORES LIST OF STUDENT REQUEST RECORDS TO A FILE *)

BEGIN

IF FRONT = NIL
  THEN
    BEGIN
      GOTOXY(10,12);
      WRITELN('STUDENT REQUEST LIST IS EMPTY');
    END
  ELSE
    BEGIN
      WHILE FRONT <> NIL DO
        BEGIN
          STREC^ := FRONT^;
          STREC^.LINK1 := NIL;
          PUT(STREC);
          FRONT := FRONT^.LINK1
        END;
    END;
CLOSE(STREC,LOCK);
END;

PROCEDURE SAVECRS;
   (* OPENS FILE TO SAVE COURSE LIST *)

BEGIN
  REWRITE(STREC,CONCAT('ST:STREC',GR));
  FILECRS(HEAD1);
```

```
  PAGE(OUTPUT);
  GOTOXY(10,5);
  WRITELN('FILE SAVED ON DISK');
  WAIT;
END;

PROCEDURE SETCRS;
   (* SETS UP SCREEN FOR ENTRY OF STUDENTS COURSES *)

BEGIN
PAGE(OUTPUT);
GOTOXY(15,3);
WRITELN('STUDENT COURSE INFORMATION');
GOTOXY(0,6);
WRITELN('STUDENT NAME :                          ');
GOTOXY(0,7);
WRITELN('STUDENT ID   :             ');
GOTOXY(0,8);
WRITELN('GRADE        :        ');
GOTOXY(0,9);
WRITELN('COURSE 1     :        ');
GOTOXY(0,10);
WRITELN('COURSE 2     :        ');
GOTOXY(0,11);
WRITELN('COURSE 3     :        ');
GOTOXY(0,12);
WRITELN('COURSE 4     :        ');
GOTOXY(0,13);
WRITELN('COURSE 5     :        ');
GOTOXY(0,14);
WRITELN('COURSE 6     :        ');
GOTOXY(0,15);
WRITELN('COURSE 7     :        ');
GOTOXY(0,16);
WRITELN('COURSE 8     :        ');
GOTOXY(0,17);
WRITELN('COURSE 9     :        ');
GOTOXY(0,18);
WRITELN('COURSE 10    :        ');
END;

PROCEDURE WRCRS(FRONT : POINT1);
   (* WRITES STUDENTS COURSES TO A SET UP SCREEN *)

BEGIN
PAGE(OUTPUT);
SETCRS;
GOTOXY(17,6);
WRITELN(FRONT^.STUDNAM);
GOTOXY(17,7);
WRITELN(FRONT^.STUDID);
GOTOXY(17,8);
WRITELN(FRONT^.GRADE);
GOTOXY(17,9);
WRITELN(FRONT^.CRSREQ[1]);
GOTOXY(17,10);
WRITELN(FRONT^.CRSREQ[2]);
GOTOXY(17,11);
WRITELN(FRONT^.CRSREQ[3]);
GOTOXY(17,12);
WRITELN(FRONT^.CRSREQ[4]);
GOTOXY(17,13);
WRITELN(FRONT^.CRSREQ[5]);
GOTOXY(17,14);
WRITELN(FRONT^.CRSREQ[6]);
GOTOXY(17,15);
```

```
WRITELN(FRONT^.CRSREQ[7]);
GOTOXY(17,16);
WRITELN(FRONT^.CRSREQ[8]);
GOTOXY(17,17);
WRITELN(FRONT^.CRSREQ[9]);
GOTOXY(17,18);
WRITELN(FRONT^.CRSREQ[10]);
END;

PROCEDURE LOCATECRS(VAR FL :BOOLEAN; VAR L1,C1 : POINT1);
   (* WRITES A STUDENT COURSE RECORD TO THE SCREEN *)

BEGIN
NEW(STNODE);
FL := FALSE;
L1 := HEAD1;
C1 := HEAD1;
GOTOXY(0,2);
WRITELN('ENTER THE LAST NAME FIRST');
WRITELN('STUDENT NAME :                    ');
GOTOXY(15,3);
READLN(STNODE^.STUDNAM);
FINDSTNODE(L1,C1,STNODE,FL);
IF FL = FALSE
  THEN WRCRS(STNODE);
END;

PROCEDURE QUERYCRS;
   (* CALLS LOCATCRS TO FIND A STUDENTS COURSE REQUEST RECORD *)

VAR FLAG1      : BOOLEAN;                (* CRS REQ - NODE FOUND
    LOOK1,                               (*         - LOOKAHEAD
    CHAS1      : POINT1;                 (*         - CHASER

BEGIN
PAGE(OUTPUT);
GOTOXY(0,1);
WRITELN('WHAT STUDENT WOULD YOU LIKE TO FIND? ');
LOCATECRS(FLAG1,LOOK1,CHAS1);
WAIT;
END;

PROCEDURE SORTARRAY(VAR LIST :COURREQLIST);
   (*  SORTS THE COURSE ARRAY IN DECENDING ORDER *)

VAR TEMP       : COURSENUMBER;           (* TEMP CRS NUMBER *)
    I, J       : INTEGER;                (* LOOP COUNTER    *)

BEGIN
FOR I := 1 TO (MAX-1) DO
  FOR J := (I + 1) TO MAX DO
    IF LIST[I] < LIST[J]
      THEN
        BEGIN
          TEMP := LIST[I];
          LIST[I] := LIST[J];
          LIST[J] := TEMP
        END;
END;

PROCEDURE NEWCRS(VAR NODE1 : POINT1);
   (* ALLOWS USER TO INPUT COURSES AND CALLS THE SORT ROUTINE *)
```

```
BEGIN
GOTOXY(40,9);
READLN(NODE1^.CRSREQ[1]);
GOTOXY(40,10);
READLN(NODE1^.CRSREQ[2]);
GOTOXY(40,11);
READLN(NODE1^.CRSREQ[3]);
GOTOXY(40,12);
READLN(NODE1^.CRSREQ[4]);
GOTOXY(40,13);
READLN(NODE1^.CRSREQ[5]);
GOTOXY(40,14);
READLN(NODE1^.CRSREQ[6]);
GOTOXY(40,15);
READLN(NODE1^.CRSREQ[7]);
GOTOXY(40,16);
READLN(NODE1^.CRSREQ[8]);
GOTOXY(40,17);
READLN(NODE1^.CRSREQ[9]);
GOTOXY(40,18);
READLN(NODE1^.CRSREQ[10]);
SORTARRAY(NODE1^.CRSREQ);
END;

PROCEDURE CHANGECRS;
   (* ALLOWS USER TO CHANGE OR ADD COURSE TO THE STUDENT REQEST RECORD *)

VAR  FLAG1      : BOOLEAN;                (* ST REQ NODE  - NODE FOUND *)
     SSTNODE,                             (*              - NEW NODE   *)
     LOOK1,                               (*              - LOOKAHEAD  *)
     CHAS1      : POINT1;                 (*              - CHASER     *)
     ANS        : CHAR;                   (* Y OR N - ANSWER           *)

BEGIN
PAGE(OUTPUT);
GOTOXY(0,1);
WRITELN('CHANGE OR ADD COURSES FOR WHAT STUDENTS');
LOCATECRS(FLAG1,LOOK1,CHAS1);
IF FLAG1 = FALSE
  THEN
    BEGIN
      NEW(SSTNODE);
      GOTOXY(45,2);
      WRITELN('TO MAKE CHANGES');
      GOTOXY(45,3);
      WRITELN('REENTER ALL COURSE NUMBERS');
      NEWCRS(SSTNODE);
      GOTOXY(10,23);
      WRITE('DO YOU WANT TO MAKE THE CHANGE IN THIS RECORD? (Y OR N) ');
      READLN(ANS);
      IF ANS = 'Y'
        THEN
          BEGIN
           SSTNODE^.LINK1 := NIL;
           IF HEAD1^.STUDNAM = STNODE^.STUDNAM
             THEN HEAD1 := LOOK1^.LINK1
             ELSE
               BEGIN
                CHAS1^.LINK1 := LOOK1^.LINK1;
                LOOK1^.LINK1 := NIL;
               END;
           SSTNODE^.STUDNAM := STNODE^.STUDNAM;
           SSTNODE^.STUDID  := STNODE^.STUDID;
           SSTNODE^.GRADE   := STNODE^.GRADE;
           IF HEAD1 = NIL
             THEN HEAD1 := SSTNODE
             ELSE INSERT1T(HEAD1,SSTNODE);
```

```
            WRCRS(SSTNODE);
            GOTOXY(10,2);
            WRITELN('THE CHANGE HAS BEEN MADE.');
            WAIT;
           END
         END
    END;


PROCEDURE OPTION3;
   (* SELECTION FOR OPTION #3 *)

VAR  OP3    : INTEGER;                    (* OPTION NUMBER *)

BEGIN
REPEAT
  PAGE(OUTPUT);
  MENUOP3(OP3);
  CASE  OP3  OF
    1  : FETCHCRS;
    2  : SAVECRS;
    3  : CHANGECRS;
    4  : QUERYCRS;
    5  : PRINT1LIST(HEAD1);
    6  : PAGE(OUTPUT);
  END;
UNTIL OP3  = 6

END;

PROCEDURE SELECTOPTION(VAR OPTION : INTEGER);
   (* MENU FOR MAIN PROGRAM *)

BEGIN
  PAGE(OUTPUT);
  GOTOXY(38,1);
  WRITELN('OPTIONS');
  GOTOXY(12,3);
  WRITELN('1. OPEN NEW FILE FOR THE FIRST TIME.');
  GOTOXY(12,5);
  WRITELN('2. USE EXISTING FILE TO ENTER AND CHANGE STUDENT INFORMATION.');
  GOTOXY(12,7);
  WRITELN('3. USE EXISTING FILE TO ENTER AND CHANGE COURSE REQUESTS.');
  GOTOXY(12,9);
  WRITELN('4. EXIT PROGRAM.');
  GOTOXY(15,21);
  WRITELN('ENTER A NUMBER FROM 1 TO 4');
  GOTOXY(15,22);
  WRITELN('WHICH OPTION :          ');
  GOTOXY(32,22);
  READLN(OPTION);

  WHILE (OPTION < 1) OR (OPTION > 4) DO
    BEGIN
      GOTOXY(10,23);
      WRITELN('YOU MUST SELECT A NUMBER FROM 1 TO 4');
      GOTOXY(32,22);
      READLN(OPTION);
    END
END;
```

```
BEGIN                                                    (* ENTSTUDDATA *)

  PAGE(OUTPUT);
  HEAD1 := NIL;
  HEAD2 := NIL;
  PAGE(OUTPUT);

  REPEAT
  SELECTOPTION(NUMBER);
  CASE NUMBER OF
    1     : OPTION1;
    2     : OPTION2;
    3     : OPTION3;
    4     : PAGE(OUTPUT);
  END;
  UNTIL NUMBER = 4;

END.                                                     (* ENTSTUDDATA *)
```

10.2. UNIT ONE

```
(*$S+*)                                    (* ALLOW SWAPPING *)

UNIT ONE;  INTRINSIC  CODE 25 DATA 26;

   (*************************************************************)
   (*                                                           *)
   (*    THIS UNIT CONTAINS SUBROUTINES USED BY OPTIONS ONE AND TWO    *)
   (*    OF PROGRAM 'ENTERSTUDDATA'                             *)
   (*                                                           *)
   (*************************************************************)


INTERFACE

CONST
  MAX = 10;                                (* MAX NUMBER OF COURSES *)

TYPE
   POINT1             = ^STUDENT;          (* TO STUDENT RECORD *)
   POINT2             = ^GENINFO;          (* TO GENERAL INFORMATION RECORD
   YEAR               = 7..12;
   COURSENUMBER       = 0..999;
   COURREQLIST        = ARRAY[1..MAX] OF COURSENUMBER;
   STUDENT            =
     RECORD
       STUDNAM        : STRING[20];        (* STUDENT NAME *)
       STUDID         : INTEGER;           (* STUDENT ID NUMBER *)
       GRADE          : YEAR;              (* GRADE IN SCHOOL *)
       CRSREQ         : COURREQLIST;       (* ARRAY OF STUDENT COURSES *)
       LINK1          : POINT1;            (* POINTER TO NEXT RECORD *)
     END;
   GENINFO            =
     RECORD
       STUDNAM        : STRING[20];        (* STUDENT NAME *)
       STUDID         : INTEGER;           (* STUDENT ID NUMBER *)
       STREET         : STRING[15];        (* STREET NAME *)
       TOWN           : STRING[10];            (* TOWN NAME *)
       ZIP            : INTEGER;           (* ZIP CODE *)
       PHONE          : STRING[8];         (* PHONE NUMBER *)
       HR             : STRING[3];         (* HOME ROOM NUMBER *)
       BIRTH          : STRING[8];         (* BIRTHDATE *)
       SEX            : CHAR;              (* SEX *)
       LINK2          : POINT2;            (* POINTER TO NEXT RECORD *)
     END;

 VAR
   STREET             :STRING[15];
   TOWN               :STRING[10];
   GR,
   TITLE1,                                 (* STUDENT COURSE FILE *)
   TITLE2             : STRING;            (* GENERAL INFORMATION FILE *)
   PHONE,
   BIRTH              : STRING[8];
   HR                 : STRING[3];
   SEX                : CHAR;
   ZIP,
   NUMBER             : INTEGER;           (* MENU SELECTION *)
   HEAD1,                                  (* HEAD OF COURSE LIST *)
```

```
   STNODE              : POINT1;          (* STUDENT COURSE RECORD *)
   HEAD2,                                 (* HEAD OF GEN INFO LIST *)
   GENNODE             : POINT2;          (* GEN INFO RECORD *)
   STREC               : FILE OF STUDENT; (* FILE OF STUDENT RECORDS *)
   GENREC              : FILE OF GENINFO; (* FILE OF GEN INFO RECORDS *)
   LENGT,                                 (* COUNTER *)
   INDEX               : 0..MAX;          (* ARRAY INDEX *)
   OUTDEVICE           : TEXT;            (* PRINTER *)

PROCEDURE WAIT;
PROCEDURE SETUP(GRAD : STRING);
PROCEDURE INSERTST(VAR FRONT, ITEM :POINT1);
PROCEDURE INSERTGEN(VAR FRONT,ITEM :POINT2);
PROCEDURE GETDATA(GRAD :STRING; VAR NODE1  :POINT1; VAR NODE2 : POINT2);
PROCEDURE OPENFILE(GRAD : STRING; VAR FLAG : INTEGER);
PROCEDURE FILEIT(FRONT1 : POINT1; FRONT2 : POINT2);
PROCEDURE PRINT1LIST(FRONT : POINT1);
PROCEDURE PRINT2LIST(FRONT : POINT2);
PROCEDURE FINDGRADE(VAR GRAD : STRING);

IMPLEMENTATION

VAR
  CH    : CHAR;                          (* ANY CHARACTER *)

PROCEDURE WAIT;
   (* LETS THE USER DECIDE WHEN TO GO TO NEXT SCREEN*)

BEGIN
  GOTOXY(10,23);
  WRITELN('[ PRESS ANY KEY TO CONTINUE ]');
  READ(CH)
END;


PROCEDURE SETUP;
   (* DISPLAY SCREEN FOR INPUT OF STUDENT DATA *)

BEGIN
  PAGE(OUTPUT);
  GOTOXY(15,3);
  WRITE('STUDENT INFORMATION  GRADE ');
  WRITELN(GRAD);
  GOTOXY(0,6);
  WRITELN('STUDENT NAME :                              );
  GOTOXY(0,7);
  WRITELN('STUDENT ID   :           ');
  GOTOXY(0,8);
  WRITELN('GRADE        :           ');
  GOTOXY(0,9);
  WRITELN('STREET       :                              ');
  GOTOXY(0,10);
  WRITELN('TOWN         :                              ');
  GOTOXY(0,11);
  WRITELN('ZIP          :             ');
  GOTOXY(0,12);
  WRITELN('BIRTHDATE    :               ');
  GOTOXY(0,13);
  WRITELN('SEX (M,F)    :       ');
  GOTOXY(0,14);
  WRITELN('HOMEROOM     :        ');
  GOTOXY(0,15);
  WRITELN('PHONE        :              );

END;
```

```
PROCEDURE INSERTST;
   (* INSERTS A STUDENT COURSE RECORD INTO LINKED LIST *)

VAR  LOOKAHEAD,                          (* POINTER TO FIND PLACE *)
     CHASER        : POINT1;             (* POINTER FOR INSERTION *)

BEGIN
  LOOKAHEAD := FRONT;
  CHASER    := FRONT;
  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.STUDNAM > LOOKAHEAD^.STUDNAM) DO
    BEGIN                                (* TRAVERSE LIST TO INSERT POINT *)
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK1
    END;

  IF LOOKAHEAD <> NIL
    THEN                                 (* NOT AT END OF LIST *)
    IF LOOKAHEAD <> FRONT
      THEN
        BEGIN                            (* INSERT IN MIDDLE OF LIST *)
          ITEM^.LINK1 := LOOKAHEAD;
          CHASER^.LINK1 := ITEM;
        END
      ELSE
        BEGIN                            (* INSERT AT FRONT *)
          ITEM^.LINK1 := FRONT;
          FRONT := ITEM;
        END
    ELSE
      CHASER^.LINK1 := ITEM;


END;

PROCEDURE INSERTGEN;
   (* INSERTS GEN INFO RECORD INTO LINKED LIST *)

VAR  LOOKAHEAD,                          (* GEN INFO - LOOKAHEAD *)
     CHASER        : POINT2;             (*          - CHASER    -

BEGIN
  LOOKAHEAD := FRONT;
  CHASER    := FRONT;

  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.STUDNAM > LOOKAHEAD^.STUDNAM) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK2
    END;

  IF LOOKAHEAD <> NIL
    THEN
    IF LOOKAHEAD <> FRONT
      THEN
        BEGIN
          ITEM^.LINK2 := LOOKAHEAD;
          CHASER^.LINK2 := ITEM;
        END
      ELSE
        BEGIN
          ITEM^.LINK2 := FRONT;
          FRONT := ITEM;
        END
    ELSE
      CHASER^.LINK2 := ITEM;
```

```
END;


PROCEDURE GETDATA;
   (* READS INFORMATION INPUTED ON SETUP SCREEN *)

BEGIN

  WITH NODE1^ DO                              (* GET NAME *)
    BEGIN
      GOTOXY(15,6);
      READLN(STUDNAM);
    END;

  IF (NODE1^.STUDNAM <> 'STOP')  THEN    (* IF 'STOP' THEN EXIT *)
  BEGIN
  WITH NODE1^ DO
    BEGIN
      GOTOXY(15,7);
      READLN(STUDID);
      GOTOXY(15,8);
      READLN(GRADE);
    END;
  NODE2^.STUDNAM := NODE1^.STUDNAM;
  NODE2^.STUDID  := NODE1^.STUDID;

  WITH NODE2^ DO
    BEGIN
      GOTOXY(15,9);
      READLN(STREET);
      GOTOXY(15,10);
      READLN(TOWN);
      GOTOXY(15,11);
      READLN(ZIP);
      GOTOXY(15,12);
      READLN(BIRTH);
      GOTOXY(15,13);
      READLN(SEX);
      GOTOXY(15,14);
      READLN(HR);
      GOTOXY(15,15);
      READLN(PHONE);
    END;

  NODE1^.LINK1 := NIL;
  NODE2^.LINK2 := NIL;
  FOR LENGT := 1 TO MAX DO              (* INITIALIZE COURSE LIST *)
    NODE1^.CRSREQ[LENGT] := 0;
  IF HEAD1 = NIL                         (* INSERT AT HEAD OF LIST *)
    THEN HEAD1 := NODE1
    ELSE INSERTST(HEAD1,NODE1);          (* INSERT IN THE LIST *)
  IF HEAD2 = NIL
    THEN HEAD2 := NODE2
    ELSE INSERTGEN(HEAD2,NODE2);
  END;

END;

PROCEDURE OPENFILE;
   (* OPEN FILES FOR STUDENT COURSE RECORDS AND GEN INFO RECORDS *)

BEGIN
  TITLE1  := CONCAT('ST:STREC',GRAD);
  TITLE2  := CONCAT('ST:GINFO',GRAD);
```

```
  IF FLAG = 1
    THEN
      BEGIN
        REWRITE(STREC,TITLE1);
        REWRITE(GENREC,TITLE2);
      END
    ELSE
      BEGIN
        RESET(STREC,TITLE1);
        RESET(GENREC,TITLE2);
      END

END;

PROCEDURE FILEIT;
   (* SAVE LINKED LISTS TO FILE *)

BEGIN
  IF FRONT1 = NIL                          (* STUDENT COURSE REQUEST LIST *)
    THEN
      BEGIN
        GOTOXY(10,12);
        WRITELN('STUDENT REQUEST LIST IS EMPTY');
      END
    ELSE
      BEGIN
        WHILE FRONT1 <> NIL DO
          BEGIN
            STREC^ := FRONT1^;
            STREC^.LINK1 := NIL;
            PUT(STREC);
            FRONT1 := FRONT1^.LINK1;
          END;
        CLOSE(STREC,LOCK);
      END;

  IF FRONT2 = NIL                          (* GEN INFO LIST *)
    THEN
      BEGIN
        GOTOXY(10,14);
        WRITELN('GENERAL INFORMATION LIST IS EMPTY');
      END
    ELSE
      BEGIN
        WHILE FRONT2 <> NIL DO
          BEGIN
            GENREC^ := FRONT2^;
            GENREC^.LINK2 := NIL;
            PUT(GENREC);
            FRONT2 := FRONT2^.LINK2;
          END;
        CLOSE(GENREC,LOCK);
      END


END;

PROCEDURE PRINT1LIST;
   (* PRINTS LIST OF STUDENT COURSE REQUESTS *)


BEGIN
  REWRITE(OUTDEVICE,'PRINTER:');
  PAGE(OUTDEVICE);
  IF FRONT = NIL
    THEN
```

```
      BEGIN
        WRITELN(OUTDEVICE,'                STUDENT LIST IS EMPTY');
      END
    ELSE
      BEGIN
        WRITELN(OUTDEVICE,'                STUDENT COURSE REQUEST LIST );
        WRITELN(OUTDEVICE);
        WRITELN(OUTDEVICE,'   STUDENT NAME        GR  ST.NO.');
        WRITELN(OUTDEVICE);
        WHILE FRONT <> NIL DO
          BEGIN
            WITH FRONT^ DO
              BEGIN
              WRITE(OUTDEVICE,STUDNAM,'  ');
              FOR NUMBER := 1 TO (20 - LENGTH(STUDNAM)) DO
                 WRITE(OUTDEVICE,' ');
              WRITE(OUTDEVICE,GRADE : 2,    ');
              WRITELN(OUTDEVICE,STUDID);
              WRITE(OUTDEVICE,'          ');
              FOR INDEX := 1 TO MAX DO
                BEGIN
                   IF CRSREQ[INDEX] <> 0
                     THEN WRITE(OUTDEVICE,CRSREQ[INDEX],' ');
                END
              END;
              WRITELN(OUTDEVICE);WRITELN(OUTDEVICE);
            FRONT := FRONT^.LINK1;
          END
      END;
    CLOSE(OUTDEVICE);
END;


PROCEDURE PRINT2LIST;
   (* PRINTS LIST OF GENERAL STUDENT INFORMATION *)


