DATABASE SYSTEMS DESIGN DECISIONS

A Thesis

Presented To

the Chancellor's Scholars Council

of Pembroke State University

In Partial Fulfillment

of the Requirements for Completion of

the Chancellor's Scholars Program

---

by

Deborah Calabrese

April 16, 1990

TABLE OF CONTENTS

I. INTRODUCTION

II. RESEARCH

A. INTRODUCTION

B. ANALYSIS OF DATA

1. STRUCTURE

2. REQUIREMENTS

3. DEGREES OFFERED

4. COURSES

a. DUAL FULFILLMENT

b. DUAL LISTING

c. MULTIPLE FULFILLMENT

5. STUDENT DATA

a. GRADE FILE

b. STUDENT FILE

III. POSSIBLE SOLUTIONS

A. INTRODUCTION

B. SOLUTION TO STRUCTURE

C. SOLUTION TO REQUIREMENTS

D. SOLUTION TO DEGREES OFFERED

E. SOLUTIONS TO COURSES

1. DUAL FULFILLMENT

2. DUAL LISTING

3. MULTIPLE FULFILLMENT

F. SOLUTIONS TO STUDENT DATA

1. GRADE FILE

2. STUDENT FILE

IV. DESIGN

A. INTERVIEWS

B. DATA FLOW DIAGRAMS

C. SYSTEM FLOWCHARTS

D. HIPO CHARTS

V. IMPLEMENTATION

A. OVERVIEW

B. CHOICE OF IMPLEMENTATION

C. INTERNAL STRUCTURE

D. SEARCH AND SORT ROUTINES

VI. ANOTHER ALTERNATIVE

A. INTRODUCTION

B. DATABASES

1. INTERNAL STRUCTURE

2. LANGUAGE INDEPENDENCE

C. DATABASE MANAGEMENT SYSTEMS

VII. CONCLUSION

VIII. ACKNOWLEDGEMENTS

IX. APPENDIX A - EXAMPLES

X. APPENDIX B - EXAMPLES

XI. APPENDIX C - EXAMPLES

XII. APPENDIX D - FILE DESCRIPTIONS AND STRUCTURES

XIII. APPENDIX E - BASIC STUDIES FILES

XIV. APPENDIX F - DATA FLOW DIAGRAMS, SYSTEM FLOWCHARTS, HIPO CHARTS

XV. APPENDIX G - PROGRAM

XVI. BIBLIOGRAPHY

DATABASE SYSTEMS DESIGN DECISIONS

I. INTRODUCTION

The use of computers to generate reports has been in practice for many years. It is not novel in its' conception, nor is it hard to implement, given the proper equipment and data. Computers are used in just this way every day. Banks use computers to generate bank statements, schools use computers to produce report cards, retail companies use them to provide monthly billing statements, etc.. So, the idea to generate a report which would calculate the graduation requirements for students is not a brand new area of interest to research, but it can be a hard task to carry out. For my Chancellor's Scholars Thesis/Project, I wanted to find out if it was possible to generate a report that would show the courses that had been taken and those still necessary to fulfill the graduation requirements of each student, using the student files that Pembroke State University maintains.

II. RESEARCH

A. INTRODUCTION

The definition of a report, as translated by the Webster's dictionary, is "a statement of facts given in reply to inquiry." My own personal definition of a report

is not unlike this one. I consider a report to be a collection of related data that has been arranged in such a way as to have a significant meaning to the person the report is intended for. But, before a report can be created, the data must exist in some form. This is where the research aspect of my project began, with the data.

In order for me to generate a report that would contain an evaluation of a student's courses, I first needed to compile a listing of all the necessary data that would be contained in the report. From this listing that had been generated, I concluded that I would need to have information on the student's background, their past performance in classes, and the requirements for their selected majors. This information, I realized, could be located in the student catalogs distributed by the university, and the student and grade files that were maintained by the university at the Triangle Universities Computation Center (TUCC).