BEGIN
  REWRITE(OUTDEVICE,'PRINTER:');
  PAGE(OUTDEVICE);
  IF FRONT = NIL
    THEN
      BEGIN
        WRITELN(OUTDEVICE,'     GENERAL INFORMATION STUDENT LIST IS EMPTY');
      END
    ELSE
      BEGIN
        WRITELN(OUTDEVICE,'             GENERAL INFORMATION STUDENT RECORDS');
        WRITELN(OUTDEVICE);
        WRITE(OUTDEVICE,'   STUDENT NAME        ST.NO.  HR      BIRTH     ');
        WRITELN(OUTDEVICE,'  PHONE   SEX');
        WRITELN(OUTDEVICE);
        WHILE FRONT <> NIL DO
          BEGIN
            WITH FRONT^ DO
              BEGIN
                WRITE(OUTDEVICE,STUDNAM,'    ');
                FOR NUMBER := 1 TO (20 - LENGTH(STUDNAM)) DO
                   WRITE(OUTDEVICE,' ');
                WRITELN(OUTDEVICE,STUDID:6,HR:5,BIRTH:10,PHONE:10,SEX:4);
                WRITELN(OUTDEVICE,'       ',STREET);
                WRITELN(OUTDEVICE,'       ',TOWN,'    ',ZIP);
              END;
            WRITELN(OUTDEVICE);
            FRONT := FRONT^.LINK2;
          END
```

```
    CLOSE(OUTDEVICE);
END;


PROCEDURE FINDGRADE;
   (* FINDS OUT STUDENT GRADE FROM USER *)


    BEGIN
      PAGE(OUTPUT);
      GOTOXY(0,2);
      WRITELN('FOR WHAT GRADE DO YOU WISH TO ENTER STUDENTS?');
      WRITELN('ENTER (7,8,9,10,11 OR 12) :         ');
      GOTOXY(30,3);
      READLN(GRAD);
    END;


BEGIN                                   (* UNIT ONE *)

END.                                    (* UNIT ONE *)
```

10.3. CMATRIX

```
PROGRAM CMATRIX;

    (*********************************************************************)
    (*                                                                   *)
    (*     PRODUCES CONFLICT MATRIX                                      *)
    (*                                                                   *)
    (*     THE USER ENTERS                                               *)
    (*        - THE RANGE OF GRADES IN THE SCHOOL                        *)
    (*        - THE NUMBER OF THE COURSE THAT HE/SHE WANTS A LIST FOR    *)
    (*                                                                   *)
    (*     FILES USED                                                    *)
    (*        - CR:COURSEFILE                                            *)
    (*        - ST:STREC(GRADE)                                          *)
    (*                                                                   *)
    (*     OUTPUT                                                        *)
    (*        - A LIST OF EVERY COURSE WHICH WOULD CONFLICT WITH THE     *)
    (*          INPUTED COURSE AND A TALLY OF HOW MANY STUDENTS HAVE     *)
    (*          THAT GIVEN COMBINATION OF COURSES.                       *)
    (*                                                                   *)
    (*********************************************************************)


CONST
  MAXN      = 10;                          (* MAX NUMBER OF COURSES *)
  MAXC      = 999;                         (* MAX COURSE NUMBER     *)

TYPE
  POINT1            = ^STUDENT;            (* TO STUDENT RECORD     *)
  POINT2            = ^COURSE;             (* TO COURSE RECORD      *)
  YEAR              = 7..12;
  COURSENUMBER      = 0..999;
  COURSESECTION     = 1..99;
  SEMESTER          = 1..3;
  PERIODNUMBER      = 1..16;
  TEACHERNUMBER     = 1..99;
  COURREQLIST       = ARRAY[1..MAXN] OF COURSENUMBER;
  STUDENT           =
     RECORD
       STUDNAM      : STRING[20];         (* STUDENT NAME *)
       STUDID       : INTEGER;            (* STUDENT ID NUMBER *)
       GRADE        : YEAR;               (* GRADE IN SCHOOL *)
       CRSREQ       : COURREQLIST;        (* ARRAY OF STUDENT COURSES *)
       LINK1        : POINT1;             (* POINTER TO NEXT RECORD *)
     END;
  COURSE            =
    RECORD
      CRNUM     : COURSENUMBER;           (* COURSE NUMBER (0..999) *)
      CRNAM     : STRING[10];             (* COURSE NAME ( 10 CHAR) *)
      CRSEC     : COURSESECTION;          (* COURSE SECTION (1..99) *)
      MAX       : INTEGER;                (* MAX SIZE OF CLASS      *)
      PERIOD    : PERIODNUMBER;           (* PERIOD OF DAY (1..16   *)
      PD        : PERIODNUMBER;           (* PERIOD FOR SCHEDULING  *)
      DAY       : STRING[5];              (* DAY OF WEEK (MTWRF)    *)
      SEM       : SEMESTER;               (* SEMESTER  (1..2..3)    *)
      ROOM      : STRING[3];              (* ROOM NUMBER            *)
      TEACHER   : TEACHERNUMBER;          (* TEACHER NUMBER         *)
      LINK2     : POINT2;                 (* POINTER TO NEXT RECORD *)
    END;
```

```
  NAMEA             = ARRAY[1..MAXC] OF STRING[10];
  TALLY             = ARRAY[1..MAXC] OF INTEGER;

 VAR
   LOW,                                      (* LOWEST GRADE IN SCHOOL  *)
   HIGH             : YEAR;                  (* HIGHEST GRADE IN SCHOOL *)
   CRSNA            : NAMEA;                 (* COURSE NAME ARRAY *)
   NUMBER           : COURSENUMBER;          (* MATRIX REQUEST *)
   COUNT            : TALLY;                 (* COURSE COUNT *)


PROCEDURE WAIT;
   (* ALLOWS USER TO DECIDE WHEN TO CONTINUE *)

VAR CH  : CHAR;                             (* ANY  KEY *)

BEGIN
GOTOXY(10,23);
WRITELN('[ PRESS ANY KEY TO CONTINUE]');
READ(CH)
END;

FUNCTION ASKCHOICE(X,Y :INTEGER; Q :STRING; MIN,MAX :INTEGER) : INTEGER;
   (* ASK USER FOR A NUMBER FROM MIN TO MAX, AND REPEAT UNTIL OBTAINED *)

VAR   ANSWER      : INTEGER;                (* NUMBER INPUTED *)

BEGIN
REPEAT
  GOTOXY(X,Y);
  WRITE(Q,' ( ',MIN,'..',MAX,' ) : ');
  READLN(ANSWER);
UNTIL (ANSWER >= MIN) AND (ANSWER <= MAX);
ASKCHOICE := ANSWER
END;

PROCEDURE INITIALIZE( VAR N  :NAMEA; VAR T : TALLY);
   (* INITIALIZES NAMEARRAY AND  TALLYARRAY *)

VAR   ROW      : COURSENUMBER;              (* LIST INDEX *)

BEGIN

FOR ROW := 1 TO MAXC DO                     (* INITIALIZES THE NAME  ARRAY *)
  N[ROW] := '  ';

FOR ROW := 1 TO MAXC DO                     (* INITIALIZES THE COUNT ARRAY *)
  T[ROW] := 0;

END;

PROCEDURE GETCRSNAMES(VAR TABLE1 : NAMEA);
   (* LOADS COURSE NAMES INTO COURSE ARRAY *)

VAR  CRREC        : FILE OF COURSE;         (* COURSE RECORDS *)

BEGIN
RESET(CRREC, 'CR:COURSEFILE');

WHILE NOT EOF (CRREC) DO
  BEGIN
    TABLE1[CRREC^.CRNUM] := CRREC^.CRNAM;
    GET (CRREC);
  END;

CLOSE(CRREC);
```

```
END;

PROCEDURE FILLMAT(L,H : YEAR; VAR C : TALLY; VAR N :NAMEA;CNUM :COURSENUMBER);
   (* FINDS ALL COURSES WHICH MIGHT BE CONFLICTS *)

VAR
  GETOUT          : BOOLEAN;              (* EXIT LOOP *)
  NULLA           : COURREQLIST;          (* NULL ARRAY *)
  PRESENT         : YEAR;                 (* CURRENT VALUE OF YEAR *)
  STREC           : FILE OF STUDENT;      (* STUDENT RECORD *)
  FILEGRADE,                              (* STRING VALUE OF YEAR *)
  FILENAME        : STRING;               (* NAME OF STUDENT FILE *)
  I , J           : INTEGER;              (* LOOP COUNTERS *)

BEGIN
  FOR J := 1 TO MAXN DO                  (* SET NULL ARRAY TO 0 *)
    NULLA[J] := 0;

  GETOUT := FALSE;
  PRESENT := L;
  REPEAT
    STR(PRESENT,FILEGRADE);
    FILENAME := CONCAT('ST:STREC',FILEGRADE);
    RESET(STREC,FILENAME);
    WHILE NOT EOF (STREC) DO
      BEGIN
        IF STREC^.CRSREQ <> NULLA
          THEN
            BEGIN
              FOR J := 1 TO MAXN DO
                BEGIN
                  IF STREC^.CRSREQ[J] = CNUM
                    THEN
                     BEGIN
                      J := MAXN;
                      FOR I := 1 TO MAXN DO
                       BEGIN
                        WITH STREC^ DO
                         BEGIN
                          IF CRSREQ[I] > 0
                           THEN C[CRSREQ[I]] := C[CRSREQ[I]] + 1;
                         END;
                       END;
                     END
                    ELSE
                      BEGIN
                       IF STREC^.CRSREQ[J]   CNUM
                         THEN J := MAXN
                       END;
                END;
            END;
        GET(STREC);
      END;
    CLOSE(STREC);
    IF PRESENT = H
      THEN GETOUT := TRUE
      ELSE PRESENT := PRESENT + 1;
  UNTIL GETOUT = TRUE
END;

PROCEDURE PRINTMAT(C : TALLY; N  :NAMEA; NUM : COURSENUMBER);
   (* PRINTS THE CONFLICT MATRIX *)

VAR ROW : COURSENUMBER;                   (* LIST INDEX *)
    OUTDEVICE : TEXT;                     (* PRINTER *)
```

```
BEGIN
REWRITE(OUTDEVICE,'PRINTER:');
PAGE(OUTDEVICE);
WRITE(OUTDEVICE,'                          ');
WRITELN(OUTDEVICE,'CONFLICT MATRIX FOR COURSE ',NUM);
WRITELN(OUTDEVICE);WRITELN(OUTDEVICE);
WRITE(OUTDEVICE,'           ');
WRITELN(OUTDEVICE,'CRS NAME        CODE      TALLY');
WRITELN(OUTDEVICE);

FOR ROW := 1 TO MAXC DO
  BEGIN
    IF C[ROW] <> 0
      THEN
        BEGIN
          WRITE(OUTDEVICE,'          ');
          WRITE(OUTDEVICE,N[ROW],'       ');
          WRITE(OUTDEVICE,' ':11 - LENGTH(N[ROW]));
          WRITE(OUTDEVICE,ROW:3,'       ');
          WRITELN(OUTDEVICE,C[ROW]:3);
        END
  END;
CLOSE(OUTDEVICE);
END;

BEGIN                                          (* CONFLICT MATRIX *)

PAGE(OUTPUT);
LOW := ASKCHOICE(0,5,'WHAT IS THE LOWEST GRADE IN THE SCHOOL',7,12);
HIGH := ASKCHOICE(0,10,'WHAT IS THE HIGHEST GRADE IN THE SCHOOL',7,12);
WAIT;
PAGE(OUTPUT);
NUMBER := ASKCHOICE(0,10,'FOR WHAT COURSE WOULD YOU LIKE A MATRIX :  ',0,999);

WHILE NUMBER <> 0 DO
  BEGIN

    INITIALIZE(CRSNA,COUNT);

    GETCRSNAMES(CRSNA);

    FILLMAT(LOW,HIGH,COUNT,CRSNA,NUMBER);

    PRINTMAT(COUNT,CRSNA,NUMBER);

    PAGE(OUTPUT);

    WRITELN('ENTER 0 TO EXIT');

    NUMBER := ASKCHOICE(0,10,'IF YOU WOULD LIKE ANOTHER COURSE MATRIX ENTER THE
NUMBER  : ',0,999);

  END;

END.                                           (* CONFLICT MATRIX *)
```

10.4. TALLYLIST

```
PROGRAM TALLYLIST;

    (*********************************************************************)
    (*                                                                   *)
    (*     PRINTS  THE COURSE TALLY LIST                                 *)
    (*                                                                   *)
    (*     THE USER ENTERS                                               *)
    (*        - THE RANGE OF GRADES TO BE TALLIED                        *)
    (*                                                                   *)
    (*     FILES USED                                                    *)
    (*        - CR:COURSEFILE                                            *)
    (*        - ST:STREC(GRADE)                                          *)
    (*                                                                   *)
    (*     OUTPUT - A LIST OF                                            *)
    (*        - COURSE NAME                                              *)
    (*        - COURSE NUMBER                                            *)
    (*        - TOTAL COURSE ENROLLMENT                                  *)
    (*        - ENROLLMENT BY GRADE                                      *)
    (*                                                                   *)
    (*********************************************************************)


CONST
  MAXN      = 10;                       (* MAX NUMBER OF COURSES *)
  MAXC      = 999;                      (* MAX COURSE NUMBER     *)

TYPE
  POINT1            = ^STUDENT;         (* TO STUDENT RECORD     *)
  POINT2            = ^COURSE;          (* TO COURSE RECORD      *)
  YEAR              = 7..12;
  COURSENUMBER      = 0..999;
  COURSESECTION     = 1..99;
  SEMESTER          = 1..3;
  PERIODNUMBER      = 1..16;
  TEACHERNUMBER     = 1..99;
  COURREQLIST       = ARRAY[1..MAXN] OF COURSENUMBER;
  STUDENT           =
     RECORD
       STUDNAM      : STRING[20];       (* STUDENT NAME *)
       STUDID       : INTEGER;          (* STUDENT ID NUMBER *)
       GRADE        : YEAR;             (* GRADE IN SCHOOL *)
       CRSREQ       : COURREQLIST;      (* ARRAY OF STUDENT COURSES *)
       LINK1        : POINT1;           (* POINTER TO NEXT RECORD *)
     END;
  COURSE            =
    RECORD
      CRNUM     : COURSENUMBER;         (* COURSE NUMBER (0..999) *)
      CRNAM     : STRING[10];           (* COURSE NAME ( 10 CHAR) *)
      CRSEC     : COURSESECTION;        (* COURSE SECTION (1..99) *)
      MAX       : INTEGER;              (* MAX SIZE OF CLASS      *)
      PERIOD    : PERIODNUMBER;         (* PERIOD OF DAY (1..16   *)
      PD        : PERIODNUMBER;         (* PERIOD FOR SCHEDULING  *)
      DAY       : STRING[5];            (* DAY OF WEEK (MTWRF)    *)
      SEM       : SEMESTER;             (* SEMESTER  (1..2..3)    *)
      ROOM      : STRING[3];            (* ROOM NUMBER            *)
      TEACHER   : TEACHERNUMBER;        (* TEACHER NUMBER         *)
      LINK2     : POINT2;               (* POINTER TO NEXT RECORD *)
    END;
  TWOD              = ARRAY[1..MAXC,7..12] OF INTEGER;
```

```
  NAMEA            = ARRAY[1..MAXC] OF STRING[10];

 VAR
   LOW,                                    (* LOWEST GRADE            *)
   HIGH            : YEAR;                 (* HIGHEST GRADE           *)
   MAT             : TWOD;                 (* TABLE OF CRS TALLIES    *)
   CRSNA           : NAMEA;                (* LIST OF CRS NAMES       *)


FUNCTION ASKCHOICE(X,Y :INTEGER; Q :STRING; MIN,MAX :INTEGER) : INTEGER;
   (* ASK USER FOR A NUMBER FROM MIN TO MAX, AND REPEAT UNTIL OBTAINED *)

VAR   ANSWER      : INTEGER;                (* GRADE *)

BEGIN
REPEAT
  GOTOXY(X,Y);
  WRITE(Q,' ( ',MIN,'..',MAX,' ) : ');
  READLN(ANSWER);
UNTIL (ANSWER >= MIN) AND (ANSWER <= MAX);
ASKCHOICE := ANSWER
END;

PROCEDURE INITIALIZE(VAR L,H :YEAR; VAR TABLE2 :TWOD; VAR TABLE1 :NAMEA);
   (* INITIALIZES NAMEARRAY AND COURSE TALLY MATRIX *)

VAR   ROW      : COURSENUMBER;             (* TABLE INDEX - CRS NUMBER *)
      COL      : YEAR;                     (*             - GRADE      *)

BEGIN
PAGE(OUTPUT);
L := ASKCHOICE(0,5,'WHAT IS THE LOWEST GRADE IN THE SCHOOL',7,12);
H := ASKCHOICE(0,10,'WHAT IS THE HIGHEST GRADE IN THE SCHOOL',7,12);

FOR ROW := 1 TO MAXC DO                   (* INITIALIZES THE ARRAY *)
  FOR COL := L TO H DO
    TABLE2[ROW,COL] := 0;

FOR ROW := 1 TO MAXC DO
  TABLE1[ROW] := '  ';

END;

PROCEDURE GETCRSNAMES (VAR TABLE1 : NAMEA);
   (* LOADS COURSE NAMES INTO COURSE ARRAY *)

VAR  CRREC        : FILE OF COURSE;       (* COURSE RECORDS *)

BEGIN
RESET(CRREC,'CR:COURSEFILE');

WHILE NOT EOF (CRREC) DO
  BEGIN
    TABLE1[CRREC^.CRNUM] := CRREC^.CRNAM;
    GET (CRREC);
  END;

CLOSE(CRREC);

END;

PROCEDURE GETSTUDREQ(L,H : YEAR; VAR TABLE2 :TWOD; VAR TABLE1 :NAMEA);
   (* TALLIES STUDENT REQUESTS BY COURSE NUMBER AND GRADE *)

VAR
```

```
  GETOUT          : BOOLEAN;              (* EXIT LOOP *)
  NULLA           : COURREQLIST;          (* NULL ARRAY *)
  ROW             : 1..MAXN;              (* ROW *)
  PRESENT         : YEAR;                 (* CURRENT VALUE OF YEAR *)
  STREC           : FILE OF STUDENT;      (* STUDENT RECORD *)
  FILEGRADE,                              (* STRING VALUE OF YEAR *)
  FILENAME        : STRING;               (* NAME OF STUDENT FILE *)

BEGIN
  FOR ROW := 1 TO MAXN DO                 (* SET NULL ARRAY TO 0 *)
    NULLA[ROW] := 0;

  GETOUT := FALSE;
  PRESENT := L;
  REPEAT
    STR(PRESENT,FILEGRADE);
    FILENAME := CONCAT('ST:STREC',FILEGRADE);
    RESET(STREC,FILENAME);
    WHILE NOT EOF (STREC) DO
      BEGIN
        IF STREC^.CRSREQ <> NULLA
          THEN
            BEGIN
              FOR ROW := 1 TO MAXN DO
                BEGIN
                  IF STREC^.CRSREQ[ROW] = 0
                    THEN ROW := MAXN
                    ELSE
                      BEGIN
                        TABLE2[STREC^.CRSREQ[ROW],PRESENT] :=
                          TABLE2[STREC^.CRSREQ[ROW],PRESENT] + 1
                      END;
                END;
            END;
        GET(STREC);
      END;
    CLOSE(STREC);
    IF PRESENT = H
      THEN GETOUT := TRUE
      ELSE PRESENT := PRESENT + 1;
  UNTIL GETOUT = TRUE
END;

PROCEDURE PRINTMAT(L,H :YEAR; VAR TABLE2 :TWOD; VAR TABLE1 :NAMEA);
   (* PRINTS THE TALLIES LIST *)

VAR ROW : COURSENUMBER;                   (* TABLE INDEX - COURSE NUMBER *)
    COL : YEAR;                           (*               - GRADE *)
    ENROL : INTEGER;                      (* COURSE ENROLLMENT   *)
    OUTDEVICE : TEXT;                     (* PRINTER *)

BEGIN
REWRITE(OUTDEVICE, 'PRINTER: ');
PAGE(OUTDEVICE);
WRITELN(OUTDEVICE, 'SODUS',37);
WRITELN(OUTDEVICE,'STUDENT TALLY OF COURSES OFFERED :50);
WRITELN(OUTDEVICE);WRITELN(OUTDEVICE);
WRITE(OUTDEVICE,'                                  ');
FOR COL := L TO H DO
  WRITE(OUTDEVICE,' GR   ');
WRITELN(OUTDEVICE);
WRITE(OUTDEVICE,' COURSE    CODE    TOTAL  ');
FOR COL := L TO H DO
  WRITE(OUTDEVICE,COL:3,'   ');
WRITELN(OUTDEVICE);WRITELN(OUTDEVICE);
```

```
FOR ROW := 1 TO MAXC DO
  BEGIN
    IF TABLE1[ROW] <> ' '
      THEN
        BEGIN
          ENROL := 0;
          WRITE(OUTDEVICE,TABLE1[ROW],'   ');
          WRITE(OUTDEVICE,' ':11 - LENGTH(TABLE1[ROW]));
          WRITE(OUTDEVICE,ROW:3,'     ');
          FOR COL := L TO H DO
            ENROL := ENROL + TABLE2[ROW,COL];
          WRITE(OUTDEVICE,ENROL:3,'     ');
          FOR COL := L TO H DO
            WRITE(OUTDEVICE,TABLE2[ROW,COL]:3,'   ');
          WRITELN(OUTDEVICE);
        END
  END
END;

BEGIN

INITIALIZE(LOW,HIGH,MAT,CRSNA);

GETCRSNAMES(CRSNA);

GETSTUDREQ(LOW,HIGH,MAT,CRSNA);

PRINTMAT(LOW,HIGH,MAT,CRSNA);

END.
```

10.5. ENTCOURSEDATA

```
PROGRAM ENTCOURSEDATA;

   (*****************************************************************)
   (*                                                               *)
   (*     THIS PROGRAM STORES COURSE RECORDS IN CR:COURSEFILE       *)
   (*                                                               *)
   (*     THE USER HAS 10 OPTIONS                                   *)
   (*        1. ENTER COURSE NUMBERS AND NAMES FOR FIRST TIME.      *)
   (*        2. LOAD THE COURSEFILE FROM DISK.                      *)
   (*        3. ENTER OTHER COURSE INFORMATION.                     *)
   (*        4. ADD COURSES.                                        *)
   (*        5. DELETE COURSES.                                     *)
   (*        6. CHANGE COURSE DATA.                                 *)
   (*        7. QUERY FILE BY COURSE NUMBER.                        *)
   (*        8. SEND COURSE LIST TO PRINTER.                        *)
   (*        9. SAVE COURSE LIST TO DISK.                           *)
   (*       10. EXIT PROGRAM.                                       *)
   (*                                                               *)
   (*****************************************************************)


TYPE
  POINTER           = ^COURSE;
  COURSENUMBER      = 0..999;
  COURSESECTION     = 1..99;
  SEMESTER          = 1..3;
  PERIODNUMBER      = 1..16;
  TEACHERNUMBER     = 1..99;
  COURSE            =
    RECORD
      CRNUM     : COURSENUMBER;          (* COURSE NUMBER (0..999)  *)
      CRNAM     : STRING[10];            (* COURSE NAME ( 10 CHAR)  *)
      CRSEC     : COURSESECTION;         (* COURSE SECTION (1..99)  *)
      MAX       : INTEGER;               (* MAX SIZE OF CLASS       *)
      PERIOD    : PERIODNUMBER;          (* PERIOD OF DAY (1..16    *)
      PD        : PERIODNUMBER;          (* PERIOD FOR SCHEDULING   *)
      DAY       : STRING[5];             (* DAY OF WEEK (MTWRF      *)
      SEM       : SEMESTER;              (* SEMESTER  (1..2..3)     *)
      ROOM      : STRING[3];             (* ROOM NUMBER             *)
      TEACHER   : TEACHERNUMBER;         (* TEACHER NUMBER          *)
      LINK      : POINTER;               (* POINTER TO NEXT RECORD  *)
    END;