After I had come to the conclusion that the necessary data did exist and I knew where to locate it, the next step was to analyze the data to see how it was organized, and try to find any possible problems that might arise.

B. ANALYSIS

While analyzing the student files and catalogs, I

broke my research down into five separate areas: the structure of the requirement system at Pembroke State University, the requirements for a student's major and basic studies, the type of degrees offered by the university, the course information obtained from the student catalogs, and the student data located on file at TUCC.

1. Structure

Based upon my research into the previous catalogs that Pembroke State University had distributed, I found that prior to the academic year 1982-1983 a firm structure of basic studies course requirements did not exist. For instance, for the academic year 1980-1981, a student could select twenty-four semester hours out of any of the 100- or 200-level courses listed under the title of Humanities to fulfill the Humanities requirement of their basic studies. (See Appendix A for an example of the basic studies requirements for the academic school years 1980-1981 and 1982-1983.) After the 1981-1982 school year, the basic studies requirements were standardized. This restructuring caused certain classes to be excluded from the basic studies requirements, and the students' choices became more limited. This change to the structure was noted because of

the possible problems it could create in the design of my program.

2. Requirements

Another avenue of research that I pursued was into a student's requirements for graduation, including their major and basic studies requirements. After the standardization of the basic studies requirements, I found that the hours in each area were not changed yearly. However, I did discover that in some areas classes had been added or deleted from year to year. As an example, under the Humanities section for the school year 1982-1983, a student was offered World Literature I (CMA 205) and World Literature II (CMA 206) as options to fulfill the Literature portion of the Humanities section. In the following year, Introduction to Literature (CMA 203) was added to this list. I found changes similar to this relating to the courses that could apply as a major course. Up until the academic year 1988-1989, a student wishing to receive a Bachelor of Science degree in Mathematics with a major in Computer Science was required to take Introduction to Computer Science (CSC 100). For the school year 1988-1989, this course was deleted from the major requirement and Foundations of Computing (CSC 155) was added as a replacement. Although both courses are similar, their

course descriptions in the catalogs are different. (See Appendix B for examples of both illustrations listed above.) Even though the changes to the major requirements were not as marked as those for basic studies, my findings indicated that because of the changes I would need to evaluate each year separately in the program that I was striving to create.

3. Degrees Offered

Another potential problem I discovered, while researching the requirements for graduation, was that of degrees being added and deleted from the university's selection. Prior to school year 1981-1982, a student could receive a Bachelor of Science degree or an Associate in Applied Science in Home Economics. In the catalog presented for the academic year 1981-1982, this degree plan had been deleted. I also found that during the year 1986-1987, a new degree in the Mathematics department had been introduced. (See Appendix A for the listing of Home Economics and Appendix B for the new degree offered in the Mathematics department.) My findings in this area made me realize that the degrees offered at the university could not be considered as stable, and ,therefore, need to be considered in the program that I was hoping to create.

4. Courses

Performing the necessary research on the courses offered at Pembroke State University was the most tedious and perplexing task of all. While reviewing all of the potential problems that I discovered, I realized that I could distribute the problems into three main areas of concern: courses that could satisfy both basic studies and major requirements, courses which had different titles, and courses which could satisfy several requirements in the basic studies section.

A. Dual Fulfillment

When a student is designing his or her course schedule, he or she would wish to create one that will allow a minimal amount of hours and yet still fulfill both the basic studies and major requirements. I found that in the catalogs that I had evaluated there existed courses that were offered as basic studies courses, and yet were required courses for a particular major. For instance, if a student was interested in receiving a Bachelor of Science degree in Chemistry with a Biomedical Concentration, listed in the catalog for the school year 1988-1989, there are ten courses listed in the Natural Sciences and Mathematics section of the basic studies requirements that the student must fulfill to meet the requirements for this major. (See

Appendix C for the listing of these courses.) A student who was unaware of the rules governing the amount of credit hours necessary to fulfill the requirements for the basic studies and the requirements for the major could try to apply at least four of these courses to fulfill both areas. This problem existed in most majors that I evaluated, although a majority of them did not have as many courses that could apply in both areas as my illustration did. This finding could generate problems in the design of my program, and will need to be considered carefully to avoid any dual assignments of hours because of the problem of fulfilling two areas.