 VAR
   CRNUM      : COURSENUMBER;
   CRNAM      : STRING[10];
   CRSEC      : COURSESECTION;
   MAX        : INTEGER;
   PD,
   PERIOD     : PERIODNUMBER;
   DAY        : STRING[5];
   TEACHER    : TEACHERNUMBER;
   SEM        : SEMESTER;
   ROOM       : STRING[3];
   NODE,                               (* POINTER TO COURSE REC   *)
   HEAD       : POINTER;               (* POINTER TO START OF LIST *)
   ANS        : CHAR;                  (* YES OR NO (Y OR N)      *)
   CRREC      : FILE OF COURSE;        (* FILE BUFFER             *)
```

```
PROCEDURE WAIT;  -
   (* ALLOWS USER TO DECIDE WHEN TO CONTINUE *)

VAR CH  : CHAR;                                  (* INPUT CHAR  *)

BEGIN
GOTOXY(10,23);
WRITELN('[ PRESS ANY KEY TO CONTINUE]');
READ(CH)
END;



PROCEDURE GETNUM(X,Y : INTEGER; VAR NUM : INTEGER;L,H:INTEGER);
   (* ERROR CK - IF A CHAR IS ENTERED INSTEAD OF AN INTEGER
      USER WILL BE ASKED TO RE-ENTER NUMBER  *)


BEGIN
  (*$I-*)                                        (* TURN I/O CHECKING OFF *)
  READLN(NUM);
  (*$I+*)                                        (* TURN I/O CHECKING ON  *)
  WHILE (NUM < L) OR (NUM > H)    DO
    BEGIN
      GOTOXY(10,22);
      WRITELN('ERROR IN INPUT - REENTER A NUMBER FROM  ,L,' TO ',H);
      GOTOXY(X,Y);
      (*$I-*)
      READLN(NUM);
      (*$I+*)
      GOTOXY(10,22);
      WRITELN('                                                              ');
    END;
  END;



 PROCEDURE ENTERCOURSESETUP;
    (* SETS UP SCREEN FOR ENTERING A COURSE RECORD *)

 BEGIN
   GOTOXY(15,3);
   WRITELN(' A COURSE RECORD');
   GOTOXY(0,6);


   WRITELN('COURSE NUMBER      :          ');
   GOTOXY(0,7);
   WRITELN('COURSE NAME        :                    ');
   GOTOXY(0,8);
   WRITELN('COURSE SECTION     :      ');
   GOTOXY(0,9);
   WRITELN('MAXSIZE            :      ');
   GOTOXY(0,10);
   WRITELN('PERIOD NUMBER      :      ');
   GOTOXY(0,11);
   WRITELN('PD NUMBER          :      ');
   GOTOXY(0,12);
   WRITELN('DAYS               :          ');
   GOTOXY(0,13);
   WRITELN('SEMESTER           :          ');
   GOTOXY(0,14);
   WRITELN('ROOM NUMBER        :      ');
   GOTOXY(0,15);
   WRITELN('TEACHER NUMBER     :       ');
```

```
END;


 PROCEDURE INSERTNODE (VAR FRONT, ITEM : POINTER);
  (* TO PUT NEW NODE IN LIST IN ORDER OF COURSE, SECTION AND PD NUMBER *)

  VAR  LOOKAHEAD,                          (* LOOKAHEAD POINTER TO FIND PLACE *)
       CHASER       : POINTER;             (* CHASER POINTER FOR INSERTION *)

 BEGIN (* INSERTNODE *)
  LOOKAHEAD := FRONT;
  CHASER := FRONT;

  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM > LOOKAHEAD^.CRNUM) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK
    END ;  (* WHILE *)

  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM = LOOKAHEAD^.CRNUM) AND
        (ITEM^.CRSEC > LOOKAHEAD^.CRSEC) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK
    END ;  (* WHILE *)

  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM = LOOKAHEAD^.CRNUM) AND
        (ITEM^.CRSEC = LOOKAHEAD^.CRSEC) AND (ITEM^.PD > LOOKAHEAD^.PD) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK
    END ;  (* WHILE *)


  IF (LOOKAHEAD <> NIL)
    THEN
      IF LOOKAHEAD <> FRONT
        THEN                               (* IN MIDDLE OF LIST *)
          BEGIN
            ITEM^.LINK := LOOKAHEAD;
            CHASER^.LINK := ITEM;
          END
        ELSE                               (* AT FRONT OF LIST *)
          BEGIN
            ITEM^.LINK := FRONT;
            FRONT := ITEM;
          END
    ELSE                                   (* MUST BE LARGEST, SO INSERT AT END*)
      CHASER^.LINK := ITEM;

  END;



  PROCEDURE PRINTLIST (FRONT : POINTER);
     (* PRINTS COURSE LIST GIVEN THE POINTER TO START OF LIST *)

  VAR OUTDEVICE : TEXT;                    (* PRINTER *)
      PREV      : COURSENUMBER;            (* PREV COURSE NUMBER *)
      PREVS     : COURSESECTION;           (* PREV SECTION NUMBER *)

    BEGIN
      REWRITE(OUTDEVICE, 'PRINTER:');
      PAGE (OUTDEVICE);
```

```
      IF FRONT = NIL
        THEN                                (* LIST IS EMPTY *)
          BEGIN
            WRITELN (OUTDEVICE,'LIST IS EMPTY.');
          END
        ELSE                                (* THERE ARE COURSES *)
          BEGIN
            WRITELN(OUTDEVICE,'COURSE LIST - SODUS':42);
            WRITELN(OUTDEVICE);
            WRITE(OUTDEVICE,'CODE  SEC  SEATS  SEM  PD  DAYS MET  ');
            WRITELN(OUTDEVICE,'DESCRIPTION  ROOM  TCH  PER');
            PREV := 0;
            PREVS := 1;
            WHILE FRONT <>  NIL DO
              BEGIN
                WITH FRONT^ DO
                BEGIN
                IF PREV < CRNUM THEN  WRITELN(OUTDEVICE);
                IF (PREV = CRNUM) AND (PREVS < CRSEC )
                  THEN WRITELN(OUTDEVICE);
                WRITE(OUTDEVICE,CRNUM:3,'   ',CRSEC:2,'    ',MAX:3,'      ');
                WRITE(OUTDEVICE,SEM,'   ',PD:2,'    ',DAY:5,'     '.CRNAM:10,

                WRITELN(OUTDEVICE,ROOM:3,'   ',TEACHER:2,'    ',PERIOD:2);
                END;
                PREV := FRONT^.CRNUM;
                PREVS := FRONT^.CRSEC;
                FRONT := FRONT^.LINK;
              END;
            WRITELN(OUTDEVICE);WRITELN(OUTDEVICE);
          END;
       CLOSE(OUTDEVICE);
     END;

PROCEDURE FILEIT (FRONT : POINTER);
   (* SENDS COURSE RECORD TO FILE NAMED ' COURSEFILE ' *)

    BEGIN
      REWRITE(CRREC,'CR:COURSEFILE');
      IF FRONT = NIL
        THEN                                (* LIST IS EMPTY *)
          BEGIN
            GOTOXY (10,12);
            WRITELN ('LIST IS EMPTY.');
          END
        ELSE                                (* THERE ARE COURSES *)
          BEGIN
            WHILE FRONT <>  NIL DO
              BEGIN
                CRREC^ := FRONT^;
                CRREC^.LINK := NIL;
                PUT(CRREC);
                FRONT := FRONT^.LINK;
              END;
          CLOSE(CRREC,LOCK);
          END
  END;


PROCEDURE SETUP;
  (* FORMATS SCREEN FOR ENTRY OF COURSE NUMBER AND NAME *)

BEGIN
  GOTOXY(0,10);
  WRITELN('ENTER COURSE NUMBER :     ');
  GOTOXY(0,15);
```

```
  WRITELN('ENTER COURSE NAME :              ');
  GOTOXY(22,10);
END;


PROCEDURE OPTION1;
    (* ALLOWS USER TO ENTER COURSE NUMBER AND NAME FOR FIRST TIME
       DUMMY VALUES ADDED FOR OTHER FIELDS  *)

BEGIN
  PAGE(OUTPUT);
  WRITELN('ENTER COURSE NUMBER');
  WRITELN('ON NEXT LINE ENTER COURSE NAME');
  WRITELN('WHEN YOU ARE FINISHED ENTER 999 AND STOP');
  WRITELN;WRITELN;

  SETUP;
  NEW(NODE);
      WITH NODE^ DO
        BEGIN
          GETNUM(22,10,CRNUM,1,999);
          IF CRNUM <> 999
            THEN
              BEGIN
                GOTOXY(20,15);
                READLN(CRNAM);
                CRSEC := 1;
                MAX := 35;
                PERIOD := 1;
                PD := 1;
                DAY := 'MTWRF';
                SEM := 3;
                ROOM := 'N';
                TEACHER := 99;
                LINK := NIL;
              END;
        END;
  WHILE NODE^.CRNUM <> 999 DO
    BEGIN

      IF HEAD = NIL
        THEN HEAD := NODE
        ELSE INSERTNODE( HEAD, NODE);
        NEW(NODE);
        SETUP;
      WITH NODE^ DO
        BEGIN
          GETNUM(22,10,CRNUM,1,999);
          GOTOXY(20,15);
          READLN(CRNAM);
          CRSEC := 1;
          MAX := 35;
          PERIOD := 1;
          PD := 1;
          DAY := 'MTWRF';
          SEM := 3;
          ROOM := 'N';
          TEACHER := 99;
          LINK := NIL;
        END;
    END;
END;

PROCEDURE FETCHIT(VAR FIRST : POINTER);
    (* RETRIEVES COURSE LIST FROM FILE AND ENTERS COURSES IN A
       LINKED LIST *)
```

```
VAR  CRS   : POINTER;                              (* NEW CRS NODE *)

BEGIN
RESET(CRREC,'CR:COURSEFILE');

WHILE NOT EOF(CRREC) DO
  BEGIN

    NEW(CRS);
    CRS^ := CRREC^;
    IF FIRST = NIL
      THEN FIRST := CRS
      ELSE INSERTNODE(FIRST,CRS);
    GET(CRREC);
  END;
CLOSE(CRREC);
END;


PROCEDURE MENU(VAR NUMBER : INTEGER);
   (* SENDS MAIN MENU TO SCREEN AND WAITS FOR USER CHOICE *)

BEGIN
PAGE(OUTPUT);
GOTOXY(30,3);
WRITELN('COURSE LIST OPTIONS');
GOTOXY(12,5);
WRITELN('1. ENTER COURSE NUMBERS AND NAMES FOR FIRST TIME. ');
GOTOXY(12,6);
WRITELN('2. LOAD COURSE FILE FROM DISK.');
GOTOXY(12,7);
WRITELN('3. ENTER OTHER COURSE INFORMATION. ');
GOTOXY(12,8);
WRITELN('4. ADD COURSES.');
GOTOXY(12,9);
WRITELN('5. DELETE COURSES.');
GOTOXY(12,10);
WRITELN('6. CHANGE COURSE DATA.');
GOTOXY(12,11);
WRITELN('7. QUERY FILE BY COURSE NUMBER.');
GOTOXY(12,12);
WRITELN('8. SEND COURSE LIST TO PRINTER.');
GOTOXY(12,13);
WRITELN('9. SAVE COURSE LIST TO DISK.');
GOTOXY(12,14);
WRITELN('10. EXIT PROGRAM.');
GOTOXY(15,20);
WRITELN('ENTER A NUMBER FROM 1 TO 10');
GOTOXY(15,21);
WRITELN('WHICH OPTION :        ');
GOTOXY(32,21);
GETNUM(32,21,NUMBER,1,10)
END;


PROCEDURE ENTERINFO(FRONT : POINTER);
   (* ALLOWS USER TO INPUT INFORMATION MISSING FROM A COURSE RECORD *)

BEGIN
  WHILE FRONT <> NIL DO
    BEGIN
      PAGE(OUTPUT);
      ENTERCOURSESETUP;
      GOTOXY(5,22);
      WRITE('FOR COURSE NUMBER  ');
```

```
        WRITE(FRONT^.CRNUM);
        WRITELN(' FILL IN THE MISSING INFORMATION');
        GOTOXY(20,6);
        WRITELN(FRONT^.CRNUM);
        GOTOXY(20,7);
        WRITELN(FRONT^.CRNAM);
        GOTOXY(20,8);
        GETNUM(20,8,FRONT^.CRSEC,1,99);
        GOTOXY(20,9);
        GETNUM(20,9,FRONT^.MAX,1,999);
        GOTOXY(20,10);
        GETNUM(20,10,FRONT^.PERIOD,1,16);
        GOTOXY(20,11);
        GETNUM(20,11,FRONT^.PD,1,16);
        GOTOXY(20,12);
        READLN(FRONT^.DAY);
        GOTOXY(20,13);
        GETNUM(10,13,FRONT^.SEM,1,3);
        GOTOXY(20,14);
        READLN(FRONT^.ROOM);
        GOTOXY(20,15);
        GETNUM(20,15,FRONT^.TEACHER,1,99);

        FRONT := FRONT^.LINK
      END;

END;

PROCEDURE ADDC;
    (* ALLOWS USER TO INPUT A NEW COURSE TO LIST *)


BEGIN
  PAGE(OUTPUT);
  NEW(NODE);
  ENTERCOURSESETUP;
  GOTOXY(20,6);
  GETNUM(20,6,NODE^.CRNUM,1,999);
  GOTOXY(20,7);
  READLN(NODE^.CRNAM);
  GOTOXY(20,8);
  GETNUM(20,8,NODE^.CRSEC,1,99);
  GOTOXY(20,9);
  GETNUM(20,9,NODE^.MAX,1,999);
  GOTOXY(20,10);
  GETNUM(20,10,NODE^.PERIOD,1,16);
  GOTOXY(20,11);
  GETNUM(20,11,NODE^.PD,1,16);
  GOTOXY(20,12);
  READLN(NODE^.DAY);
  GOTOXY(20,13);
  GETNUM(20,13,NODE^.SEM,1,3);
  GOTOXY(20,14);
  READLN(NODE^.ROOM);
  GOTOXY(20,15);
  GETNUM(20,15,NODE^.TEACHER,1,99);
  NODE^.LINK := NIL;

  INSERTNODE(HEAD,NODE);


END;

PROCEDURE FINDNODE (VAR LOOKAHEAD,CHASER, ITEM : POINTER; VAR TEST,DUP :BOOLEAN)
;
  (* TO LOCATE A NODE USING THE COURSE NUMBER
```

```
      IF ANOTHER NODE WITH SAME CR NUMBER EXISTS DUP IS SET TRUE*)

VAR TEMPA : POINTER;                          (* TEMP LOOKAHEAD POINTER *)


BEGIN

  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM > LOOKAHEAD^.CRNUM) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK
    END ;

  IF (LOOKAHEAD <> NIL) AND  (ITEM^.CRNUM = LOOKAHEAD^.CRNUM )
     THEN
       BEGIN
         ITEM^ := LOOKAHEAD^;
         TEMPA := LOOKAHEAD^.LINK;
         IF (TEMPA <> NIL) AND (ITEM^.CRNUM = TEMPA^.CRNUM)
           THEN DUP := TRUE
       END;


  IF (  LOOKAHEAD = NIL) OR (ITEM^.CRNUM < LOOKAHEAD^.CRNUM)
     THEN
       BEGIN
       GOTOXY(10,12);
       WRITE('COURSE ');
       WRITE( ITEM^.CRNUM );
       WRITELN(' IS NOT IN THE LIST');
       TEST := TRUE;
       WAIT;
       END;
END;

PROCEDURE WRINFO(FRONT : POINTER);
   (* WRITE A COURSE RECORD TO SCREEN *)

BEGIN
      PAGE(OUTPUT);
      ENTERCOURSESETUP;
      GOTOXY(20,6);
      WRITELN(FRONT^.CRNUM);
      GOTOXY(20,7);
      WRITELN(FRONT^.CRNAM);
      GOTOXY(20,8);
      WRITELN(FRONT^.CRSEC);
      GOTOXY(20,9);
      WRITELN(FRONT^.MAX);
      GOTOXY(20,10);
      WRITELN(FRONT^.PERIOD);
      GOTOXY(20,11);
      WRITELN(FRONT^.PD);
      GOTOXY(20,12);
      WRITELN(FRONT^.DAY);
      GOTOXY(20,13);
      WRITELN(FRONT^.SEM);
      GOTOXY(20,14);
      WRITELN(FRONT^.ROOM);
      GOTOXY(20,15);
      WRITELN(FRONT^.TEACHER);
END;

PROCEDURE CKDUP(ST:STRING;VAR L,C,N:POINTER; VAR F,D:BOOLEAN);
   (* ALLOWS USER TO CHECK THE COURSE WITH SAME CR NUMBER *)
```

```
  VAR ANS : CHAR;                                   (* Y OR N - ANSWER *)

  BEGIN
    GOTOXY(0,17);
    WRITELN('THERE IS ANOTHER COURSE WITH NUMBER ',N^.CRNUM);
    WRITE('WOULD YOU LIKE TO ',ST,' THE OTHER SECTION? ');
    WRITE('(ENTER Y OR N): ');
    READLN(ANS);
    IF ANS = 'Y'
      THEN
        BEGIN
          FINDNODE(L,C,N,F,D);
          PAGE(OUTPUT);
          WRINFO(N)
        END
      ELSE F := TRUE;
  END;

  PROCEDURE QUERY;
     (* USES COURSE NUMBER WHICH IS INPUTED TO LOCATE A COURSE *)

  VAR DUP,                                          (* ANOTHER NODE WITH SAME NUMBER EXISTS*)
      FLAG :BOOLEAN;                                (* COURSE NOT IN LIST *)
      LOOK,                                         (* LOOKAHEAD PTR TO FIND DUP *)
      CHAS :POINTER;                                (* CHASER TO MARK NODE *)

  BEGIN
    NEW(NODE);
    PAGE(OUTPUT);
    FLAG := FALSE;
    DUP  := FALSE;
    LOOK := HEAD;
    CHAS := HEAD;
    GOTOXY(0,2);
    WRITELN('WHAT COURSE WOULD YOU LIKE TO FIND?');
    WRITELN('ENTER THE COURSE NUMBER');
    WRITELN('COURSE NUMBER :      ');
    GOTOXY(16,4);
    GETNUM(16,4,NODE^.CRNUM,1,999);
    FINDNODE(LOOK,CHAS,NODE,FLAG,DUP);
    IF FLAG = FALSE
     THEN
       BEGIN
         WRINFO(NODE);
         WHILE DUP DO
           BEGIN
             CHAS := LOOK;
             LOOK := LOOK^.LINK;
             DUP := FALSE;
             CKDUP('FIND',LOOK,CHAS,NODE,FLAG,DUP);
           END;
         WAIT;
       END
  END;


  PROCEDURE REMOVE(VAR LOOK,CHAS,NODE : POINTER; VAR R:BOOLEAN);

  VAR ANS : CHAR;                                   (* Y OR N - ANSWER *)

  BEGIN
    GOTOXY(10,23);
    WRITE('DO YOU WANT TO DELETE THIS RECORD? (Y OR N)  ');
    READLN(ANS);
    IF ANS = 'Y'
      THEN
```

```
      BEGIN
        IF (HEAD^.CRNUM = NODE^.CRNUM) AND (HEAD^.CRSEC = HEAD^.CRSEC) AND
           (HEAD^.PD = NODE^.PD)
             THEN
               BEGIN                                  (* FIRST IN LIST*)
               HEAD := LOOK^.LINK;
               END
             ELSE
               BEGIN
               CHAS^.LINK := LOOK^.LINK;
               LOOK^.LINK := NIL;
               LOOK       := CHAS^.LINK;
               END;
          PAGE(OUTPUT);
          GOTOXY(0,5);
          WRITE('COURSE   ');
          WRITE(NODE^.CRNUM);
          WRITELN('  IS DELETED');
          R := TRUE;
        END;
 WAIT;


END;



PROCEDURE DELETEC;
   (* REMOVES A COURSE FROM THE LIST *)

VAR DUP,                                 (* ANOTHER CRS WITH SAME NUMBER EXISTS *)
    REM,                                 (* COURSE REMOVED *)
    FLAG :BOOLEAN;                       (* COURSE NOT IN LIST *)
    LOOK,                                (* LOOKAHEAD TO FIND DUP *)
    CHAS :POINTER;                       (* CHASER TO MARK NODE *)
    ANS  : CHAR;

BEGIN
  NEW(NODE);
  PAGE(OUTPUT);
  FLAG := FALSE;
  DUP  := FALSE;
  REM  := FALSE;
  LOOK := HEAD;
  CHAS := HEAD;
  GOTOXY(0,2);
  WRITELN('WHAT COURSE WOULD YOU LIKE TO DELETE?');
  WRITELN('ENTER THE COURSE NUMBER');
  WRITELN('COURSE NUMBER :      ');
  GOTOXY(16,4);
  GETNUM(16,4,NODE^.CRNUM,1,999);
  FINDNODE(LOOK,CHAS,NODE,FLAG,DUP);
  IF FLAG = FALSE
   THEN
     BEGIN
         WRINFO(NODE);
         REMOVE(LOOK,CHAS,NODE,REM);
         WHILE DUP DO
           BEGIN
           IF REM = FALSE
             THEN
               BEGIN
                 CHAS := LOOK;
                 LOOK := LOOK^.LINK;
               END;
           REM := FALSE;
```

```
            DUP := FALSE;
            CKDUP('FIND',LOOK,CHAS,NODE,FLAG,DUP);
            IF NOT FLAG
              THEN REMOVE(LOOK,CHAS,NODE,REM);
            END;

      END;

END;


PROCEDURE CHANGEC;
   (* ALLOWS FOR CHANGE OF ANY FIELD IN COURSE RECORD *)

VAR DUP,                              (* ANOTHER COURSE WITH SAME NUMBER EXISTS *)
    FLAG :BOOLEAN;                    (* COURSE NOT IN LIST *)
    NNODE,                            (* NEW NODE FOR CHANGES *)
    LOOK,                             (* LOOKAHEAD PTR *)
    CHAS :POINTER;                    (* CHASER *)
    ANS  : CHAR;                      (* Y OR N - ANSWER *)

 PROCEDURE DUPNODE(VAR N,L,C: POINTER; VAR F,D : BOOLEAN);
    (* CHECKS FOR NODES WITH THE SAME CR NUM *)

 BEGIN
   GOTOXY(0,17);
   WRITE('IS THIS THE RECORD YOU WISH TO CHANGE? (ENTER Y OR N) : ');
   READLN(ANS);
   IF ANS = 'N'
     THEN
      BEGIN
        WHILE D DO
          BEGIN
            GOTOXY(0,19);
            WRITELN('THERE IS ANOTHER RECORD WITH THE SAME CR NUMBER');
            WRITE('WOULD YOU RATHER CHANGE THAT RECORD? (ENTER Y OR N) : ');
            READLN(ANS);
            IF ANS = 'Y'
              THEN
                BEGIN
                  C := L;
                  L := L^.LINK;
                  D := FALSE;
                  FINDNODE(L,C,N,F,D);
                  PAGE(OUTPUT);
                  WRINFO(NODE)
                END
              ELSE D := FALSE;
          END
      END
   END;


BEGIN
  NEW(NODE);
  PAGE(OUTPUT);
  FLAG := FALSE;
  DUP  := FALSE;
  LOOK := HEAD;
  CHAS := HEAD;
  GOTOXY(0,2);
  WRITELN('WHAT COURSE WOULD YOU LIKE TO CHANGE?');
  WRITELN('ENTER THE COURSE NUMBER');
  WRITELN('COURSE NUMBER :      ');
  GOTOXY(16,4);
  GETNUM(16,4,NODE^.CRNUM,1,'99');
```

```
    FINDNODE(LOOK,CHAS,NODE,FLAG,DUP);
    IF FLAG = FALSE
     THEN
       BEGIN
         WRINFO(NODE);
         DUPNODE(NODE,LOOK,CHAS,FLAG,DUP);
         GOTOXY(45,2);
         WRITELN('MAKE CHANGES TO THIS RECORD');
         GOTOXY(45,3);
         WRITE('ENTER Y OR N :');
         READLN(ANS);
         IF ANS = 'Y' THEN
         BEGIN
           GOTOXY(45,4);
           WRITELN('RENTER ALL INFORMATION');
           NEW(NNODE);
           GOTOXY(40,6);
           GETNUM(40,6,NNODE^.CRNUM,1,999);
           GOTOXY(40,7);
           READLN(NNODE^.CRNAM);
           GOTOXY(40,8);
           GETNUM(40,8,NNODE^.CRSEC,1,99);
           GOTOXY(40,9);
           GETNUM(40,9,NNODE^.MAX,1,999);
           GOTOXY(40,10);
           GETNUM(40,10,NNODE^.PERIOD,1,16);
           GOTOXY(40,11);
           GETNUM(40,11,NNODE^.PD,1,16);
           GOTOXY(40,12);
           READLN(NNODE^.DAY);
           GOTOXY(40,13);
           GETNUM(40,13,NNODE^.SEM,1,3);
           GOTOXY(40,14);
           READLN(NNODE^.ROOM);
           GOTOXY(40,15);
           GETNUM(40,15,NNODE^.TEACHER,1,99);
           GOTOXY(10,23);
           WRITE('DO YOU WANT TO MAKE THE CHANGE IN THIS RECORD? (Y OR N)  ');
           READLN(ANS);
           IF ANS = 'Y'
             THEN
               BEGIN
                 NNODE^.LINK := NIL;
                 IF HEAD^.CRNUM = NODE^.CRNUM
                 THEN
                   BEGIN                                    (* FIRST IN LIST*)
                   HEAD := LOOK^.LINK;
                   END
                 ELSE
                   BEGIN
                   CHAS^.LINK := LOOK^.LINK;
                   LOOK^.LINK := NIL;
                   END;
                 INSERTNODE(HEAD,NNODE);
                 WRINFO(NNODE);
                 GOTOXY(10,2);
                 WRITELN('THE CHANGE HAS BEEN MADE');
                 WAIT;
               END;
             END;
       END;

END;
```

```
PROCEDURE SELECTOPTION;
   (* ALLOWS USER TO SELECT AN OPTION *)

VAR  OPTION    : INTEGER;                          (* OPTION NUMBER *)

BEGIN

REPEAT
PAGE(OUTPUT);
MENU(OPTION);

CASE OPTION OF
  1   : OPTION1;
  2   : FETCHIT(HEAD);
  3   : ENTERINFO(HEAD);
  4   : ADDC;
  5   : DELETEC;
  6   : CHANGEC;
  7   : QUERY;
  8   : PRINTLIST(HEAD);
  9   : FILEIT(HEAD);
  10  : PAGE(OUTPUT);
END;
UNTIL OPTION = 10;


END;


BEGIN                                             (* ENTERCOURSEDATA *)
  HEAD := NIL;
  PAGE(OUTPUT);

SELECTOPTION;


END.                                              (* ENTERCOURSEDATA *)
```

10.6. SCHEDULER

```
PROGRAM SCHEDULER;

    (*************************************************************************)
    (*                                                                       *)
    (*     THIS PROGRAM SCHEDULES STUDENTS AND PRINTS THE RESULTS OF THE     *)
    (*          SCHEDULING RUN.                                              *)
    (*                                                                       *)
    (*     THE USER ENTERS                                                   *)
    (*        - RUN TYPE ( TRIAL OR FINAL )                                  *)
    (*        - GRADE LEVELS                                                 *)
    (*                                                                       *)
    (*     FILES USED                                                        *)
    (*        - ST:CRREC(GRADE)                                              *)
    (*        - CR:COURSEFILE                                                *)
    (*        - SCH:SCH(GRADE)                                               *)
    (*                                                                       *)
    (*     UNIT USED                                                         *)
    (*        - UNIT TWO ( IN SYSTEM.LIBRARY )                               *)
    (*            THIS UNIT CONTAINS PROCEDURES USED IN SCHEDULING STUDENTS  *)
    (*                                                                       *)
    (*     OUTPUT                                                            *)
    (*        - SCHEDULES PRINTED FOR CONFLICT SCHEDULES.                    *)
    (*        - NUMBER OF STUDENTS ASSIGNED TO STUDY HALL ARE PRINTED        *)
    (*          BY GRADE LEVEL.                                              *)
    (*        - SCHEDULING RESULTS ARE REPORTED BY GRADE LEVEL.              *)
    (*        - CLASS TALLY RESULTS ARE PRINTED BY CRS AND SECTION           *)
    (*        - ALL STUDENTS ARE SCHEDULED IN EACH RUN BUT THE SCHEDULES     *)
    (*          ARE ONLY SAVED ON THE FINAL RUN TO FILE (SCH:SCH(GRADE))     *)
    (*                                                                       *)
    (*************************************************************************)

USES TWO;


 VAR
   CRNSCH1,                                (* 1ST COURSE NOT SCHEDULED*)
   CRNSCH2,                                (* 2ND COURSE NOT SCHEDULED*)
   CRNUM      : COURSENUMBER;              (* CRS NUMBER              *)
   RUNTYPE,                                (* TRIAL OR FINAL          *)
   CRNAM      : STRING[10];                (* COURSE NAME             *)
   CRSEC      : COURSESECTION;             (* CRS SECTION             *)
   TEST       : INTEGER;                   (* COUNTS RECORDS FOR FINAL RUN *
   PD,                                     (* SCHEDULING PERIOD       *)
   PERIOD     : PERIODNUMBER;              (* PERIOD IN DAY           *)
   DAY        : STRING[5];                 (* DAYS SCH IN WEEK        *)
   TEACHER    : TEACHERNUMBER;             (* TEAACHER NUMBER         *)
   SEM        : SEMESTER;                  (* SEM NUMBER              *)
   ROOM       : STRING[3];                 (* ROOM NUMBER             *)
   ANS        : CHAR;                      (* YES OR NO (Y OR N)       *)
   NODE       : PTR;                       (* POINTER TO COURSE REC    *)
   SH1,                                    (* STUDY HALL SCH - 1ST SEM *)
   SH2,                                    (* STUDY HALL SCH - 2ND SEM *)
   SB1,                                    (* TEMP SCHEDULE - 1ST SEM  *)
   S1,                                     (* SCHEDULE - 1ST SEM       *)
   SB2,                                    (* TEMP SCHEDULE - 2ND SEM  *)
   S2         : SCH;                       (* SCHEDULE - 2ND SEM       *)
   RUN        : CHAR;                      (* TRIAL OR FINAL RUN       *)
```

```
(*************************************************************************)
(*                                                                       *)
(*       SEGMENT PROCEDURE STUDSCHEDULE                                  *)
(*                                                                       *)
(*************************************************************************)

SEGMENT PROCEDURE STUDSCHEDULE;

CONST
   ALT1          = 700;                    (* FIRST ALTERNATE *)
   ALT2          = 701;                    (* SECOND ALTERNATE *)


 VAR
   STNODE             : POINT1;            (* STUDENT COURSE RECORD *)
   STREC              : FILE OF STUDENT;   (* FILE OF STUDENT RECORDS *)
   LENGT,                                  (* COUNTER *)
   FULL,                                   (* NUMBER OF FULL SCHEDULES *)
   PART,                                   (* NUMBER OF PARTIAL SCHED  *)
   IRR             : INTEGER;              (* NUMBER OF IRR CONFLICTS  *)
   ALTUSED,                                (* MARKS IF ALT IS USED     *)
   NOTOPEN,                                (* FILE IS NOT YET OPEN     *)
   ALTOF,                                  (* TURNS ALT OFF            *)
   GETOUT          : BOOLEAN;               (* EXIT LOOP *)
   PRESENT         : YEAR;                  (* CURRENT VALUE OF YEAR *)
   FILEGRADE,                               (* STRING VALUE OF YEAR *)
   S,                                       (* STRING VALUE OF YEAR *)
   FNAME,                                   (* NAME OF SCHEDULE FILE *)
   FILENAME        : STRING;                (* NAME OF STUDENT FILE *)
   LOW,                                     (* LOWEST GRADE IN SCHOOL  *)
   HIGH            : YEAR;                  (* HIGHEST GRADE IN SCHOOL *)
   HEAP1,                                   (* TOP OF HEAP STUD SCH    *)
   HEAP            : ^INTEGER;              (* TOP OF HEAP MARKER      *)
   ENDROW          : 1..MAXROWS;            (* LAST ROW USED IN TABLE *)
   T               : TABLE;                 (* SCHEDULING PRIORITY TABLE *)
   TAL             : TALLY;                 (* LIST OF COURSES TO BE TALLIED *)
   SCHNODE,                                 (* STUDENT SCHEDULE RECORD *)
   SHEAD           : SCHPTR;                (* HEAD OF STUD SCH LIST   *)


(*************************************************************************)
(*                                                                       *)
(*    MAKESCHEDULE                                                       *)
(*                                                                       *)
(*************************************************************************)

PROCEDURE MAKESCHEDULE(VAR TAB : TABLE; VAR S1,S2 :SCH; VAR TAL : TALLY;
                       NAME : STRING);
   (* GIVEN A PRIORITIZED LOOKUP TABLE THIS PROCEDURE SCHEDULES A STUDENTS COURS
ES *)


TYPE
    WEEK          = (M,T,W,R,F);          (* DAYS OF WEEK *)

VAR
    NUMBERCONF,                           (* COUNTS CONFLICTS *)
    PROBCRS,                              (* COURSE CAUSING CONFLICT *)
    BESTBALNUM,                           (* CRS WITH BEST BAL FOR RESCHEDULE *)
    BESTBALSEC,                           (* SEC WITH BEST BAL FOR RESCHEDULE *)
    REMOVE        : INTEGER;              (* COURSE TO BE REMOVED IF CONFLICT *)
    FINALCHANCE,                          (* TRY COURSES ON PROBLEM LIST      *)
```

```
    FOUND,                                  (* COURSE LOCATED *)
    OK,                                     (* SCHEDULED OK *)
    MARKER,                                 (* PREVIOUS CONFLICT *)
    PROB,                                   (* KEEPS TRACK OF BESTBAL *)
    CONFLICT     : BOOLEAN;                 (* CONFLICT EXISTS *)
    J            : 1..TOTC;                 (* ROWS - COURSE TALLIES LIST *)
    PROBLIST     : COURREQLIST;             (* COURSES WITH SCHEDULING PROB  *)

(*************************************************************************)

PROCEDURE REMOVETL( VAR TAL: TALLY; CRS : INTEGER);
   (* REMOVES A COURSE FROM LIST OF COURSES TO BE TALLIED *)

VAR  I       : 1..TOTC;                     (* LOOP COUNTER *)
     KEEP    : SC;                          (* HOLDS A COURSE AND SECTION *)
     FLAG    : BOOLEAN;                     (* COURSE FOUND *)

BEGIN
  KEEP[1] := 0;
  KEEP[2] := 0;
  FLAG := TRUE;
  FOR J := 1 TO TOTC DO
    IF FLAG
      THEN IF TAL[J,1] = CRS
             THEN
              BEGIN
               TAL[J] := KEEP;
               FLAG   := FALSE
              END;
  FLAG := TRUE;
  FOR J := 1 TO TOTC - 1 DO
    BEGIN
      IF FLAG
        THEN IF TAL[J,1] = 0
               THEN FOR I := J TO TOTC - 1 DO
                      BEGIN
                        TAL[I] := TAL[I + 1];
                        FLAG := FALSE
                      END
    END
END;


PROCEDURE TALLYLIST(VAR TAL : TALLY; CRS, SEC : INTEGER);
   (* KEEPS LIST OF SCHEDULED CRS FOR COURSE TALLIES LIST *)

VAR  KEEP    : SC;                          (* HOLDS ONE COURSE AND SECTION *)
     FLAG    : BOOLEAN;                     (* RECORDS INSERTON OF COURSE   *)

BEGIN
  KEEP[1] := CRS;
  KEEP[2] := SEC;
  FLAG := TRUE;
  FOR J := 1 TO TOTC DO
    IF TAL[J,1] = CRS
      THEN
       BEGIN
        TAL[J] := KEEP;
        FLAG := FALSE
       END;
  FOR J := 1 TO TOTC DO
    IF FLAG
      THEN IF TAL[J,1] = 0
             THEN
               BEGIN
                 TAL[J] := KEEP;
```

```
      END
    ELSE IF CRNSCH1 > 0
           THEN
            BEGIN
             PART := PART + 1;
             TALLYC(TAL);
             SHCOUNT(SH1,SH2,S1,S2);
             IF (RUN = 'F') OR (RUN = 'T')
               THEN
                BEGIN
                 REWRITE(OUTDEVICE,'PRINTER:');
                 WRITELN(OUTDEVICE);
                 WRITELN(OUTDEVICE,NAME);
                 WRITELN(OUTDEVICE);
                 WRITE(OUTDEVICE,'PARTIAL SCHEDULE - COURSE ',CRNSCH1);
                 WRITELN(OUTDEVICE,' IS NOT SCHEDULED');
                 CLOSE(OUTDEVICE);
                 STDSCH(S1,S2)
                END;
            END;
IF (CRNSCH1 = 0) AND (CRNSCH2 = 0)
  THEN
   BEGIN
    FULL := FULL + 1;
    TALLYC(TAL);
    SHCOUNT(SH1,SH2,S1,S2);
   END
END;

(*************************************************************************)

PROCEDURE QUERYCL(CRS,SEC : INTEGER;VAR NODE,CH,CT : PTR; VAR FOUND,NEXT : BOOLE
AN);
   (* FINDS COURSE IN CIRCULAR LIST
            - IF FOUND THEN FOUND IS TRUE AND COURSE NODE RETRIEVED
            - IF NEXT CRS IS SAME THEN NEXT SET TRUE *)

VAR
   CHMARK  : PTR;                        (* MARKS HEAD OF LIST *)


BEGIN
  FOUND := FALSE;
  CHMARK := CH;
  REPEAT
    IF (CRS = CH^.CRNUM) AND (SEC = CH^.CRSEC)
      THEN
        BEGIN
          NODE := CH;
          FOUND := TRUE
        END;
    CH := CH^.LINK;
  UNTIL FOUND OR ( CH = CHMARK);
  IF FOUND
    THEN IF (CRS = CH^.CRNUM) AND (SEC = CH^.CRSEC)
            THEN NEXT := TRUE
            ELSE NEXT := FALSE;
END;

PROCEDURE SCHEDULECRS(CRS,SEC:INTEGER; VAR OK,CONFLICT,MARKER : BOOLEAN;
                      VAR PROBCRS :INTEGER; VAR SB1,SB2 : SCH);
   (* PLACES A CRS AND SEC INTO SCHEDULE ARRAY- SETS OK,CONFLICT AND
      MARKER FLAGS TO INDICATE RESULT OF OPERATION *)

VAR
  NEXT  : BOOLEAN;                    (* = TRUE IF ANOTHER PD OF THIS SECTION EXISTS-
```

```
                - FLAG := FALSE
                 END
END;


PROCEDURE TALLYC(VAR TAL : TALLY);
   (* TAKES COURSES IN LIST AND SENDS THEM TO FINDC TO BE COUNTED *)

BEGIN
  FOR J := 1 TO TOTC DO
     IF TAL[J,1] <> 0
       THEN FINDC(HEAD,TAL[J,1],TAL[J,2]);
END;

PROCEDURE INITTAL(VAR TAL: TALLY);
   (* INITIALIZE LIST TO ALL 0 *)

VAR  KEEP  : SC;                              (* HOLDS A COURSE AND A SECTION

BEGIN
  KEEP[1] := 0;
  KEEP[2] := 0;
  FOR J := 1 TO TOTC DO
    TAL[J] := KEEP;
END;

PROCEDURE PRINTTAL(TAL :TALLY);
   (* PRINTS LIST OF COURSES THAT A STUDENT SCHEDULED *)

VAR  I      : 1..TOTC;                        (* LOOP COUNTER *)

BEGIN
 REWRITE(OUTDEVICE,'PRINTER:');
 WRITELN(OUTDEVICE);
 WRITELN(OUTDEVICE,'COURSES SCHEDULED ');
 WRITELN(OUTDEVICE);
 FOR I := 1 TO TOTC DO
   IF TAL[I,1] > 0
    THEN WRITELN(OUTDEVICE,TAL[I,1],'    ',TAL[I,2]);
 CLOSE(OUTDEVICE);
END;


(*****************************************************************************)

PROCEDURE PRINTSTSCH( VAR S1,S2 : SCH);
    (* PRINTS CONFLICT MESSEGES WITH STUDENT SCHEDULES *)

BEGIN
  IF (CRNSCH1 > 0) AND (CRNSCH2 > 0)
   THEN
    BEGIN
     IRR := IRR + 1;
     IF (RUN  = 'F') OR (RUN = 'T')
       THEN
        BEGIN
         REWRITE(OUTDEVICE,'PRINTER:');
         WRITELN(OUTDEVICE);
         WRITELN(OUTDEVICE,NAME);
         WRITELN(OUTDEVICE);
         WRITE(OUTDEVICE,'IRRESOLIBLE CONFLICT - COURSES ',CRNSCH1);
         WRITELN(OUTDEVICE, ' AND ', CRNSCH2, ' ARE PROBLEMS');
         CLOSE(OUTDEVICE);
         STDSCH(S1,S2);
        END;
```

```
  DAYS   : STRING[5];                    (* DAYS OF WEEK *)
  D      : INTEGER;                      (* INDEX INTO DAYS OF WEEK *)
  J      : WEEK;                         (* LOOP COUNTER - WEEK *)
  HEAD   : PTR;                          (* HEAD OF LIST *)

PROCEDURE FILLMAT( VAR SB: SCH);
   (* PLACES COURSE IN SCHEDULE MATRIX BY SEM, PD AND DAY *)

BEGIN
D := 1;
FOR J := M TO F DO
  BEGIN
    IF NOT CONFLICT THEN
     BEGIN
      IF NODE^.DAY[D] = DAYS[D]
       THEN IF SB[NODE^.PD,D] = Ø
              THEN
               BEGIN
                SB[NODE^.PD,D] := CRS;
                OK := TRUE;
                CONFLICT := FALSE;
               END
              ELSE
               BEGIN
                OK := FALSE;
                CONFLICT := TRUE;
                PROBCRS := SB[NODE^.PD,D];
               END;
      D := D + 1;
     END
   END

END;


BEGIN                                    (* SCHEDULECRS *)
 DAYS := 'MTWRF';
 CONFLICT := FALSE;
 NEXT := TRUE;
 HEAD := CHEAD;
 WHILE NEXT AND NOT CONFLICT DO
  BEGIN
   NEW(NODE);
   QUERYCL(CRS,SEC,NODE,HEAD,CTAIL,FOUND,NEXT);
   IF FOUND
    THEN
     BEGIN
       IF (NODE^.SEM < 1 ) OR (NODE^.SEM > 3)
         THEN WRITELN('ERROR IN SEMESTER ')
         ELSE
           BEGIN
             CASE NODE^.SEM OF
              1 : FILLMAT(SB1);
              2 : FILLMAT(SB2);
              3 : BEGIN
                    FILLMAT(SB1);
                    FILLMAT(SB2)
                  END
             END
           END;
     END
    ELSE
     BEGIN
      REWRITE(OUTDEVICE,'PRINTER:');
      WRITELN(OUTDEVICE,'COURSE  ',CRS,' SECTION ',SEC,' IS NOT IN LIST');
      CLOSE(OUTDEVICE)
```

```
      END;
  END
END;                                          (* SCHEDULECRS *)

(*************************************************************************)

PROCEDURE ADDPROBLIST( VAR LIST : COURREQLIST; PROBCRS : INTEGER):
    (* ADDS A COURSE TO PROBLEM LIST SO OTHER SECTIONS CAN BE
       SCHEDULED IF IT BECOMES NECESSARY *)

VAR    INDEX    : 1..MAXN;                    (* LIST INDEX *)
       FLAG     : BOOLEAN;                    (* COURSE IS IN LIST *)

BEGIN
  IF LIST = NULLA
    THEN LIST[1] := PROBCRS
    ELSE
     BEGIN
      FLAG := TRUE;
      FOR INDEX := 1 TO MAXN DO
        IF LIST[INDEX] = PROBCRS
          THEN FLAG := FALSE;
      IF FLAG
        THEN
         BEGIN
          INDEX := 2;
          WHILE FLAG DO
           BEGIN
            IF LIST[INDEX] = 0
             THEN
              BEGIN
               LIST[INDEX] := PROBCRS;
               FLAG := FALSE
              END;
            IF INDEX < 10
             THEN INDEX := INDEX + 1
             ELSE FLAG := FALSE
           END
         END
     END
END;

PROCEDURE TRYALT(VAR ATAB : TABLE);
    (* TRIES TO SCHEDULE ALTERNATE COURSES *)

PROCEDURE ADDALT(VAR NODE : POINT1; FALTCRS,RALTCRS : INTEGER);
    (* CREATES NEW CLIST USING THE ALTERNATE COURSE *)

 VAR
   CH,                                        (* PTR TO HEAD OF CLIST *)
   CT.                                        (* PTR TO TAIL OF CLIST *)
   LOOKAHEAD.                                 (* CRS LIST - LOOKAHEAD *)
   CHASER       : PTR;                        (*          - CHASER    *)

BEGIN
  FOR INDEX := 1 TO MAXN DO
    IF NODE^.CRSREQ[INDEX] = FALTCRS
      THEN NODE^.CRSREQ[INDEX] := RALTCRS;
   CH := CHEAD;
   CT := CTAIL;
   LOOKAHEAD := HEAD;
   CHASER    := HEAD;
   WHILE (LOOKAHEAD <> NIL) AND (RALTCRS > LOOKAHEAD^.CRNUM) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK
```

```
    END;
   WHILE (LOOKAHEAD <> NIL) AND (RALTCRS = LOOKAHEAD^.CRNUM) DO
    BEGIN
     IF LOOKAHEAD^.ENROL < LOOKAHEAD^.MAX
      THEN
       BEGIN
        NEW(CORNODE);
        CORNODE^ := LOOKAHEAD^;
        CORNODE^.LINK := CH;
        CT^.LINK := CORNODE;
        CH := CORNODE
       END;
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK
    END
END;

BEGIN
  IF ATAB[1,1] = ALT1
   THEN
    BEGIN
     NEXTCRS(ATAB);
     REPLACE(ALT2,HEAD,CTAIL,ATAB);
     IF NOT ALTUSED
       THEN
        BEGIN
         ADDALT(STNODE,ALT1,ALT2);
         ALTUSED := TRUE
        END
    END
   ELSE IF ATAB[1,1] = ALT2
         THEN
          BEGIN
           NEXTCRS(ATAB);
           REPLACE(ALT1,HEAD,CTAIL,ATAB);
           IF NOT ALTUSED
             THEN
              BEGIN
               ADDALT(STNODE,ALT2,ALT1);
               ALTUSED := TRUE
              END
          END;
END;