B. Dual Listing

Another area of concern that I found with the courses offered was that some of the courses had two different department headings. For instance, the course History of the American Indian can be listed as either HST 210 or as AIS 210. When evaluating a student's requirements, the course could be listed as HST 210 on the major requirements and as AIS 210 on an official transcript. This could cause some problems to the student if he or she was unaware of the dual listing of the course. Having two different department headings for one course could also create a

problem in the design of my program if I were to ignore the problem.

C. Multiple Fulfillment

The last major problem that I encountered, during my research into the courses offered at Pembroke State University, was that of courses that could be applied to different areas of the basic studies structure to fulfill the requirements. Perhaps the best example of this is the Chancellor's Scholars courses that are offered. Under the 1988-1989 catalog, Current World Problems (CSP 100) can be used to fulfill a Social Science requirement of the basic studies, which does not cause any real concern. The problem became enlarged when I realized that CSP 100 could be used to satisfy any of the five sub-categories listed under Social Sciences. This fact, I knew, would create many problems in the design of my program.

5. Student Data

The last item that I chose to research was the student data. I decided to use the files that were maintained by Pembroke State University, because if this was to be a feasible program, I would need to know if it worked under current conditions. (See Appendix D for a sample of the file structures and a description of the files.) I discovered two shortcomings with the existing files. The

first problem that I encountered was within the file that contained the grade information, and the second problem was discovered in the file that contained the student information.

A. Grade File

The grade file, as described in the appendix, contains information concerning the courses that a student has taken. Within each student record, there are two fields that describe the course that a student has taken. The first field is three bytes long and contains the department that the course is listed under. The second field is also three bytes long, and it contains the course number of the class. This, at first, did not seem to present a problem. But, I found that by using this method, both BIO 100 and BIO 100L would appear to be the same course in the grade file. Because the width of the field is only three characters long, the computer would truncate the 'L' from the course number when the information was entered. Therefore, the only way to distinguish between the two courses would be look at the credit hours received by the student. This alternative, although seeming easy, could cause considerable problems in the design of the program.

B. Student File

The student file contains all of the pertinent

information about a particular student, including the student's major and degree choice. It would seem, then, that it would be a simple task to evaluate the student's major courses, since their major is listed in their record. This was the assumption that I had made, until I realized that some of the degrees offered by the university have both the majors listed alike and the degrees listed alike. For example, a student who wishes to receive a Bachelor of Science degree in Computer Science would have a field in their record that contained a 'BS' to indicate their type of degree. They would also have 'CSC' listed as their major in their record. A student who wished to receive a Bachelor of Science degree in Mathematics with a major in Computer Science would also have the same two codes listed in their record. Although both degrees require many of the same courses, the two are different. Under the present system, there is no clear way of distinguishing between the two degree plans. This information, I realized, would create a major stumbling block in the design of my program.

III. POSSIBLE SOLUTIONS

A. INTRODUCTION

Many of the problems that I uncovered during the research phase of this project have simple solutions, while

many others have difficult ones. The ease of the solution that I generated depended on whether or not it was in my power to change or create the extra data that was needed to correct the problem. I have outlined below solutions to the problems that I illustrated above.