PROCEDURE TRYPROBLIST(VAR TAB:TABLE;VAR PROBLIST:COURREQLIST;VAR TAL:TALLY;
                      VAR S1,S2 : SCH);
   (* ADDS ALL SECTIONS OF COURSES ON PROBLEM LIST TO THE TABLE AND TRIES
      TO SCHEDULE THEM *)

VAR
  STAB,                                  (* TABLE FOR OTHEROPT           *)
  BTAB,                                  (* TABLE FOR BACKUP             *)
  CTAB,                                  (* COPY OF TAB FOR REPLACEMENT  *)
  NTAB          : TABLE;                 (* COPY OF TAB          *)
  BS1,                                   (* SCHED FOR BACKUP     *)
  BS2,                                   (* SCHED FOR BACKUP     *)
  SB1,                                   (* BACKUP COPY OF NS1   *)
  SB2,                                   (* BACKUP COPY OF NS2   *)
  SS1,                                   (* SCH FOR OTHEROPT     *)
  SS2,                                   (* SCH FOR OTHEROPT     *)
  NS1,                                   (* COPY OF S1           *)
  NS2           : SCH;                   (* COPY OF S2           *)
  LIST          : COURREQLIST;           (* COPY OF PROBLIST     *)
  STAL,                                  (* TAL FOR OTHEROPT     *)
  NTAL          : TALLY;                 (* COPY OF TALLY        *)
  PROBCRS,                               (* COURSE CAUSING CONF      *)
```

```
  RESTORE        : INTEGER;
  SINDEX,
  INDEX          : 0..MAXN;          (* INDEX FOR OTHEROPT       *)
  ALTOFF,                            (* LIST INDEX               *)
  OTHEROPT,                          (* ALT IS TURNED OFF        *)
  BACKUP,                            (* TRY OTHER OPTION         *)
  REM,                               (* TRY REST OF SECTIONS     *)
  OK,                                (* MARKS REMOVAL OF A CRS   *)
  CONFLICT,                          (* SCHED OK          *)
  MARKER,                            (* SCHED WITH CONFLICT *)
  NOCONFLICT     : BOOLEAN;          (* MARKS PREV CONFLICT *)
                                     (* RECORDS CONFLICT    *)

PROCEDURE SAVEENVIROMENT;
   (* SAVES THE VALUES IN CASE A SCHEDULE TRY DOESN'T WORK *)

BEGIN
  CTAB := TAB;
  NEXTCRS(CTAB);
  NTAB := CTAB;
  NS1  := S1;
  NS2  := S2;
  LIST := PROBLIST;
  NTAL := TAL;
  RESTORE := TAB[1,1];
  NOCONFLICT := TRUE
END;

PROCEDURE RESTOREENVIROMENT;
   (* SCHEDULE WAS SUCESSFUL DO RESTORE ENVIROMENT BEFORE RETURNING *)

BEGIN
  TAB[1,1] := 0;
  S1 := NS1;
  S2 := NS2;
  TAL :=NTAL
END;


PROCEDURE CRSSCHEDULED;
   (* COURSE WAS SCHEDULED SO UPDATE SCHDEDULES *)

BEGIN
 FOR RO := 1 TO NUMPD DO
  FOR CO := 1 TO NUMD DO
   IF NS1[RO,CO] = NTAB[1,1]
    THEN
     BEGIN
      SB1[RO,CO] := 0;
      REM := TRUE
     END;
 FOR RO := 1 TO NUMPD DO
  FOR CO := 1 TO NUMD DO
   IF NS2[RO,CO] = NTAB[1,1]
    THEN
     BEGIN
      SB2[RO,CO] := 0;
      REM := TRUE
     END;
 IF REM
  THEN REMOVETL(NTAL,NTAB[1,1]);
 TALLYLIST(NTAL,NTAB[1,1],NTAB[1,2]);
 IF (NTAB[1,1] = NTAB[2,1]) AND (NOT BACKUP)
  THEN
   BEGIN
    BACKUP := TRUE;
    NEXTSEC(NTAB);
```

```
    BTAB := NTAB;
    BS1 := NS1;
    BS2 := NS2
   END;
 NS1 := SB1;
 NS2 := SB2;
 NEXTCRS(NTAB);
 IF NOT OTHEROPT
  THEN
   BEGIN
    STAB := NTAB;
    SS1  := NS1;
    SS2  := NS2;
    STAL := NTAL;
    OTHEROPT := TRUE
   END
END;

PROCEDURE CHECKT(CRSNUM : COURSENUMBER; VAR TAB : TABLE);
   (* IF CRS IS IN TABLE THEN IT IS REMOVED *)

VAR  J,                                        (* LOOP COUNTER *)
     C      : 1..MAXROWS;                      (* LOOP COUNTER *)

BEGIN
 FOR J := 1 TO MAXROWS DO
   IF TAB[J,1] = CRSNUM THEN TAB[J,1] := 0;
 FOR J := 1 TO MAXROWS DO
   IF TAB[J,2] > 0 THEN IF TAB[J,1] = 0
                          THEN FOR C:= J TO MAXROWS - 1 DO
                                 TAB[C] := TAB[C+1];
END;


BEGIN                                          (* TRYPROBLIST *)
 SAVEENVIROMENT;
 REPLACE(RESTORE,CHEAD,CTAIL,NTAB);
 INDEX := 1;
 SINDEX := 0;
 OTHEROPT := FALSE;
 BACKUP := FALSE;
 WHILE LIST[INDEX] <> 0 DO
  BEGIN
   ALTOFF := TRUE;
   CHECKT(LIST[INDEX],NTAB);
   REPLACE(LIST[INDEX],CHEAD,CTAIL,NTAB);
   WHILE (NTAB[1,1] <> 0) AND (NOCONFLICT) DO
    BEGIN
     SB1 := NS1;
     SB2 := NS2;
     SCHEDULECRS(NTAB[1,1],NTAB[1,2],OK,CONFLICT,MARKER,PROBCRS,SB1,SB2),
     REM := FALSE;
     IF OK
      THEN CRSSCHEDULED;
      IF CONFLICT
       THEN
        BEGIN
         IF SECT(PROBCRS,CHEAD,CTAIL) > 1
          THEN ADDPROBLIST(LIST,PROBCRS);
         IF NTAB[2,1] = NTAB[1,1]
           THEN
            BEGIN
             SB1 := NS1;
             SB2 := NS2;
             NEXTSEC(NTAB)
            END
```

```
            ELSE IF ((NTAB[1,1] = ALT1) OR (NTAB[1,1] = ALT2)) AND ALTOFF
                   THEN
                    BEGIN
                     TRYALT(NTAB);
                     ALTOFF := FALSE
                    END
                   ELSE
                    BEGIN
                     IF BACKUP
                      THEN
                       BEGIN
                        BACKUP := FALSE;
                        NTAB  := BTAB;
                        NS1   := BS1;
                        NS2   := BS2
                       END
                      ELSE
                       BEGIN
                        NTAB[1,1] := 0;
                        NOCONFLICT := FALSE
                       END;
                    END
        END
     END;
    IF NOCONFLICT
     THEN
      BEGIN
       RESTOREENVIROMENT;
       LIST[INDEX + 1] := 0
      END
     ELSE
      BEGIN
       NOCONFLICT := TRUE;
       IF OTHEROPT
        THEN
         BEGIN
          NS1 := SS1;
          NS2 := SS2;
          NTAL := STAL;
          NTAB := STAB;
          OTHEROPT := FALSE;
         END
        ELSE
         BEGIN
          NTAB  := CTAB;
          RESTORE := TAB[1,1];
          REPLACE(RESTORE,CHEAD,CTAIL,NTAB);
          NS1 := S1;
          NS2 := S2;
         END;
       END;
    IF INDEX < 10
      THEN INDEX := INDEX + 1
      ELSE LIST[INDEX] := 0;
   END
END;                                    (* TRYPROBLIST *)

(*****************************************************************)

BEGIN                                   (* MAKESCHEDULE *)

 OK := TRUE;
 MARKER := FALSE;
 ALTUSED := FALSE;
 INITSCH(S1);
```

```
SB1 := S1;
INITSCH(S2);
SB2 := S2;
NUMBERCONF := 0;
INITTAL(TAL);
PROBLIST := NULLA;
FINALCHANCE := FALSE;
WHILE TAB[1,1] <> 0 DO                    (* WHILE MORE COURSES TO SCHEDULE *)
 BEGIN
  IF OK THEN
   BEGIN
    PROB := FALSE;
    BESTBALNUM := TAB[1,1];
    BESTBALSEC := TAB[1,2]
   END;
  SCHEDULECRS(TAB[1,1],TAB[1,2],OK,CONFLICT,MARKER,PROBCRS,SB1,SB2);
  IF OK
   THEN
    BEGIN
     S1 := SB1;
     S2 := SB2;
     TALLYLIST(TAL,TAB[1,1],TAB[1,2]);
     NEXTCRS(TAB);
     MARKER := FALSE;
     PROBLIST := NULLA;
     FINALCHANCE := FALSE;
    END;
  IF CONFLICT
   THEN
    BEGIN
     SB1 := S1;
     SB2 := S2;
     IF SECT(PROBCRS,CHEAD,CTAIL) > 1
      THEN
       BEGIN
        ADDPROBLIST(PROBLIST,PROBCRS);
        FINALCHANCE := TRUE
       END;
     IF TAB[2,1] = TAB[1,1]
      THEN                                (* TRY OTHER SECTIONS *)
       BEGIN
        IF NOT PROB
         THEN
          BEGIN
           REMOVE := PROBCRS;
           PROB := TRUE;
          END;
        NEXTSEC(TAB)
       END
      ELSE IF MARKER                      (* NO OTHER SECTION *)
            THEN
             BEGIN
              MARKER := FALSE;
              NUMBERCONF := NUMBERCONF + 1;
              IF NUMBERCONF = 1 THEN CRNSCH1 := TAB[1,1];
              IF NUMBERCONF = 2
               THEN
                BEGIN
                 CRNSCH2 := TAB[1,1];
                 TAB[1,1] := 0;
                END
               ELSE NEXTCRS(TAB);
             END
            ELSE
             BEGIN
```

```
                IF FINALCHANCE
                 THEN TRYPROBLIST(TAB,PROBLIST,TAL,S1,S2);
                IF TAB[1,1] <> 0
                 THEN
                  BEGIN
                   IF (TAB[1,1] = ALT1) OR (TAB[1,1] = ALT2) AND ALTOF
                    THEN
                     BEGIN
                      TRYALT(TAB);
                      ALTOF := FALSE
                     END
                    ELSE
                     BEGIN
                      IF NOT PROB
                       THEN
                        BEGIN
                         REMOVE := PROBCRS;
                         PROB := TRUE
                        END;
                      REMOVETL(TAL,REMOVE);
                      FOR RO := 1 TO NUMPD DO
                       FOR CO := 1 TO NUMD DO
                        IF SB1[RO,CO] = REMOVE
                           THEN SB1[RO,CO] := 0;
                      FOR RO := 1 TO NUMPD DO
                       FOR CO := 1 TO NUMD DO
                        IF SB2[RO,CO] = REMOVE
                           THEN SB2[RO,CO] := 0;
                      NEXTCRS(TAB);

                      SCHEDULECRS(BESTBALNUM,BESTBALSEC,OK,CONFLICT,MARKER,PROBCR
S,SB1,SB2);
                      IF NOT OK
                       THEN
                        BEGIN
                         SB1 := S1;
                         SB2 := S2;
                         NUMBERCONF := NUMBERCONF + 1;
                         IF NUMBERCONF = 1
                           THEN CRNSCH1 := BESTBALNUM
                           ELSE
                            BEGIN
                             CRNSCH2 := BESTBALNUM;
                             TAB[1,1] := 0
                            END
                        END
                       ELSE
                        BEGIN
                         MARKER := TRUE;
                         S1 := SB1;
                         S2 := SB2;
                         TALLYLIST(TAL,BESTBALNUM,BESTBALSEC);
                         IF SECT(REMOVE,CHEAD,CTAIL) < 2
                           THEN
                            BEGIN
                             NUMBERCONF := NUMBERCONF + 1;
                             IF NUMBERCONF = 1
                               THEN CRNSCH1 := REMOVE
                               ELSE IF NUMBERCONF = 2
                                      THEN
                                       BEGIN
                                        CRNSCH2 := REMOVE;
                                        TAB[1,1] := 0
                                       END
                            END
                           ELSE REPLACE(REMOVE,CHEAD,CTAIL,TAB);
```

```
                        END;
                      END;
                    END
                END
      END
  END;
  PRINTSTSCH(S1,S2);
END;                                        (* MAKESCHEDULE *)

(*************************************************************************)
(*************************************************************************)
PROCEDURE SCHEDASTUD;
  (* SCHEDULES ONE STUDENT *)

BEGIN
  CRNSCH1 := 0;
  CRNSCH2 := 0;
  ALTOF := TRUE;
  NEW(SCHNODE);
  MARK(HEAP);
  NEW(STNODE);
  STNODE^ := STREC^;                              (* GET STUDENT RECORD *)
  MAKECLIST(STNODE);
  IF STNODE^.CRSREQ <> NULLA
   THEN
    BEGIN                                         (* SCHEDULE THE STUDENT *)
     MAKETABLE(T,COUNT,CHEAD,CTAIL);
     MAKESCHEDULE(T,S1,S2,TAL,STNODE^.STUDNAM);
     FILEST(SHEAD,SCHNODE,STNODE,TAL)
    END;
  TEST := TEST + 1;
  RELEASE(HEAP);
  IF (TEST = 100) AND NOTOPEN AND (RUN = 'F')
   THEN
    BEGIN                                         (* SAVE SCH IF FINAL RUN *)
     STR(PRESENT,S);
     FNAME := CONCAT('SCH:SCH',S);
     OPENFILE(FNAME);
     IF SHEAD <> NIL
      THEN SAVESCH(SHEAD);
     NOTOPEN := FALSE;
     RELEASE(HEAP1);
     SHEAD := NIL;
     TEST := 1;
    END;
  GET(STREC);

END;


BEGIN                                           (* STUDSCHEDULE  *)


  FOR INDEX := 1 TO MAXN DO
    NULLA[INDEX] := 0;
  GOTOXY(10,10);
  MESS('TRIAL OR FINAL RUN ? (ENTER T OR F):');
  READ(RUN);
  IF RUN = 'T'
   THEN
    BEGIN
     MESS('WHAT RUN IS THIS : ');
     READLN(RUNTYPE)
    END
   ELSE RUNTYPE := 'FINAL';
  PAGE(OUTPUT);
```

```
  LOW := ASKCHOICE(0,5,'WHAT IS THE LOWEST GRADE YOU WISH TO SCHEDULE',7.12);
  HIGH := ASKCHOICE(0,10,'WHAT IS THE HIGHEST GRADE YOU WISH TO SCHEDULE',7.12);

  GETOUT := FALSE;
  PRESENT := LOW;
  REPEAT
    TEST := 1;
    FULL := 0;
    PART := 0;
    IRR  := 0;
    NOTOPEN := TRUE;
    MARK(HEAP1);
    SHEAD := NIL;
    INITSCH(SH1);
    INITSCH(SH2);
    STR(PRESENT,FILEGRADE);
    FILENAME := CONCAT('ST:STREC',FILEGRADE);
    RESET(STREC,FILENAME);
    WHILE NOT EOF (STREC)   DO
      BEGIN
       SCHEDASTUD
      END;
    CLOSE(STREC);
    REWRITE(OUTDEVICE,'PRINTER:');
    WRITELN(OUTDEVICE);
    WRITELN(OUTDEVICE,'      RUN - ',RUNTYPE);
    WRITELN(OUTDEVICE,'RESULTS FOR GRADE ',PRESENT);
    WRITELN(OUTDEVICE,'       FULL SCHEDULES     - ',FULL);
    WRITELN(OUTDEVICE,'       PARTIAL SCHEDULES - ',PART);
    WRITELN(OUTDEVICE,'       IRR CONFLICTS      - ',IRR);
    WRITELN(OUTDEVICE,'       TOTAL              - ',FULL + PART + IRR);
    WRITELN(OUTDEVICE);
    WRITELN(OUTDEVICE,' STUDY HALL COUNT ');
    CLOSE(OUTDEVICE);
    STDSCH(SH1,SH2);
    IF NOTOPEN AND (RUN = 'F')
     THEN
      BEGIN
       STR(PRESENT,S);
       FNAME := CONCAT('SCH:SCH',S);
       OPENFILE(FNAME);
       IF SHEAD <> NIL
         THEN SAVESCH(SHEAD);
      END
     ELSE
      IF (SHEAD <> NIL) AND (RUN = 'F')
       THEN SAVESCH(SHEAD);
    IF RUN = 'F'
     THEN CLOSE(SCHRECD,LOCK);
    IF PRESENT = HIGH
      THEN GETOUT := TRUE
      ELSE PRESENT := PRESENT + 1;
  RELEASE(HEAP1);
  UNTIL GETOUT = TRUE

END;                                        (* STUDSCHEDULE *)


  BEGIN                                     (* SCHEDULER *)
    STUDSCHEDULE;
    PRINTLIST(HEAD);

  END.                                      (* SCHEDULER *)
```

10.7. UNIT TWO

```
(*$S+*)                                        (* ALLOW SWAPPING *)

UNIT TWO; INTRINSIC CODE 23 DATA 24;


  (*************************************************************************)
  (*                                                                       *)
  (*      THIS UNIT CONTAINS SUBROUTINES USED BY 'SCHED'. WHEN CALLED      *)
  (*           THIS UNIT LOADS ALL COURSES INTO A LIST AND SETS CLASS      *)
  (*           ENROLLMENT TO 0.                                            *)
  (*                                                                       *)
  (*************************************************************************)


INTERFACE

CONST
  MAXN                = 10;                (* MAX NUMBER IN COURSES *)
  MAXCOLS             =  4;                (* MAX COLUMNS IN TABLE  *)
  MAXROWS             = 75;                (* MAX ROWS IN TABLE     *)
  NUMPD               = 13;                (* PERIODS IN SCHEDULE   *)
  NUMD                =  5;                (* DAYS IN SCHEDULE      *)
  TOTC                = 11;                (* TOTAL COURSES         *)

TYPE
  POINTER             = ^COURSE;
  PTR                 = ^CRENROL;
  COURSENUMBER        = 0..999;
  COURSESECTION       = 1..99;
  SEMESTER            = 1..3;
  PERIODNUMBER        = 1..16;
  TEACHERNUMBER       = 1..99;
  COURSE              =
    RECORD
      CRNUM     : COURSENUMBER;          (* COURSE NUMBER (0..999)  *)
      CRNAM     : STRING[10];            (* COURSE NAME ( 10 CHAR)  *)
      CRSEC     : COURSESECTION;         (* COURSE SECTION (1..99)  *)
      MAX       : INTEGER;               (* MAX SIZE OF CLASS       *)
      PERIOD    : PERIODNUMBER;          (* PERIOD OF DAY (1..16    *)
      PD        : PERIODNUMBER;          (* PERIOD FOR SCHEDULING   *)
      DAY       : STRING[5];             (* DAY OF WEEK (MTWRF)     *)
      SEM       : SEMESTER;              (* SEMESTER  (1..2..3)     *)
      ROOM      : STRING[3];             (* ROOM NUMBER             *)
      TEACHER   : TEACHERNUMBER;         (* TEACHER NUMBER          *)
      LINK1     : POINTER;               (* POINTER TO NEXT RECORD  *)
    END;

  CRENROL             =
    RECORD
      CRNUM     : COURSENUMBER;          (* COURSE NUMBER (0..999)  *)
      CRNAM     : STRING[10];            (* COURSE NAME ( 10 CHAR)  *)
      CRSEC     : COURSESECTION;         (* COURSE SECTION (1..99)  *)
      MAX       : INTEGER;               (* MAX SIZE OF CLASS       *)
      PERIOD    : PERIODNUMBER;          (* PERIOD OF DAY (1..16    *)
      PD        : PERIODNUMBER;          (* PERIOD FOR SCHEDULING   *)
      DAY       : STRING[5];             (* DAY OF WEEK (MTWRF)     *)
```

```
      SEM       : SEMESTER;                (* SEMESTER  (1..2..3)     *)
      ROOM      : STRING[3];               (* ROOM NUMBER             *)
      TEACHER   : TEACHERNUMBER;           (* TEACHER NUMBER          *)
      ENROL     : INTEGER;                 (* COURSE ENROLMENT        *)
      LINK      : PTR;                     (* POINTER TO NEXT RECORD *)
    END;
  POINT1              = ^STUDENT;         (* TO STUDENT RECORD *)
  YEAR                = 7..12;
  COURREQLIST         = ARRAY[1..MAXN] OF COURSENUMBER;
  STUDENT             =
    RECORD
      STUDNAM         : STRING[20];       (* STUDENT NAME *)
      STUDID          : INTEGER;          (* STUDENT ID NUMBER *)
      GRADE           : YEAR;             (* GRADE IN SCHOOL *)
      CRSREQ          : COURREQLIST;      (* ARRAY OF STUDENT COURSES *)
      LINK1           : POINT1;           (* POINTER TO NEXT RECORD *)
    END;
  ONER                = ARRAY[1..NUMD] OF INTEGER;
  SCH                 = ARRAY[1..NUMPD] OF ONER;
  ONECOLUMN           = ARRAY[1..NUMPD] OF INTEGER;
  ONEROW              = ARRAY[1..MAXCOLS] OF INTEGER;
  TABLE               = ARRAY[1..MAXROWS] OF ONEROW;
  ONECOL              = ARRAY[1..MAXROWS] OF INTEGER;
  SC                  = ARRAY[1..2] OF INTEGER;   (* CRS NUMBER AND SEC *)
  TALLY               = ARRAY[1..TOTC] OF SC;     (* LIST OF CRS SCHED  *)
  SCHPTR              = ^SCHREC;                  (* PTR TO SCH RECORDS *)
  SCHREC              =
    RECORD
      STUDNAM : STRING[20];             (* STUDENT NAME *)
      STUDID  : INTEGER;                (* STUDENT ID   *)
      GRADE   : YEAR;                   (* STUDENT GRADE*)
      CRSSCH  : TALLY;                  (* STUD SCHEDULE*)
      L       : SCHPTR;                 (* POINTER      *)
    END;

VAR
  HEAD       : PTR;                      (* POINTER TO START OF LIST *)
  CRREC      : FILE OF COURSE;           (* FILE BUFFER              *)
  SCHRECD    : FILE OF SCHREC;           (* FILE OF STUDENT SCHEDULES *)
  OUTDEVICE  : TEXT;                     (* PRINTER                  *)
  RO         : 1..NUMPD;                 (* ROWS - PD                *)
  CO         : 1..NUMD;                  (* COLUMNS - DAYS           *)
  CORNODE    : PTR;                      (* COURSE RECORD            *)
  INDEX      : 0..MAXN;                  (* ARRAY INDEX              *)
  NULLA      : COURREQLIST;              (* ALL COURSES ARE 0        *)
  CHEAD,                                 (* HEAD OF CIRCULAR LIST    *)
  CTAIL      : PTR;                      (* TAIL OF CIRCULAR LIST    *)
  COUNT      : INTEGER;                  (* NUMBER OF COURSES IN C.L.*)
  ROW        : 1..MAXROWS;               (* INDEX FOR TABLE          *)
  HOLD       : ONEROW;                   (* TEMP STORAGE             *)
  COL        : 1..MAXCOLS;               (* A COLUMN                 *)


PROCEDURE OPENFILE(FILENAME : STRING);
PROCEDURE SAVESCH(FRONT : SCHPTR);
PROCEDURE INSERTSCH(VAR FRONT, ITEM :SCHPTR);
PROCEDURE FILEST(VAR SHEAD, SCHNODE :SCHPTR; NODE : POINT1; TAL :TALLY);
PROCEDURE IOPRINT(ERRTYPE:INTEGER);
PROCEDURE NODISK(DISKNAME:STRING; DISKERR : INTEGER);
PROCEDURE INSERTNODE (VAR FRONT, ITEM : PTR);
PROCEDURE FETCHIT(VAR FIRST : PTR);
PROCEDURE PRINTLIST (FRONT : PTR);
PROCEDURE INITSCH(VAR S : SCH);
PROCEDURE PRINTSCH(S:SCH);
PROCEDURE MESS(STR:STRING);
PROCEDURE STDSCH(VAR S1,S2 : SCH);
```

```
PROCEDURE FETCHCR(NODE :POINT1; VAR CH,CT,HD :PTR);
PROCEDURE MAKECLIST( VAR NODE :POINT1);
FUNCTION ASKCHOICE(X,Y :INTEGER; Q :STRING; MIN,MAX :INTEGER) : INTEGER;
PROCEDURE PRINTMAT(MAT : TABLE; ST : STRING);
PROCEDURE MAKETABLE(VAR M: TABLE; ROWCOUNT : INTEGER; VAR CH,CT : PTR);
FUNCTION SECT(REMOVE : INTEGER; CH,CT : PTR) : INTEGER;
PROCEDURE REPLACE(REMOVE : INTEGER; CH,CT : PTR; VAR TAB : TABLE);
PROCEDURE FINDC(FRONT:PTR; CRS, SEC : INTEGER);
PROCEDURE SHCOUNT(VAR SH1,SH2 : SCH; S1,S2 :SCH);
PROCEDURE NEXTCRS(VAR TAB : TABLE);
PROCEDURE NEXTSEC(VAR TAB : TABLE);


IMPLEMENTATION



PROCEDURE OPENFILE;
   (* OPENS A FILE AND CHECKS FOR ERRORS *)

VAR IOERR    : INTEGER;                    (* ERROR NUMBER *)

BEGIN
 REPEAT
   (*$I-*)
   REWRITE(SCHRECD,FILENAME);
   (*$I+*)
   IOERR := IORESULT;
   NODISK(FILENAME,IOERR);
 UNTIL IOERR = 0;
END;

PROCEDURE SAVESCH;
     (* SAVES SCHEDULE RECORDS FOR A GRADE LEVEL TO A FILE *)


BEGIN
 WHILE FRONT <> NIL DO
   BEGIN
    SCHRECD^ := FRONT^;
    SCHRECD^.L := NIL;
    PUT(SCHRECD);
    FRONT := FRONT^.L
   END
END;

PROCEDURE INSERTSCH;

    (* INSERTS A STUDENT COURSE RECORD INTO LINKED LIST *)

VAR  LOOKAHEAD,                            (* POINTER TO FIND PLACE *)
     CHASER        : SCHPTR;               (* POINTER FOR INSERTION *)

BEGIN
  LOOKAHEAD := FRONT;
  CHASER    := FRONT;
  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.STUDNAM > LOOKAHEAD^.STUDNAM) DO
                                           (* TRAVERSE LIST TO INSERT POINT *)
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.L
    END;

  IF LOOKAHEAD <> NIL
    THEN                                   (* NOT AT END OF LIST *)
```

```
      IF LOOKAHEAD <> FRONT
        THEN
          BEGIN                                  (* INSERT IN MIDDLE OF LIST *)
            ITEM^.L := LOOKAHEAD;
            CHASER^.L := ITEM;
          END
        ELSE
          BEGIN                                  (* INSERT AT FRONT *)
            ITEM^.L := FRONT;
            FRONT := ITEM;
          END
      ELSE
        CHASER^.L := ITEM;


END;

PROCEDURE FILEST;
   (* SENDS STUDENT SCHEDULE RECORD TO FILE *)

BEGIN
 SCHNODE^.STUDNAM := NODE^.STUDNAM;
 SCHNODE^.STUDID  := NODE^.STUDID;
 SCHNODE^.GRADE   := NODE^.GRADE;
 SCHNODE^.CRSSCH  := TAL;
 SCHNODE^.L       := NIL;
 IF SHEAD = NIL
   THEN SHEAD := SCHNODE
   ELSE INSERTSCH(SHEAD,SCHNODE)
END;



PROCEDURE IOPRINT;
    (* PRINT ERROR MESSAGE *)

BEGIN
 GOTOXY(0,22);
 WRITE(CHR(7));
 CASE ERRTYPE OF
   1 : WRITELN('DISK READ ERROR.');
   2 : WRITELN('BAD UNIT NUMBER.');
   3 : WRITELN('ILLEGAL OPERATION.');
   4 : WRITELN('UNDEFINED HARDWARE ERROR.');
   5 : WRITELN('UNIT NO LONGER ON LINE.');
   6 : WRITELN('FILE NOT IN DIRECTORY.');
   7 : WRITELN('ILLEGAL FILE NAME.');
   8 : WRITELN('INSUFFICIENT DISK SPACE.');
   9 : WRITELN('NO SUCH VOLUME ON LINE.');
  10 : WRITELN('NO SUCH FILE ON VOLUME.');
  11 : WRITELN('DUPLICATE FILE.');
  12 : WRITELN('ATTEMPT TO OPEN AN OPEN FILE.');
  13 : WRITELN('FILE NOT OPEN.');
  14 : WRITELN('ERROR IN READING INTEGER OR REAL DATA.');
  15 : WRITELN('RING BUFFER OVERFLOW.');
 END;
END;

PROCEDURE NODISK;
    (* CHECKS FOR ERROR 9 *)

VAR CH : CHAR;

BEGIN
 IF (DISKERR <> 0) AND (DISKERR <> 9)
   THEN IOPRINT(DISKERR);
```

```
 IF (DISKERR = 9)
   THEN
    BEGIN
     GOTOXY(0,22);
     WRITELN('NO ',DISKNAME,' DISKETTE!');
     WRITELN('PLEASE INSERT DISKETTE AND HIT ANY KEY');
     READ(KEYBOARD,CH);
    END;
END;



PROCEDURE INSERTNODE;
   (* TO PUT NEW NODE IN LIST IN ORDER OF COURSE, SECTION AND PD NUMBER *)

  VAR  LOOKAHEAD,                          (* LOOKAHEAD POINTER TO FIND PLACE *)
       CHASER      : PTR;                  (* CHASER POINTER FOR INSERTION *)

 BEGIN (* INSERTNODE *)
  LOOKAHEAD := FRONT;
  CHASER := FRONT;

  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM > LOOKAHEAD^.CRNUM) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK
    END ;  (* WHILE *)

  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM = LOOKAHEAD^.CRNUM) AND
        (ITEM^.CRSEC > LOOKAHEAD^.CRSEC) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK
    END ;  (* WHILE *)

  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM = LOOKAHEAD^.CRNUM) AND
        (ITEM^.CRSEC = LOOKAHEAD^.CRSEC) AND (ITEM^.PD > LOOKAHEAD^.PD) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK
    END ;  (* WHILE *)


  IF (LOOKAHEAD <> NIL)
    THEN
      IF LOOKAHEAD <> FRONT
        THEN
          BEGIN
            ITEM^.LINK := LOOKAHEAD;
            CHASER^.LINK := ITEM;
          END                              (* INSERT IN MIDDLE OF LIST *)
        ELSE
          BEGIN
            ITEM^.LINK := FRONT;
            FRONT := ITEM;
          END                              (* INSERT AT FRONT *)
    ELSE
      CHASER^.LINK := ITEM;

 END;

PROCEDURE FETCHIT;
   (* RETRIEVES COURSE LIST FROM FILE AND ENTERS COURSES IN A
      LINKED LIST *)
```

```
VAR  CRS   : PTR;                          (* CRS NODE WITH ENROLLMENT *)

BEGIN
RESET(CRREC,'CR:COURSEFILE');

WHILE NOT EOF(CRREC) DO
 BEGIN

   NEW(CRS);
   CRS^.CRNUM     := CRREC^.CRNUM;
   CRS^.CRNAM     := CRREC^.CRNAM;
   CRS^.CRSEC     := CRREC^.CRSEC;
   CRS^.MAX       := CRREC^.MAX;
   CRS^.PERIOD    := CRREC^.PERIOD;
   CRS^.PD        := CRREC^.PD;
   CRS^.DAY       := CRREC^.DAY;
   CRS^.SEM       := CRREC^.SEM;
   CRS^.ROOM      := CRREC^.ROOM;
   CRS^.TEACHER   := CRREC^.TEACHER;
   CRS^.ENROL     := 0;
   CRS^.LINK      := NIL;
   IF FIRST = NIL
     THEN FIRST := CRS
     ELSE INSERTNODE(FIRST,CRS);
   GET(CRREC);
 END;
CLOSE(CRREC);
END;



 PROCEDURE PRINTLIST ;
    (* PRINTS COURSE LIST GIVEN THE POINTER TO START OF LIST *)


   BEGIN
     REWRITE(OUTDEVICE,'PRINTER:');
     PAGE (OUTDEVICE);
     IF FRONT = NIL
       THEN                              (* LIST IS EMPTY *)
         BEGIN
           WRITELN (OUTDEVICE,'LIST IS EMPTY.');
         END
       ELSE                              (* THERE ARE COURSES *)
         BEGIN
           WRITELN(OUTDEVICE,'COURSES IN NUMERICAL ORDER ARE : ');
           WRITELN(OUTDEVICE);
           WRITE(OUTDEVICE,'CODE  SEC  SEATS  SEM  PD  DAYS MET  ');
           WRITELN(OUTDEVICE,'DESCRIPTION  ROOM  TCH  PER  ENROL');
           WHILE FRONT <> NIL DO
             BEGIN
               WITH FRONT^ DO
               BEGIN
               WRITE(OUTDEVICE,CRNUM:3,'  ',CRSEC:2,'  ',MAX:3,'  ');
               WRITE(OUTDEVICE,SEM,'  ',PD:2,'  ',DAY:5,'  ',CRNAM:10,'  ');

               WRITELN(OUTDEVICE,ROOM:3,'  ',TEACHER:2,'  ',PERIOD:2,ENROL:7);

               END;
               FRONT := FRONT^.LINK;
             END;
           WRITELN(OUTDEVICE);WRITELN(OUTDEVICE);
         END;
      CLOSE(OUTDEVICE);
    END;
```

```
PROCEDURE INITSCH;
  (* INITIALIZES A SCHEDULE TO ALL 0 *)

BEGIN
  FOR RO := 1 TO NUMPD DO
    FOR CO := 1 TO NUMD DO
      S[RO,CO] := 0;
END;

(*************************************************************************)

PROCEDURE PRINTSCH;
   (* PRINTS THE STUDENT  SCHEDULE  *)


BEGIN
  REWRITE(OUTDEVICE,'PRINTER:');
  FOR RO := 1 TO NUMPD DO
    BEGIN
      WRITE(OUTDEVICE, RO );
      FOR CO := 1 TO NUMD DO
        WRITE(OUTDEVICE,S[RO,CO]:6);
      WRITELN(OUTDEVICE);
    END;
  WRITELN(OUTDEVICE);
  CLOSE(OUTDEVICE);
END;

PROCEDURE MESS;
   (* PRINTS A HEADER FOR SCHEDULE *)

BEGIN
  REWRITE(OUTDEVICE,'PRINTER:');
  WRITELN(OUTDEVICE);
  WRITELN(OUTDEVICE,STR);
  CLOSE(OUTDEVICE);
END;


PROCEDURE STDSCH;
   (* PRINTS STUDENTS SCHEDULE FOR 1ST AND 2ND SEM *)

BEGIN
  MESS('1ST SEM SCHEDULE');
  PRINTSCH(S1);
  MESS('2ND SEM SCHEDULE');
  PRINTSCH(S2);
END;

 (*************************************************************************)


PROCEDURE FETCHCR;
    (* GET ALL POSSIBLE SECTIONS FOR A STUDENTS COURSE SELECTION AND PLACE
      THEM IN A CIRCULUR LIST *)

VAR
  LOOKAHEAD,                              (* CRS NODE   - LOOKAHEAD PTR *)
  CHASER          : PTR;                  (*            - CHASER        *)
  COURSEFOUND     : BOOLEAN;              (* CRS IS FOUND               *)
  NAME            : STRING[20];           (* STUDENT    - NAME          *)
  LIST            : COURREQLIST;          (*            - CRS LIST      *)
```

```
BEGIN
  NAME := NODE^.STUDNAM;
  LIST := NODE^.CRSREQ;
  FOR INDEX := 1 TO MAXN DO
    BEGIN
      IF LIST[INDEX] <> 0 THEN
      BEGIN
       LOOKAHEAD := HD;
       CHASER    := HD;
       COURSEFOUND := FALSE;
       WHILE (LOOKAHEAD <> NIL) AND (LIST[INDEX] > LOOKAHEAD^.CRNUM) DO
        BEGIN
          CHASER := LOOKAHEAD;
          LOOKAHEAD := LOOKAHEAD^.LINK
        END;
       WHILE (LOOKAHEAD <> NIL) AND (LIST[INDEX] = LOOKAHEAD^.CRNUM) DO
        BEGIN
         IF LOOKAHEAD^.ENROL < LOOKAHEAD^.MAX
          THEN
           BEGIN
            NEW(CORNODE);
            CORNODE^ := LOOKAHEAD^;
            COURSEFOUND := TRUE;
            COUNT := COUNT + 1;
            IF (CH = NIL) AND (CT = NIL)
              THEN
                BEGIN
                  CORNODE^.LINK := CORNODE;
                  CH := CORNODE;
                  CT := CORNODE
                END
              ELSE
                BEGIN
                  CORNODE^.LINK := CH;
                  CT^.LINK := CORNODE;
                  CH := CORNODE
                END;
           END;
          CHASER := LOOKAHEAD;
          LOOKAHEAD := LOOKAHEAD^.LINK
        END;
       IF ((LOOKAHEAD = NIL) OR (LIST[INDEX] < LOOKAHEAD^.CRNUM))
          AND (COURSEFOUND = FALSE)
          THEN
           BEGIN
             REWRITE(OUTDEVICE,'PRINTER:');
             WRITE(OUTDEVICE,NAME,' IS SCHEDULED FOR COURSE ',LIST[INDEX]);
             WRITELN(OUTDEVICE,' WHICH IS NOT IS COURSE LIST.');
             WRITELN(OUTDEVICE);
             CLOSE(OUTDEVICE)
           END;
      END
    END;
 WHILE ( LOOKAHEAD^.CRNUM < 999) DO
   BEGIN
     CHASER := LOOKAHEAD;
     LOOKAHEAD := LOOKAHEAD^.LINK
   END;
 WHILE ( LOOKAHEAD^.CRNUM = 999) DO
  BEGIN
    NEW(CORNODE);
    CORNODE^ := LOOKAHEAD^;
    COUNT := COUNT + 1;
    IF (CH = NIL) AND (CT = NIL)
      THEN
        BEGIN
```

```
            CORNODE^.LINK := CORNODE;
            CH := CORNODE;
            CT := CORNODE
          END
        ELSE
          BEGIN
            CORNODE^.LINK := CH;
            CT^.LINK := CORNODE;
            CH := CORNODE
          END;
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK
    END;


 END;



 PROCEDURE MAKECLIST;
    (* CREATES CLIST FOR A STUDENT COURSE SELECTIONS *)

 BEGIN
   IF NODE^.CRSREQ <> NULLA
     THEN
       BEGIN
         COUNT := 0;
         CHEAD := NIL;
         CTAIL := NIL;
         FETCHCR(NODE,CHEAD,CTAIL,HEAD)
       END
     ELSE
       BEGIN
         REWRITE(OUTDEVICE,'PRINTER:');
         WRITELN(OUTDEVICE,NODE^.STUDNAM,' IS SCHEDULED FOR NO COURSES');
         CLOSE(OUTDEVICE)
       END;
 END;


FUNCTION ASKCHOICE;
   (* ASK USER FOR A NUMBER FROM MIN TO MAX, AND REPEAT UNTIL OBTAINED *)

VAR   ANSWER      : INTEGER;

BEGIN
REPEAT
  GOTOXY(X,Y);
  WRITE(O,' ( ',MIN, '..',MAX,' ) : ');
  READLN(ANSWER);
UNTIL (ANSWER >= MIN) AND (ANSWER <= MAX);
ASKCHOICE := ANSWER
END;




PROCEDURE PRINTMAT;
   (* PRINTS LOOKUP TABLE *)

BEGIN
  REWRITE(OUTDEVICE,'PRINTER:');
  WRITELN(OUTDEVICE);
  WRITELN(OUTDEVICE);
  WRITELN(OUTDEVICE,'LOOKUP TABLE - ',ST);
```

```
  WRITELN(OUTDEVICE);
  WRITELN(OUTDEVICE,'CRNUM   SEC     BAL     PRIOR');

  FOR ROW := 1 TO MAXROWS DO
    BEGIN
      IF MAT[ROW,1] > 0 THEN
      BEGIN
      FOR COL := 1 TO MAXCOLS DO
        WRITE(OUTDEVICE,MAT[ROW,COL]:4,'    ');
      WRITELN(OUTDEVICE);
      END;
    END;
  WRITELN(OUTDEVICE);
  CLOSE(OUTDEVICE);
END;

(*************************************************************************)

PROCEDURE MAKETABLE;
   (* CREATES LOOKUP TABLE FOR CLIST *)

VAR
  BAL        : INTEGER;                  (* SEATS REMAINING IN A CLASS*)

PROCEDURE FILLTABLE(VAR LASTROW : INTEGER);
   (* FILLS LOOKUP TABLE WITH VALUES AND SETS PRIOR TO 0 *)

VAR PREVNUM   : COURSENUMBER;                    (* PREVIOUS COURSE *)
    PREVSEC   : COURSESECTION;                   (* PREVIOUS SECTION *)
    CHMARK    : PTR;                             (* MARK HEAD OF LIST *)

PROCEDURE INITTABLE;

BEGIN
  FOR ROW := 1 TO MAXROWS DO
    FOR COL := 1 TO MAXCOLS DO
      M[ROW,COL] := 0;
END;


BEGIN                                            (* FILLTABLE *)
  INITTABLE;
  ROW := 1;
  CHMARK := CH;
  REPEAT
        IF (PREVNUM = CH^.CRNUM) AND (PREVSEC = CH^.CRSEC)
          THEN
            CH := CH^.LINK
          ELSE
            BEGIN
              COL := 1;
              M[ROW,COL] := CH^.CRNUM;
              M[ROW,COL+1] := CH^.CRSEC;
              BAL := CH^.MAX - CH^.ENROL;
              M[ROW,COL+2] := BAL;
              M[ROW,COL+3] := 0;
              PREVNUM := CH^.CRNUM;
              PREVSEC := CH^.CRSEC;
              LASTROW := ROW;
              ROW := ROW + 1;
              CH := CH^.LINK
            END;
  UNTIL CH = CHMARK;
END;                                             (* FILLTABLE *)

PROCEDURE FINDPRIOR(LASTROW : INTEGER);
```

```
VAR PRIOR    : INTEGER;                          (* SCHEDULING PRIORITY *)
    TEMP,                                        (* REMEMBERS ROW *)
    INDEX    : 1..MAXROWS;                       (* LOOP COUNTER  *)

BEGIN
  FOR ROW := 1 TO LASTROW  DO
    BEGIN
      PRIOR := 1;
      IF M[ROW,1] < M[ROW+1,1]
        THEN
          M[ROW,4] := 1
        ELSE
          BEGIN
            TEMP := ROW;
            WHILE M[TEMP,1] = M[TEMP+1,1] DO
              BEGIN
                IF M[TEMP,2] <> M[TEMP+1,2]
                  THEN PRIOR := PRIOR +1;
                TEMP := TEMP + 1;
              END;
            FOR INDEX := ROW TO TEMP DO
              M[INDEX,4] := PRIOR;
            ROW := TEMP
          END;
    END;
END;

PROCEDURE SORTBYPRIOR(LIMIT : INTEGER);
   (* SORT TABLE BY PRIORITY (COLUMN 4) *)

CONST PRIMARY = 4;                               (* SORT KEY *)

VAR
   INORDER   : BOOLEAN;                          (* TRUE IS TABLE IN ORDER *)

BEGIN
  REPEAT
    INORDER := TRUE;
    IF LIMIT > 1 THEN LIMIT := LIMIT -1;
    FOR ROW := 1 TO LIMIT DO
      BEGIN
        IF M[ROW,PRIMARY] > M[ROW + 1,PRIMARY]
          THEN
            BEGIN                                (* SWAP *)
              INORDER := FALSE;
              HOLD := M[ROW];
              M[ROW] := M[ROW+1];
              M[ROW+1] := HOLD
            END;
      END;
  UNTIL INORDER;
END;

PROCEDURE SORTBYBAL(LIMIT : INTEGER);
   (* SORTS TABLE BY CLASS BALANCE *)

CONST PRIMARY = 3;                               (* BAL IS KEY *)

VAR INORDER : BOOLEAN;                           (* TRUE = TABLE IN ORDER *)

BEGIN
  REPEAT
    INORDER := TRUE;
    LIMIT := LIMIT - 1;
    FOR ROW := 1 TO LIMIT DO
```

```
        BEGIN
          IF (M[ROW,1] = M[ROW+1,1]) AND (M[ROW,PRIMARY] <  M[ROW+1,PRIMARY])
             THEN
               BEGIN
                 INORDER := FALSE;
                 HOLD := M[ROW];
                 M[ROW] := M[ROW+1];
                 M[ROW+1] := HOLD
               END;
        END;
  UNTIL INORDER;
END;

BEGIN                                          (* MAKETABLE *)
  FILLTABLE(ROWCOUNT);
  FINDPRIOR(ROWCOUNT);
  SORTBYPRIOR(ROWCOUNT);
  SORTBYBAL(ROWCOUNT);
END;                                           (* MAKETABLE *)


(*****************************************************************************)

FUNCTION SECT;
   (* FINDS THE NUMBER OF SECTIONS OF REMOVE *)

VAR PREVSEC,                                   (* PREVIOUS SECTION *)
    SEC           : INTEGER;                   (* NUMBER OF SECTIONS *)
    CHMARK        : PTR;                       (* MARKS HEAD OF LIST *)

BEGIN
 CHMARK := CH;
 SEC := 0;
 REPEAT
   IF REMOVE = CH^.CRNUM
     THEN
       BEGIN
         SEC := SEC + 1;
         PREVSEC := CH^.CRSEC
       END;
   CH := CH^.LINK;
 UNTIL (CH = CHMARK) OR (CH^.CRNUM = REMOVE);
 REPEAT
   IF (REMOVE = CH^.CRNUM) AND (PREVSEC <> CH^.CRSEC)
     THEN
       BEGIN
         SEC := SEC + 1;
         PREVSEC := CH^.CRSEC
       END;
   CH := CH^.LINK;
 UNTIL (CH = CHMARK) OR (CH^.CRNUM <> REMOVE);
 SECT := SEC

END;

(*****************************************************************************)

PROCEDURE REPLACE;
   (* PLACES THE OTHER SECTIONS OF REMOVED COURSES INTO THE TABLE *)

VAR
   COUNT,                                      (* COUNTS NUMBER OF INSERTS *)
   PREVSEC       : INTEGER;                    (* SAVES NUMBER OF PREVIOUS SECTION *)
   CHMARK        : PTR;                        (* MARKS HEAD OF LIST *)
```