B. SOLUTION TO STRUCTURE

Given the fact that the structure of the basic studies requirements changed prior to the academic year 1982-1983, I had to decide how to handle this occurrence in my program. My solution to this problem was to simply ignore the fact that a student could, in all probability, have an entrance date beyond the academic school year 1982-1983. This is not a very good solution as far as programming goes, but it is the simplest one. I felt that the probability that a student would have an entrance date before 1982 would be very slim, if none. There have been eight academic school years since the structure of the basic studies requirements have been changed. If a student were to take only twelve semester hours per semester, he or she would still have accumulated enough hours to have graduated from Pembroke State University by this time. So, although this is not necessarily the best solution to handle this problem, I felt that it was the simplest and most logical.

C. SOLUTIONS TO REQUIREMENTS

In my research into the requirements that a student would need to graduate from this university, I found out that the required courses could and have changed yearly. Because of the constant changes to the requirements, I needed to come up with a solution that could keep track of previous years requirements and yet be flexible enough to allow for change. The solution that I did come up with was generated during the implementation stage of this project. I decided to create separate files for each basic studies requirements for each year. This solution would cause some repetition; however, it would allow new classes to be inserted or deleted, without losing previous years requirements. The same format was used for the major requirements. The only difference between the two is that a separate file will be kept for each major for each year. This may appear to be a waste of space, considering the fact that the majority of classes do not change from year to year. To verify how much memory would be used using this solution, I came up with some average figures. If you estimate that there will be about 160 courses per year allowed to be designated as a basic studies requirement, and each course in the file occupies 22 bytes, then only 3,520 bytes of memory are used to store the basic studies

requirements for each year. If you estimate that there will be roughly 25 classes per major, at six bytes per class, then each major will only occupy 150 bytes of memory. (See Appendix D for a sample outline of the files that I created.) These amounts of memory used are almost negligible, when you consider the technology today.I did not have to come up with a huge structure that could hold all of the classes that were ever offered at Pembroke State University within my program, I did not have to search through a myriad of courses trying to find the ones I needed, and I had all of the information that I needed to evaluate the requirements for graduation within two files. This solution seemed, to me, the simplest to implement and the simplest to understand.

D. SOLUTIONS TO DEGREES OFFERED

The solution to this problem is simple and straightforward. If a new degree should be added to the curriculum, then all that would be needed to evaluate that degree would be to create a file that contains the courses necessary to fulfill the degree. If a degree were to be dropped from the curriculum, the file containing the major courses would be deleted from memory; after,of course, all students holding that degree plan had graduated. Because of the simplicity of the idea of separate files holding the

courses for each major and degree, this solution was easy to generate and easy to implement.

E. SOLUTIONS TO COURSES

1. Dual Fulfillment

There are two solutions that I came up with to solve the problem of having courses that apply towards the basic studies requirements and the major requirements. The first solution that I thought of, I cannot implement. This would be to set up the requirements in such a way as to have no courses that could be applied to both a major and basic studies. This decision would be a university decision, so I did not try to think of a way to implement this into my design. The second solution that I thought of was encoded into my program. In the first portion of the program, I separate the courses into three categories: basic studies, major courses, and electives. The program then evaluates the basic studies courses. If the basic studies courses are filled, and there are courses left over, then I either apply these to the major, if they qualify, or to the electives. This solution does not always provide the optimal solution, but it does eliminate the problem most of the time.

2. Dual Listing

The best solution to removing courses that have two

department listings, is to have the university decide on one department name and eliminate the dual listing. However, this is not a solution that I can implement. What I chose to do was to list the course under both departments in the files that I had constructed. Both of the different listings would have the same address code listed in the record. (See Appendix E for an example of this . Look at the address codes for HST 210 and AIS 210.) In this way, it would not matter how the course was represented to the student or on the transcript. Handling the matter in this way causes some memory to be wasted, but it saves on the amount of code that would otherwise have to written to handle the problem. In all, I feel that this is the simplest way to solve the problem of having dual department listings.