```
PROCEDURE SORT(VAR TAB: TABLE; LIMIT : INTEGER);
   (* SORTS THE SECTIONS INSERTED INTO TABLE BY BALANCE *)

CONST    PRIMARY  = 3;                    (* BALANCE IS KEY *)

VAR      INORDER  : BOOLEAN;              (* TRUE - IF TABLE IS IN ORDER *)

BEGIN
  REPEAT
    INORDER := TRUE;
    IF LIMIT > 1
     THEN
      BEGIN
       LIMIT := LIMIT - 1;
       FOR ROW := 1 TO LIMIT DO
         BEGIN
           IF (TAB[ROW,1] = TAB[ROW+1,1]) AND (TAB[ROW,PRIMARY] < TAB[ROW+1,PRIM
ARY])
             THEN
               BEGIN
                 INORDER := FALSE;
                 HOLD := TAB[ROW];
                 TAB[ROW] := TAB[ROW+1];
                 TAB[ROW + 1] := HOLD;
               END
         END;
      END;
  UNTIL INORDER;
END;

PROCEDURE MOVEDOWN(VAR TAB : TABLE);
   (*MOVE EVERYTHING IN THE TABLE DOWN A ROW TO INSERT AT HEAD OF TABLE *)

VAR
  SAVE1,                                (* SAVE FIRST ROW FOR SWAP *)
  SAVE2    : ONEROW;                    (* SAVE SECOND ROW FOR SWAP *)
  C        : INTEGER;                   (* TABLE INDEX *)

BEGIN
  SAVE1 := TAB[1];
  SAVE2 := TAB[2];
  C := 1;
  WHILE TAB[C,1] <> 0 DO
    BEGIN
      C := C + 1;
      TAB[C] := SAVE1;
      SAVE1 := SAVE2;
      SAVE2 := TAB[C+1]
    END
END;

PROCEDURE ASSIGN;
   (* ASSIGNS A COURSE TO TABLE *)

BEGIN
  PREVSEC := CH^.CRSEC;
  COUNT := COUNT + 1;
  MOVEDOWN(TAB);
  TAB[1,1] := CH^.CRNUM;
  TAB[1,2] := CH^.CRSEC;
  TAB[1,3] := CH^.MAX - CH^.ENROL;
  TAB[1,4] := 0
END;

BEGIN                                   (* REPLACE *)
```

```
 COUNT := 0;
 CHMARK := CH;
 IF REMOVE = CH^.CRNUM
  THEN ASSIGN
  ELSE
   BEGIN
    REPEAT
      CH := CH^.LINK;
      IF REMOVE = CH^.CRNUM
        THEN ASSIGN;
    UNTIL (CH = CHMARK) OR (REMOVE = CH^.CRNUM);
   END;
 REPEAT
   IF (REMOVE = CH^.CRNUM) AND (CH^.CRSEC <> PREVSEC)
     THEN ASSIGN;
   CH := CH^.LINK;
 UNTIL (REMOVE <> CH^.CRNUM) OR (CH = CHMARK);
 SORT(TAB,COUNT);
END;                                        (* REPLACE *)

(*************************************************************************)


PROCEDURE FINDC;
   (* FINDS THE COURSE IN LIST AND ADDS ONE TO COURSE ENROLLMENT          *)

  VAR  LOOKAHEAD,                           (* LOOKAHEAD POINTER TO FIND PLACE *)
       CHASER       : PTR;                  (* CHASER POINTER FOR INSERTION *)

 BEGIN
  LOOKAHEAD := FRONT;
  CHASER := FRONT;

  WHILE (LOOKAHEAD <> NIL) AND (CRS > LOOKAHEAD^.CRNUM) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK
    END ;   (* WHILE *)

  WHILE (LOOKAHEAD <> NIL) AND (CRS = LOOKAHEAD^.CRNUM) AND
        (SEC <> LOOKAHEAD^.CRSEC) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK
    END ;   (* WHILE *)

  WHILE (LOOKAHEAD <> NIL) AND (CRS = LOOKAHEAD^.CRNUM) AND
        (SEC = LOOKAHEAD^.CRSEC) DO
    BEGIN
      LOOKAHEAD^.ENROL := LOOKAHEAD^.ENROL + 1;
      CHASER := LOOKAHEAD;                  (* IF CRS AND SEC NUMBER = *)
      LOOKAHEAD := LOOKAHEAD^.LINK          (* INCREMENT ENROL *)
    END ;   (* WHILE *)
  END;


(*************************************************************************)

PROCEDURE SHCOUNT;
   (* ADDS STUDENT TO STUDY HALL COUNT FOR UNSCHEDULED PERIODS *)

BEGIN
  FOR RO := 3 TO NUMPD DO
    FOR CO := 1 TO NUMD DO
      BEGIN
```

```
        IF S1[RO,CO] = Ø
          THEN SH1[RO,CO] := SH1[RO,CO] + 1;
        IF S2[RO,CO] = Ø
          THEN SH2[RO,CO] := SH2[RO,CO] + 1;
      END
END;

(*************************************************************************)

PROCEDURE NEXTCRS;
   (* REMOVES ALL SECTIONS OF A COURSE FROM TABLE AND MOVES UP THE
      NEXT COURSE TO ROW 1 *)

VAR
   ROWC,                                  (* COUNTS ROWS *)
   PREV  : INTEGER;                       (* PREVIOUSLY SCHEDULED COURSE*)

BEGIN
  PREV := TAB[1,1];
  REPEAT
    ROWC := 1;
    WHILE TAB[ROWC,1] > Ø DO
      BEGIN
        TAB[ROWC] := TAB[ROWC + 1];
        ROWC := ROWC + 1
      END;
  UNTIL PREV <> TAB[1,1]
END;

(*************************************************************************)

PROCEDURE NEXTSEC;
   (* MOVES NEXT SECTION OF A COURSE TO ROW 1 OF TABLE *)

VAR
  PREVSEC,                                (* PREVIOUS SECTION *)
  ROWC     : INTEGER;                     (* COUNTS SECTIONS *)

BEGIN
  PREVSEC := TAB[1,2];
  REPEAT
    ROWC := 1;
    WHILE TAB[ROWC,1] > Ø DO
      BEGIN
        TAB[ROWC] := TAB[ROWC + 1];
        ROWC := ROWC + 1
      END;
  UNTIL PREVSEC <> TAB[1,2]
END;

(*************************************************************************)



(*************************************************************************)

BEGIN                                     (* UNIT TWO   *)
  HEAD := NIL;
  FETCHIT(HEAD);

END.                                      (* UNIT TWO   *)
```

10.8. PRINTSCHEDULES

```
PROGRAM PRINTSCHEDULES;

    (*************************************************************************)
    (*                                                                       *)
    (*     THIS PROGRAM PRINTS OUT STUDENT SCHEDULES AFTER THE FINAL RUN     *)
    (*          OF THE SCHEDULE HAS BEEN COMPLETED.                          *)
    (*                                                                       *)
    (*     FILES USED                                                        *)
    (*          - SCH : SCH(GRADES)                                          *)
    (*                                                                       *)
    (*     OUTPUT                                                            *)
    (*          - STUDENT SCHEDULES BY GRADE                                 *)
    (*          - A SCHEDULE IS A LIST OF                                    *)
    (*               - STUDENT NAME                                          *)
    (*               - STUDENT ID                                            *)
    (*               - STUDENT GRADE                                         *)
    (*               - COURSES SCHEDULED ( A LIST OF )                       *)
    (*                    - COURSE NAME                                      *)
    (*                    - COURSE NUMBER                                    *)
    (*                    - PERIOD                                           *)
    (*                    - DAYS SCHEDULED                                   *)
    (*                    - ROOM                                             *)
    (*                    - SEMESTER                                         *)
    (*                    - TEACHER NUMBER                                   *)
    (*                                                                       *)
    (*************************************************************************)


CONST
   TOTC          = 11;                    (* TOTAL COURSES        *)
   TOT           = 16;                    (* MAX PERIODS          *)


TYPE
   SC            = ARRAY[1..2] OF INTEGER;
   TALLY         = ARRAY[1..TOTC] OF SC;
   SCHPTR        = ^SCHREC;
   YEAR          = 7..12;
   SCHREC        =
     RECORD
       STUDNAM : STRING[20];              (* STUDENT NAME *)
       STUDID  : INTEGER;                 (* STUDENT ID   *)
       GRADE   : YEAR;                    (* STUDENT GRADE*)
       CRSSCH  : TALLY;                   (* STUD SCHEDULE*)
       L       : SCHPTR;                  (* POINTER      *)
     END;
   POINTER           = ^COURSE;
   COURSENUMBER      = 0..999;
   COURSESECTION     = 1..99;
   MAXSIZE           = 1..999;
   PERIODNUMBER      = 1..16;
   SEMESTER          = 1..3;
   TEACHERNUMBER     = 1..99;
   COURSE            =
     RECORD
       CRNUM     : COURSENUMBER;          (* COURSE NUMBER (0..999) *)
       CRNAM     : STRING[10];            (* COURSE NAME ( 10 CHAR) *)
       CRSEC     : COURSESECTION;         (* COURSE SECTION (1..99) *)
       MAX       : INTEGER;               (* MAX SIZE OF CLASS      *)
```

```
      PERIOD    : PERIODNUMBER;          (* PERIOD OF DAY (1..16    *)
      PD        : PERIODNUMBER;          (* SCHEDULING PERIOD       *)
      DAY       : STRING[5];             (* DAY OF WEEK (MTWRF)     *)
      SEM       : SEMESTER;              (* SEMESTER                *)
      ROOM      : STRING[3];             (* ROOM NUMBER             *)
      TEACHER   : TEACHERNUMBER;         (* TEACHER NUMBER          *)
      LINK      : POINTER;               (* POINTER TO NEXT RECORD *)
    END;
  PRINTREC          =
    RECORD
      PD        : PERIODNUMBER;          (* SCHEDULING PERIOD       *)
      PERIOD    : PERIODNUMBER;          (* PERIOD OF DAY (1..16    *)
      DAY       : STRING[5];             (* DAY OF WEEK (MTWRF)     *)
      COURSE    : STRING[10];            (* COURSE NAME ( 10 CHAR) *)
      ROOM      : STRING[3];             (* ROOM NUMBER             *)
      SEM       : SEMESTER;              (* SEMESTER                *)
      TEACHER   : TEACHERNUMBER;         (* TEACHER NUMBER          *)
      CODE      : COURSENUMBER;          (* COURSE NUMBER (0..999) *)
    END;
  PRAR          = ARRAY[1..TOT] OF PRINTREC;
  INDEXLIST     = ARRAY[1..TOT] OF INTEGER;


VAR
  SCHRECD           : FILE OF SCHREC;   (* FILE OF STUDENT SCHEDULES *)
  DIFPER,                               (* ANOTHER SECTION W DIF PER *)
  DIFSEM,                               (* ANOTHER SECT W DIF SEM    *)
  FOUND,                                (* COURSE LOCATED            *)
  GETOUT          : BOOLEAN;            (* EXIT LOOP *)
  PRESENT         : YEAR;               (* CURRENT VALUE OF YEAR *)
  FILEGRADE,                            (* STRING VALUE OF YEAR *)
  S,                                    (* STRING VALUE OF YEAR *)
  FNAME,                                (* NAME OF SCHEDULE FILE *)
  FILENAME        : STRING;             (* NAME OF STUDENT FILE *)
  LOW,                                  (* LOWEST GRADE IN SCHOOL  *)
  HIGH            : YEAR;               (* HIGHEST GRADE IN SCHOOL *)
  HEAP1,                                (* TOP OF HEAP STUD SCH    *)
  HEAP            : ^INTEGER;           (* TOP OF HEAP MARKER     *)
  TAL             : TALLY;              (* LIST OF COURSES TO BE TALLIED *)
  SCHNODE,                              (* STUDENT SCHEDULE RECORD *)
  SHEAD           : SCHPTR;             (* HEAD OF STUD SCH LIST   *)
  CRNUM      : COURSENUMBER;
  CRNAM      : STRING[10];
  CRSEC      : COURSESECTION;
  MAX        : MAXSIZE;
  SAVEPER,
  PERIOD     : PERIODNUMBER;
  SAVESEM,
  SEM        : SEMESTER;
  DAY        : STRING[5];
  TEACHER    : TEACHERNUMBER;
  ROOM       : STRING[3];
  NODE,                                 (* POINTER TO COURSE REC    *)
  HEAD       : POINTER;                 (* POINTER TO START OF LIST *)
  ANS        : CHAR;                    (* YES OR NO (Y OR N)       *)
  CRREC      : FILE OF COURSE;          (* FILE BUFFER              *)
  OUTDEVICE  : TEXT;                    (* PRINTER                  *)
  LIST       : PRAR;                    (* LIST OF SCHEDULED CRS    *)
  INDLIST    : INDEXLIST;               (* INDEX FOR PRINT ORDER    *)
  INDEX,                                (* COUNT OF COURSES         *)
  C          : INTEGER;                 (* COUNTS STUDENTS          *)

SEGMENT PROCEDURE PRINTSTSCH;
    (* PRINTS OUT STUDENT SCHEDULES FROM SCH FILE *)
```

```
PROCEDURE FINDDSNODE (HD: POINTER;VAR  ITEM : POINTER; VAR TEST,DP,DS :BOOLEAN);
  (* TO LOCATE A NODE WHICH HAS THE SAME CRS NUM AND SEC BUT A DIF SEM  *)

VAR  LOOKAHEAD,                                  (* LOOKAHEAD POINTER TO FIND PLACE *)
       CHASER       : POINTER;                   (* CHASER POINTER FOR INSERTION *)

BEGIN
  TEST := FALSE;
  DP := FALSE;
  DS := FALSE;
  LOOKAHEAD := HD;
  CHASER    := HD;
  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM > LOOKAHEAD^.CRNUM) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK;
    END ;
  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM = LOOKAHEAD^.CRNUM) AND
        (ITEM^.CRSEC <> LOOKAHEAD^.CRSEC) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK;
    END ;
WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM = LOOKAHEAD^.CRNUM) AND
        (ITEM^.CRSEC  = LOOKAHEAD^.CRSEC) AND
        (ITEM^.SEM = LOOKAHEAD^.SEM)  DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK;
    END ;

  IF (LOOKAHEAD <> NIL) AND  (ITEM^.CRNUM = LOOKAHEAD^.CRNUM ) AND
     (ITEM^.CRSEC = LOOKAHEAD^.CRSEC) AND (ITEM^.SEM <> LOOKAHEAD^.SEM)
        THEN
         BEGIN
          TEST := TRUE;
          ITEM^ := LOOKAHEAD^;
         END;
END;


PROCEDURE FINDDPNODE (HD: POINTER;VAR  ITEM : POINTER; VAR TEST,DP,DS :BOOLEAN);
  (* TO LOCATE A NODE WHICH HAS THE SAME CRS NUM AND SEC BUT A DIF PER  *)

VAR  LOOKAHEAD,                                  (* LOOKAHEAD POINTER TO FIND PLACE *)
       CHASER       : POINTER;                   (* CHASER POINTER FOR INSERTION *)

BEGIN
  TEST := FALSE;
  DP := FALSE;
  DS := FALSE;
  LOOKAHEAD := HD;
  CHASER    := HD;
  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM > LOOKAHEAD^.CRNUM) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK;
    END ;
  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM = LOOKAHEAD^.CRNUM) AND
        (ITEM^.CRSEC <> LOOKAHEAD^.CRSEC) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK;
    END ;
```

```
WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM = LOOKAHEAD^.CRNUM) AND
        (ITEM^.CRSEC  = LOOKAHEAD^.CRSEC) AND
        (ITEM^.PERIOD >= LOOKAHEAD^.PERIOD)  DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK;
    END ;

  IF (LOOKAHEAD <> NIL) AND  (ITEM^.CRNUM = LOOKAHEAD^.CRNUM ) AND
     (ITEM^.CRSEC = LOOKAHEAD^.CRSEC) AND (ITEM^.PERIOD < LOOKAHEAD^.PERIOD)
        THEN
         BEGIN
          TEST := TRUE;
          ITEM^ := LOOKAHEAD^;
          CHASER := LOOKAHEAD;
          LOOKAHEAD := LOOKAHEAD^.LINK;
          WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM = LOOKAHEAD^.CRNUM) AND
                  (ITEM^.CRSEC  = LOOKAHEAD^.CRSEC) AND NOT DP  DO
              BEGIN
                IF ITEM^.PERIOD < LOOKAHEAD^.PERIOD
                  THEN
                   BEGIN
                    WRITELN('FOUND IT');
                    DP := TRUE;
                   END;
                CHASER := LOOKAHEAD;
                LOOKAHEAD := LOOKAHEAD^.LINK;
              END ;
         END;


END;

                    .


PROCEDURE FINDNODE (HD: POINTER;VAR  ITEM : POINTER; VAR TEST,DP,DS :BOOLEAN);
  (* TO LOCATE A NODE USING THE COURSE NUMBER *)

VAR  LOOKAHEAD.                          (* LOOKAHEAD POINTER TO FIND PLACE *)
       CHASER       : POINTER;           (* CHASER POINTER FOR INSERTION *)

BEGIN
  TEST := FALSE;
  DP := FALSE;
  DS := FALSE;
  LOOKAHEAD := HD;
  CHASER    := HD;
  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM > LOOKAHEAD^.CRNUM) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK;
    END ;
  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM = LOOKAHEAD^.CRNUM) AND
        (ITEM^.CRSEC <> LOOKAHEAD^.CRSEC) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK;
    END ;

  IF (LOOKAHEAD <> NIL) AND  (ITEM^.CRNUM = LOOKAHEAD^.CRNUM ) AND
     (ITEM^.CRSEC = LOOKAHEAD .CRSEC)
        THEN
         BEGIN
          TEST := TRUE;
          ITEM^ := LOOKAHEAD^;
```

```
          WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM = LOOKAHEAD^.CRNUM) AND
                (ITEM^.CRSEC  = LOOKAHEAD^.CRSEC) AND ( NOT DP AND NOT DS) DO
            BEGIN
              CHASER := LOOKAHEAD;
              LOOKAHEAD := LOOKAHEAD^.LINK;
              IF ITEM^.PERIOD <> LOOKAHEAD^.PERIOD
                THEN DP := TRUE;
              IF ITEM^.SEM  <> LOOKAHEAD^.SEM
                THEN DS := TRUE;
            END ;
        END;


END;


(*************************************************************************)

(*************************************************************************)

PROCEDURE IOPRINT(ERRTYPE:INTEGER);
   (* PRINT ERROR MESSAGE *)

BEGIN
 GOTOXY(0,22);
 WRITE(CHR(7));
 CASE ERRTYPE OF
   1 : WRITELN('DISK READ ERROR.');
   2 : WRITELN('BAD UNIT NUMBER.');
   3 : WRITELN('ILLEGAL OPERATION.');
   4 : WRITELN('UNDEFINED HARDWARE ERROR.');
   5 : WRITELN('UNIT NO LONGER ON LINE.');
   6 : WRITELN('FILE NOT IN DIRECTORY.');
   7 : WRITELN('ILLEGAL FILE NAME.');
   8 : WRITELN('INSUFFICIENT DISK SPACE.');
   9 : WRITELN('NO SUCH VOLUME ON LINE.');
  10 : WRITELN('NO SUCH FILE ON VOLUME.');
  11 : WRITELN('DUPLICATE FILE.');
  12 : WRITELN('ATTEMPT TO OPEN AN OPEN FILE.');
  13 : WRITELN('FILE NOT OPEN.');
  14 : WRITELN('ERROR IN READING INTEGER OR REAL DATA.');
  15 : WRITELN('RING BUFFER OVERFLOW.');
 END;
END;

PROCEDURE NODISK(DISKNAME:STRING; DISKERR : INTEGER);
   (* CHECKS FOR ERROR 9 *)

VAR CH : CHAR;                                  (* ANY KEY *)

BEGIN
 IF (DISKERR <> 0) AND ((DISKERR < 9) OR (DISKERR > 10))
   THEN IOPRINT(DISKERR);
 IF (DISKERR = 9)
   THEN
    BEGIN
     GOTOXY(0,22);
     WRITELN('NO ',DISKNAME,' DISKETTE!');
     WRITELN('PLEASE INSERT DISKETTE AND HIT ANY KEY');
     READ(KEYBOARD,CH);
    END;
 IF (DISKERR = 10)
   THEN REWRITE(SCHRECD,DISKNAME);
END;
```

```
PROCEDURE OPENFILE(FILENAME : STRING);
   (* OPENS A FILE AND CHECKS FOR ERRORS *)

VAR IOERR    : INTEGER;                          (* ERROR NUMBBER *)

BEGIN
 REPEAT
   (*$I-*)
   RESET(SCHRECD,FILENAME);
   (*$I+*)
   IOERR := IORESULT;
   NODISK(FILENAME,IOERR);
 UNTIL IOERR = 0;
END;


(*************************************************************************)
 PROCEDURE WRSCH;
    (* PRINTS A STUDENTS SCHEDULE *)

VAR  I,J : INTEGER;                              (* LOOP COUNTER *)

PROCEDURE INITINDEX(VAR INDEX : INDEXLIST);
    (* INITIALIZES INDEX LIST TO 99 *)

BEGIN
 FOR J := 1 TO TOT DO
    INDEX[J] := 99;
END;

PROCEDURE ADDTOINDEX(VAR INDEX : INDEXLIST; NEWPER : PERIODNUMBER);
    (* ADDS PD OF SCHEDULED CRS TO INDEX LIST *)

VAR FLAG : BOOLEAN;                              (* COURSE ENTERED INTO LIST *)
     SAVE1,SAVE2,                                (* TEMP VALUES              *)
     J1, I : INTEGER;                            (* LOOP COUNTERS            *)

BEGIN
 FLAG := TRUE;
 FOR J1 := 1 TO TOT DO
  BEGIN
   IF FLAG
    THEN
     BEGIN
      IF NEWPER = INDEX[J1]
          THEN FLAG := FALSE;
      IF NEWPER < INDEX[J1]
       THEN
        BEGIN
         I := J1;
         SAVE1 := NEWPER;
         SAVE2 := INDEX[I];
         WHILE SAVE1 < 99 DO
          BEGIN
           INDEX[I] := SAVE1;
           SAVE1 := SAVE2;
           SAVE2 := INDEX[I+1];
           I := I + 1;
          END;
         FLAG := FALSE;
        END;
     END
  END
END;

PROCEDURE ADDTOPRLIST(NODE : POINTER; VAR REC : PRINTREC);
```

```
   (* ADDS A COURSE TO LIST OF COURSES TO BE PRINTED *)

BEGIN
 WITH REC DO
  BEGIN
   PD     := NODE^.PD;
   PERIOD := NODE^.PERIOD;
   DAY    := NODE^.DAY;
   COURSE := NODE^.CRNAM;
   ROOM   := NODE^.ROOM;
   SEM    := NODE^.SEM;
   TEACHER := NODE^.TEACHER;
   CODE    := NODE^.CRNUM;
  END;
 END;

 PROCEDURE PRINTCOUR(INDLIST : INDEXLIST; LIST : PRAR; COUNT : INTEGER);
    (* PRINTS SCHEDULED COURSES FROM LIST *)

 BEGIN
  FOR J := 1 TO COUNT - 1 DO
   BEGIN
    IF INDLIST[J] < 99
     THEN
      BEGIN
       FOR I := 1 TO COUNT - 1 DO
         BEGIN
          IF LIST[I].PD = INDLIST[J]
           THEN
            BEGIN
             WITH LIST[I]   DO
              BEGIN
               WRITE(OUTDEVICE,PERIOD:4,DAY:9,COURSE:12,ROOM:7);
               WRITELN(OUTDEVICE,SEM:5,TEACHER:8,CODE:8);
              END
            END;
         END
      END
   END
 END;