3. Multiple Fulfillment

The problem of having courses that could fulfill several areas of the basic studies requirements was not an easy one to find a solution for. The best solution that I could come up with is somewhat complicated and involved. When evaluating a student's basic studies courses, I first try to find an area where the course can be used to fulfill the requirement. If the category is filled, then I look at other possible locations of both courses to see if they can

be placed elsewhere. I continue doing this until I have found another category where they can be applied or until I have run out of locations where they can be placed. If I cannot find another category to place the course into, then I check the major file to see if it can be considered towards fulfilling the major requirements. If it can, then it is placed within that area. If it cannot, then the course is listed as an elective, even though that may not be the most advantageous choice for the student. If the course is a Chancellor's Scholars course, and all of the categories where it could be placed are full, then I select one of the categories and place the Chancellor's Scholars course into it, and then check the course that was moved out against the file containing the major courses. If it can be placed within the major listing, it is. Otherwise, the course is considered as an elective. This is not an optimal solution to this problem. If I had had more time, I may have come up with a better solution. This is an area where more research was needed to come up with an optimal solution. This system does work; not very efficiently, of course, but it does work.

F. SOLUTIONS TO STUDENT DATA

1. Grade File

The problem with the grade file was that the field that contained the course number was not long enough. So, the most practical solution would be to extend the field one more byte to allow for the extra character. Another solution would be to add the address code of the course to each student record. This would allow the program to place the class into the proper category without having to view the number of credit hours. However, I could not change the files that I was using; I had to work with the grade file as it existed. The only solution that I could come up with to distinguish between BIO 100 and BIO 100L was to view the credit hours that were given to the student before assigning the course to a category in the basic studies outline. This solution, although not the easiest one, was not hard to implement. I just added an 'IF' statement to my code to check the credit hours received. If the hours field contained '3', then the course was listed as BIO 100. If the hours contained a '1', then the course was listed as BIO 100L. This solution, although not the easiest, solved the problem.

2. Student File

The arrangement of the student files created a problem

with distinguishing between degrees that were similar. The best answer that I came up with to solve this, was to increase the size of the field that contained the degree that the student was working towards, and to restructure the way that the degrees were represented. For example, if a student wanted to receive a Bachelor of Science degree in Mathematics with a major in Computer Science, then have the degree field contain 'BSM' instead of 'BS'. If a student wanted to receive a Bachelor of Science in Computer Science, then the degree field could read 'BSC'. This would eliminate the problem of trying to distinguish between the two degrees. However good I feel that this solution is, I could not change the files that I was working with. In order for me to differentiate between the two degrees, I would have had to look at the courses that the student had previously taken to see which degree plan they were under. Even this solution would not have been too practical, because many of the courses are similar for the two degrees. I simply chose to ignore the Bachelor of Science in Computer Science degree in my program, because of time constraints. Although this is not a valid solution to the problem, the code necessary to separate the two degrees from one another would have been time-consuming and impractical for me to write.

C. ANALYSIS OF SYSTEM

1. JCL Language

The six files necessary to generate the report are stored at the Triangle Universities Computation Center, or TUCC, on the IBM 3081. This mainframe uses the OS370 operating system, with a multi-user batch processing environment. In order to run a program at TUCC, some Job Control Language (JCL) statements must be embedded in the job. The JCL statements define the work to be done, the files needed, and the hardware to be used to the OS370 operating system. In order to use the system at TUCC, the research included learning the necessary JCL code to run a program.

2. Waterloo Pascal

Besides having to learn a little about another programming language, a decision also needed to be made regarding which language to use to write the program. Each language has its own benefits and restrictions. PASCAL was chosen as the language to code the program with. I am familiar with the TURBO version of PASCAL; however, TUCC uses a version of PASCAL called Waterloo PASCAL. This version of PASCAL does not have many of the functions that TURBO does. For example, TURBO has a type called 'string', which is just a string of characters. Waterloo PASCAL does

not have this type; instead, it defines strings as a packed array. There were many small differences such as this that had to be researched into before the implementation stage of the project was reached.