BEGIN                                          (* WR SCH *)
  REWRITE(OUTDEVICE,'PRINTER:');
  WRITELN(OUTDEVICE);
  WRITELN(OUTDEVICE);
  WRITELN(OUTDEVICE);
  WRITELN(OUTDEVICE,'STUDENT NAME:  ',SCHRECD^.STUDNAM:20);
  WRITELN(OUTDEVICE,'         ID  :  ',SCHRECD^.STUDID:5);
  WRITELN(OUTDEVICE,'        GRADE: ',SCHRECD^.GRADE:2);
  WRITELN(OUTDEVICE);
  WRITELN(OUTDEVICE,'PERIOD   DAYS    COURSE     ROOM   SEM   TEACHER    CODE');
  WRITELN(OUTDEVICE);
  INDEX := 1;
  INITINDEX(INDLIST);
  FOR J := 1 TO TOTC DO
   BEGIN
    IF SCHRECD^.CRSSCH[J,1] > 0
     THEN
      BEGIN
       DIFFER := FALSE;
       DIFSEM := FALSE;
       NEW(NODE);
       NODE^.CRNUM := SCHRECD^.CRSSCH[J,1];
       NODE^.CRSEC := SCHRECD^.CRSSCH[J,2];
       FINDNODE(HEAD,NODE,FOUND,DIFFER,DIFSEM);
```

```
        WHILE FOUND DO
          BEGIN
           SAVEPER := NODE^.PERIOD;
           SAVESEM := NODE^.SEM;
           ADDTOINDEX(INDLIST,NODE^.PD);
           ADDTOPRLIST(NODE,LIST[INDEX]);
           INDEX := INDEX + 1;
           FOUND := FALSE;
           IF DIFPER OR DIFSEM
            THEN
             BEGIN
              NEW(NODE);
              NODE^.CRNUM := SCHRECD^.CRSSCH[J,1];
              NODE^.CRSEC := SCHRECD^.CRSSCH[J,2];
             END;
            IF DIFPER
             THEN
              BEGIN
               DIFPER := FALSE;
               NODE^.PERIOD := SAVEPER;
               FINDDPNODE(HEAD,NODE,FOUND,DIFPER,DIFSEM)
              END
             ELSE
              BEGIN
               IF DIFSEM
                THEN
                 BEGIN
                  DIFSEM := FALSE;
                  NODE^.SEM := SAVESEM;
                  FINDDSNODE(HEAD,NODE,FOUND,DIFPER,DIFSEM)
                 END
              END
          END;
      END;
    END;
  PRINTCOUR(INDLIST,LIST,INDEX);
  CLOSE(OUTDEVICE);

END;                                          (* WR SCH *)


BEGIN                                         (* PRINTSTSCH *)
  GETOUT := FALSE;
  PRESENT := LOW;
  REPEAT
    STR(PRESENT,FILEGRADE);
    FILENAME := CONCAT('SCH:SCH',FILEGRADE);
    OPENFILE(FILENAME);
    WHILE NOT EOF (SCHRECD)     DO
      BEGIN
        MARK(HEAP);
        WRSCH;
        RELEASE(HEAP);
        GET(SCHRECD);
      END;
    CLOSE(SCHRECD);
    IF PRESENT = HIGH
      THEN GETOUT := TRUE
      ELSE PRESENT := PRESENT + 1;
  UNTIL GETOUT = TRUE;
 END;                                         (* PRINTSTSCH *)


PROCEDURE INSERTNODE (VAR FRONT, ITEM : POINTER);
```

```
  (* TO PUT NEW NODE IN LIST IN ORDER OF COURSE NUMBER, CRS SECTION, AND PER *)

  VAR  LOOKAHEAD,                          (* LOOKAHEAD POINTER TO FIND PLACE *)
       CHASER       : POINTER:             (* CHASER POINTER FOR INSERTION *)

BEGIN
  LOOKAHEAD := FRONT;
  CHASER := FRONT;

  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM > LOOKAHEAD^.CRNUM) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK;
    END ;
  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM = LOOKAHEAD^.CRNUM) AND
        (ITEM^.CRSEC > LOOKAHEAD^.CRSEC)  DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK;
    END ;
  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM = LOOKAHEAD^.CRNUM) AND
       (ITEM^.CRSEC = LOOKAHEAD^.CRSEC) AND (ITEM^.PD > LOOKAHEAD^.PD) DO
    BEGIN
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LINK;
    END ;

  IF (LOOKAHEAD <> NIL)
    THEN
      IF LOOKAHEAD <> FRONT
        THEN
          BEGIN
            ITEM^.LINK := LOOKAHEAD:
            CHASER^.LINK := ITEM;
          END                              (* INSERT IN MIDDLE OF LIST *)
        ELSE
          BEGIN
            ITEM^.LINK := FRONT;
            FRONT := ITEM;
          END                              (* INSERT AT FRONT *)
    ELSE                                   (* MUST BE LARGEST, SO INSERT AT END*)
      CHASER^.LINK := ITEM;

  END;

PROCEDURE FETCHIT(VAR FIRST : POINTER);
   (* RETRIEVES COURSE LIST FROM FILE AND ENTERS COURSES IN A
      LINKED LIST *)

VAR  CRS  : POINTER:                       (* COURSE RECORD *)

BEGIN
RESET(CRREC,'CR:COURSEFILE');

WHILE NOT EOF(CRREC) DO
  BEGIN

    NEW(CRS);
    CRS^ := CRREC^;
    IF FIRST = NIL
      THEN FIRST := CRS
      ELSE INSERTNODE(FIRST,CRS);
    GET(CRREC);
  END;
CLOSE(CRREC);
END;
```

```
FUNCTION ASKCHOICE(X,Y : INTEGER; Q :STRING; MIN,MAX :INTEGER) : INTEGER;
   (* ASK USER FOR A NUMBER FROM MIN TO MAX, AND REPEAT UNTIL OBTAINED *)

VAR   ANSWER      : INTEGER;                    (* A NUMBER *)

BEGIN
REPEAT
  GOTOXY(X,Y);
  WRITE(Q,' ( ',MIN,'..',MAX,' ) : ');
  READLN(ANSWER);
UNTIL (ANSWER >= MIN) AND (ANSWER <= MAX);
ASKCHOICE := ANSWER
END;



BEGIN                                           (* PRINTSCHEDULES *)
  HEAD := NIL;
  FETCHIT(HEAD);
  PAGE(OUTPUT);
  LOW := ASKCHOICE(0,5,'WHAT IS THE LOWEST GRADE YOU WISH CLASS LISTS FOR',7,12)
;
  HIGH := ASKCHOICE(0,10,'WHAT IS THE HIGHEST GRADE YOU WISH CLASS LISTS FOR',7,
12);
  PRINTSTSCH;

END.                                            (* PRINTSCHEDULES *)
```

10.9. PRINTCLASSLISTS

```
PROGRAM PRINTCLASSLISTS;

    (*************************************************************************)
    (*                                                                       *)
    (*      THIS PROGRAM PRINTS CLASS LISTS USING RECORDS FROM  THE          *)
    (*          SCHEDULE FILES.                                              *)
    (*                                                                       *)
    (*      FILES USED                                                       *)
    (*         - CR : COURSEFILE                                             *)
    (*         - SCH : SCH(GRADE)                                            *)
    (*                                                                       *)
    (*      OUTPUT                                                           *)
    (*         - A CLASS LIST FOR EVERY COURSE SCHEDULED.                    *)
    (*         - A CLASS LIST CONTAINS                                       *)
    (*              - COURSE NAME                                            *)
    (*              - COURSE NUMBER                                          *)
    (*              - COURSE SECTION                                         *)
    (*              - DAYS THE CLASS IS SCHEDULED                            *)
    (*              - ROOM                                                   *)
    (*              - TEACHER NUMBER                                         *)
    (*              - LIST OF STUDENTS  IN CLASS                             *)
    (*                   - STUDENT ID                                        *)
    (*                   - STUDENT GRADE                                     *)
    (*                   - STUDENT NAME                                      *)
    (*                                                                       *)
    (*************************************************************************)


CONST
   TOTC         = 11;                       (* TOTAL COURSES       *)


TYPE
   SC           = ARRAY[1..2] OF INTEGER;
   TALLY        = ARRAY[1..TOTC] OF SC;
   SCHPTR       = ^SCHREC;
   CLPTR        = ^CLREC;
   YEAR         = 7..12;
   POINTER          = ^COURSE;
   COURSENUMBER     = 0..999;
   COURSESECTION    = 1..99;
   MAXSIZE          = 1..999;
   PERIODNUMBER     = 1..16;
   SEMESTER         = 1..3;
   TEACHERNUMBER    = 1..99;
   CLREC        =
     RECORD
       CRS      : COURSENUMBER;           (* COURSE NUMBER*)
       SEC      : COURSESECTION;          (* COURSE SECT  *)
       STUDNAM  : STRING[20];             (* STUDENT NAME *)
       STUDID   : INTEGER;                (* STUDENT ID   *)
       GRADE    : YEAR;                   (* STUDENT GRADE*)
       LCL      : CLPTR;                  (* POINTER      *)
     END;
   SCHREC       =
     RECORD
       STUDNAM  : STRING[20];             (* STUDENT NAME *)
       STUDID   : INTEGER;                (* STUDENT ID   *)
```

```
      GRADE    : YEAR;                    (* STUDENT GRADE*)
      CRSSCH   : TALLY;                   (* STUD SCHEDULE*)
      L        : SCHPTR;                  (* POINTER      *)
    END;
  COURSE            =
    RECORD
      CRNUM     : COURSENUMBER;           (* COURSE NUMBER (0..999) *)
      CRNAM     : STRING[10];             (* COURSE NAME ( 10 CHAR) *)
      CRSEC     : COURSESECTION;          (* COURSE SECTION (1..99) *)
      MAX       : INTEGER;                (* MAX SIZE OF CLASS      *)
      PERIOD    : PERIODNUMBER;           (* PERIOD OF DAY (1..16   *)
      PD        : PERIODNUMBER;           (* SCHEDULING PERIOD      *)
      DAY       : STRING[5];              (* DAY OF WEEK (MTWRF)    *)
      SEM       : SEMESTER;               (* SEMESTER               *)
      ROOM      : STRING[3];              (* ROOM NUMBER            *)
      TEACHER   : TEACHERNUMBER;          (* TEACHER NUMBER         *)
      LINK      : POINTER;                (* POINTER TO NEXT RECORD *)
    END;


VAR
  SCHREC
  TEMP
  FOUND,
  GETOUT
  PRESEN
  FILEGR
  S,
  FNAME,
  FILENA
  LOW,
  HIGH
  HEAP
  SCHNODI
  SHEAD
  CRNUM
  CRNAM
  CRSEC
  MAX
  PERIOD
  SEM
  DAY
  TEACHEI
  ROOM
  NODE,
  HEAD
  CRREC
  OUTDEV
  C
  TAL
  STUDC
  J
  CLNODE
  CLHEAD


(*********)

 PROCEDUF
    (* PF


   BEGIN

     REWF
     PAGE
```

```
      IF FRONT = NIL
        THEN                                    (* LIST IS EMPTY *)
          BEGIN
            WRITELN (OUTDEVICE,'LIST IS EMPTY.');
          END                                   (* EMPTY *)
        ELSE                                    (* THERE ARE COURSES *)
          BEGIN
            WRITELN(OUTDEVICE,'CLASS LIST     COURSE  : ',NODE^.CRNAM);
            WRITELN(OUTDEVICE,'               NUMBER  : ',NODE^.CRNUM:3,' SEC ',N
ODE^.CRSEC:2);
            WRITELN(OUTDEVICE,'               PERIOD  : ',NODE^.PERIOD:2);
            WRITELN(OUTDEVICE,'               DAYS    : ',NODE^.DAY:5);
            WRITELN(OUTDEVICE,'               SEM     : ',NODE^.SEM:2);
            WRITELN(OUTDEVICE,'               ROOM    : ',NODE^.ROOM:3);
            WRITELN(OUTDEVICE,'               TEACHER : ',NODE^.TEACHER:2);
            WRITELN(OUTDEVICE);
            WRITELN(OUTDEVICE);
            WRITELN(OUTDEVICE,' ID#    GRADE         STUDENT NAME');
            WRITELN(OUTDEVICE);
            WHILE FRONT <> NIL DO
              BEGIN
               WRITELN(OUTDEVICE,FRONT^.STUDID:5,FRONT^.GRADE:6,'      ',FRONT^.S
TUDNAM);
               FRONT := FRONT^.LCL;
              END;                              (* WHILE *)
          END;                                  (* ELSE LIST NOT EMPTY *)
       CLOSE(OUTDEVICE);                        (* TURN OFF PRINTER *)
     END;                                       (* PRINTLIST *)


(*************************************************************************)

PROCEDURE INSERTSCH(VAR FRONT, ITEM : CLPTR);

   (* INSERTS A STUDENT CLASS LIST  RECORD INTO LINKED LIST *)

VAR  LOOKAHEAD,                                 (* POINTER TO FIND PLACE *)
     CHASER        :  CLPTR;                    (* POINTER FOR INSERTION *)

BEGIN
  LOOKAHEAD := FRONT;
  CHASER    := FRONT;
  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.STUDNAM > LOOKAHEAD^.STUDNAM) DO
    BEGIN                                       (* TRAVERSE LIST TO INSERT POINT *)
      CHASER := LOOKAHEAD;
      LOOKAHEAD := LOOKAHEAD^.LCL
    END;

  IF LOOKAHEAD <> NIL
    THEN                                        (* NOT AT END OF LIST *)
    IF LOOKAHEAD <> FRONT
      THEN
        BEGIN                                   (* INSERT IN.MIDDLE OF LIST *)
          ITEM^.LCL := LOOKAHEAD;
          CHASER^.LCL := ITEM;
        END
      ELSE
        BEGIN                                   (* INSERT AT FRONT *)
          ITEM^.LCL := FRONT;
          FRONT := ITEM;
        END
   ELSE
      CHASER^.LCL := ITEM;


END;
```

```
(*************************************************************************

PROCEDURE IOPRINT(ERRTYPE:INTEGER);
   (* PRINT ERROR MESSAGE *)

BEGIN
 GOTOXY(0,22);
 WRITE(CHR(7));
 CASE ERRTYPE OF
   1 : WRITELN('DISK READ ERROR.');
   2 : WRITELN('BAD UNIT NUMBER.');
   3 : WRITELN('ILLEGAL OPERATION.');
   4 : WRITELN('UNDEFINED HARDWARE ERROR.');
   5 : WRITELN('UNIT NO LONGER ON LINE.');
   6 : WRITELN('FILE NOT IN DIRECTORY.');
   7 : WRITELN('ILLEGAL FILE NAME.');
   8 : WRITELN('INSUFFICIENT DISK SPACE.');
   9 : WRITELN('NO SUCH VOLUME ON LINE.');
  10 : WRITELN('NO SUCH FILE ON VOLUME.');
  11 : WRITELN('DUPLICATE FILE.');
  12 : WRITELN('ATTEMPT TO OPEN AN OPEN FILE.');
  13 : WRITELN('FILE NOT OPEN.');
  14 : WRITELN('ERROR IN READING INTEGER OR REAL DATA.');
  15 : WRITELN('RING BUFFER OVERFLOW.');
 END;
END;

PROCEDURE NODISK(DISKNAME:STRING; DISKERR : INTEGER);
   (* CHECKS FOR ERROR 9 *)

VAR CH : CHAR;

BEGIN
 IF (DISKERR <> 0) AND ((DISKERR < 9) OR (DISKERR > 10))
   THEN IOPRINT(DISKERR);
 IF (DISKERR = 9)
   THEN
    BEGIN
     GOTOXY(0,22);
     WRITELN('NO ',DISKNAME,  DISKETTE!');
     WRITELN('PLEASE INSERT DISKETTE AND HIT ANY KEY');
     READ(KEYBOARD,CH);
    END;
 IF (DISKERR = 10)
   THEN REWRITE(SCHRECD,DISKNAME);
END;


PROCEDURE OPENFILE(FILENAME : STRING);
  (* OPENS A FILE AND CHECKS FOR ERRORS *)

VAR IOERR   : INTEGER;

BEGIN
 REPEAT
  (*$I-*)
  RESET(SCHRECD,FILENAME);
  (*$I+*)
  IOERR := IORESULT;
  NODISK(FILENAME,IOERR);
 UNTIL IOERR = 0;
END;
```

```
PROCEDURE OPENCFILE(FILENAME : STRING);
  (* OPENS A COURSE FILE AND CHECKS FOR ERRORS *)

VAR IOERR    : INTEGER;

BEGIN
 REPEAT
  (*$I-*)
  RESET(CRREC,FILENAME);
  (*$I+*)
  IOERR := IORESULT;
  NODISK(FILENAME,IOERR);
 UNTIL IOERR = 0;
END;

(*************************************************************************)




PROCEDURE INSERTNODE (VAR FRONT, ITEM : POINTER);
  (* TO PUT NEW NODE IN LIST IN ORDER OF COURSE NUMBER, CRS SECTION, AND PER *)

  VAR  LOOKAHEAD,
       CHASER       : POINTER;
       GETOUT       : BOOLEAN;                    (* DON'T INSERT THIS NODE*)

BEGIN (* INSERTNODE *)
  LOOKAHEAD := FRONT;                          (* LOOKAHEAD POINTER TO FIND PLACE *)
  CHASER := FRONT;                             (* CHASER POINTER FOR INSERTION *)
  GETOUT := FALSE;
  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM > LOOKAHEAD^.CRNUM) DO
    BEGIN                                 (* TRAVERSE THE LIST TO THE INSERT POINT *)
      CHASER := LOOKAHEAD                      (* MOVE CHASER DOWN LIST *);
      LOOKAHEAD := LOOKAHEAD^.LINK             (* MOVE LOOKAHEAD DOWN *);
    END ;  (* WHILE *)
  WHILE (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM = LOOKAHEAD^.CRNUM) AND
         (ITEM^.CRSEC > LOOKAHEAD^.CRSEC)  DO
    BEGIN                                 (* TRAVERSE THE LIST TO THE INSERT POINT *)
      CHASER := LOOKAHEAD                      (* MOVE CHASER DOWN LIST *);
      LOOKAHEAD := LOOKAHEAD^.LINK             (* MOVE LOOKAHEAD DOWN *);
    END ;  (* WHILE *)
  IF  (LOOKAHEAD <> NIL) AND (ITEM^.CRNUM = LOOKAHEAD^.CRNUM) AND
        (ITEM^.CRSEC = LOOKAHEAD^.CRSEC)
        THEN GETOUT := TRUE;
  IF NOT GETOUT
    THEN
     BEGIN
      IF (LOOKAHEAD <> NIL)
        THEN                                       (* NOT AT END OF LIST *)
          IF LOOKAHEAD <> FRONT
            THEN                                   (* IN MIDDLE OF LIST  *)
              BEGIN
                ITEM^.LINK := LOOKAHEAD;
                CHASER^.LINK := ITEM;
              END                                  (* INSERT IN MIDDLE OF LIST *)
            ELSE                                   (* AT FRONT OF LIST *)
              BEGIN
                ITEM^.LINK := FRONT;
                FRONT := ITEM;
              END                                  (* INSERT AT FRONT *)
        ELSE                                       (* MUST BE LARGEST, SO INSERT AT END*

          CHASER^.LINK := ITEM;
```

```
    END;

  END;                                          (* INSERTNODE *)

PROCEDURE FETCHIT(VAR FIRST : POINTER);
   (* RETRIEVES COURSE LIST FROM FILE AND ENTERS COURSES IN A
      LINKED LIST *)

VAR  CRS  : POINTER;

BEGIN
RESET(CRREC,'CR:COURSEFILE');

WHILE NOT EOF(CRREC) DO
  BEGIN

    NEW(CRS);
    CRS^ := CRREC^;
    IF FIRST = NIL
      THEN FIRST := CRS
      ELSE INSERTNODE(FIRST,CRS);
    GET(CRREC);
  END;
CLOSE(CRREC);
END;


FUNCTION ASKCHOICE(X,Y : INTEGER; Q :STRING; MIN,MAX :INTEGER) : INTEGER;
   (* ASK USER FOR A NUMBER FROM MIN TO MAX, AND REPEAT UNTIL OBTAINED *)

VAR   ANSWER     : INTEGER;

BEGIN
REPEAT
  GOTOXY(X,Y);
  WRITE(Q,' ( ',MIN,'..',MAX,' ) : ');
  READLN(ANSWER);
UNTIL (ANSWER >= MIN) AND (ANSWER <= MAX);
ASKCHOICE := ANSWER
END;
PROCEDURE FILLTEMP;
   (* FILLS TEMP FILE WITH STUDENT CLASS LIST RECORDS *)

BEGIN
  PAGE(OUTPUT);
  LOW := ASKCHOICE(0,5,'WHAT IS THE LOWEST GRADE YOU WISH CLASS LISTS FOR',7,12)
;
  HIGH := ASKCHOICE(0,10,'WHAT IS THE HIGHEST GRADE YOU WISH CLASS LISTS FOR ',7,
12);
  GETOUT := FALSE;
  PRESENT := LOW;
  REWRITE(TEMP,'TEMP:T');
  REPEAT
    STR(PRESENT,FILEGRADE);
    FILENAME := CONCAT('SCH:SCH',FILEGRADE);
    OPENFILE(FILENAME);
    C := 1;
    WHILE NOT EOF (SCHRECD)   DO
      BEGIN
        FOR J := 1 TO TOTC DO
         BEGIN
          IF SCHRECD^.CRSSCH[J,1] > 0
           THEN
            BEGIN
             STUDC.CRS        := SCHRECD^.CRSSCH[J,1];
             STUDC.SEC        := SCHRECD^.CRSSCH[J,2];
```

```
                STUDC.STUDNAM    := SCHRECD^.STUDNAM;
                STUDC.STUDID     := SCHRECD^.STUDID;
                STUDC.GRADE      := SCHRECD^.GRADE;
                STUDC.LCL        := NIL;
                TEMP^ := STUDC;
                PUT(TEMP);
              END;
            END;
            GET(SCHRECD);
        END;
      CLOSE(SCHRECD);
      IF PRESENT = HIGH
        THEN GETOUT := TRUE
        ELSE PRESENT := PRESENT + 1;
    UNTIL GETOUT = TRUE;
    CLOSE(TEMP,LOCK);

END;

BEGIN                                          (* PRINT CLASS LISTS *)

 FILLTEMP;
 HEAD := NIL;
 FETCHIT(HEAD);
 NODE := HEAD;
 WHILE NODE <> NIL DO
  BEGIN
    RESET(TEMP,'TEMP:T ');
    MARK(HEAP);
    CLHEAD := NIL;
    REPEAT
     IF (NODE^.CRNUM = TEMP^.CRS) AND (NODE^.CRSEC = TEMP^.SEC)
       THEN
        BEGIN
         NEW(CLNODE);
         CLNODE^ := TEMP^;
         IF CLHEAD = NIL
          THEN CLHEAD := CLNODE
          ELSE INSERTSCH(CLHEAD,CLNODE);
        END;
     GET(TEMP);
     UNTIL EOF(TEMP);
     CLOSE(TEMP);
     PRINTLIST(CLHEAD,NODE);
     RELEASE(HEAP);
     NODE := NODE^.LINK
   END;

END.                                           (* PRINT CLASS LISTS *)
```