IV. DESIGN

The design stage of this project was the most critical part of the project. Without a good design, the research on the data would have been useless. During the design stage of a project, diagrams and charts are constructed to represent the system as accurately as possible. Throughout the design phase of this project, many standard tools were used to help ease the process. Interviews were conducted, and data flow diagrams, flow charts, and HIPO charts were all drawn to help facilitate the implementation phase. (See Appendix F for these charts and diagrams.)

A. INTERVIEWS

Interviews are a very important part of the design stage. They allow the programmer to find out exactly what the user wants and needs on the final output. Generating the correct questions for an interview is essential to the design stage. Without complete knowledge of the user's expectations, a design can be wonderful, but not to the specifications of the user. Since the first part of this project was for CSC 350, my other group members and I

conducted our formal interviews with Dr. Goldston. He was to receive the final output, so his requirements were what we based our design on. Informally, questions were asked to the staff at the computer center, other professors, students, and aids at the Registrar's Office. The answers received were used to generate the other charts for the design phase.

B. DATA FLOW DIAGRAMS

The next step, after the interviews were conducted, was to create some data flow diagrams for the project. A data flow diagram usually depicts the logical model of a system. Simply put, a data flow diagram shows how the data is used in a system, where it comes from, where it goes, and how it ends up. Based upon the results of the data flow diagrams that the group produced, our project was broken up into four separate programs. The data flow diagrams are excellent tools to help design the physical design of the system.

C. SYSTEM FLOWCHARTS

After the data flow diagrams had been refined, system flowcharts were created. System flowcharts describe the physical system. We began with a very simple flowchart, and added or deleted steps within it until we came up with a system flowchart that represented the best way to

implement our designs. Systems flowcharts, if used correctly, can help to produce a guideline to the coding of the program that will be virtually free of logic errors. Throughout the design phase of the class project for CSC 350, flowcharts were used to depict the overall system, the programs, and all of the procedures within the programs. In this way, the group came up with detailed code for the project that was free of major logic errors, which made the implementation stage much easier to accomplish.

D. HIPO CHARTS

Hierarchy plus Input/Process/Output (HIPO) charts were created next by the group. A Hierarchy chart is a chart which represents the top-down structure of the project. An IPO chart describes the inputs, the processes performed, and the outputs of the program. These charts were used in conjunction with the systems flowcharts to help produce programs and procedures that were well defined. The HIPO charts were also used in the implementation stage to ease the coding aspect of the project.

The tools that were used are very important ones that most analysts and programmers use today. They helped my group from CSC 350 design a system for our class project that was free of any major logic errors. These same tools also helped me to design the rest of this project. There

are other tools that could have been used, such as a feasibility study and a cost analysis, but were excluded because of the nature of this project. I also did not include them in my project because there is no real cost factor involved, and since I believed that this project could be done a feasibility study was unnecessary.

V. IMPLEMENTATION

Once the design phase was completed, the implementation phase was begun. In the implementation phase, the charts and diagrams were used, along with the research generated in the first phase to produce the code for the project. (See Appendix G for the program that was created.) Even though the previous research and design phases were productive and produced many results, there were still decisions that needed to be made regarding this project. The choice of implementation of the program needed to be decided, the internal structure of the files within the program needed to be chosen, and what type of search and sort routines were to be used were all decisions that needed to be discussed.

A. OVERVIEW

This program, as I have written earlier, is designed to read a student's grades, evaluate the courses they have taken, and then print a report showing what courses have

been fulfilled and what courses are still necessary to meet the requirements for graduation. I had originally intended to be able to evaluate the the basic studies courses and the math majors requirements back to 1982, when the basic studies courses were standardized. Because of time considerations, however, the actual program only evaluates the student's courses based on the calendar year 1988-1989. The project is broken down into four steps. the first step of the project separates the classes of each student into three separate files: a file containing the courses which can be considered as basic studies, a file containing those courses that could be used towards a major in Math or Computer Science, and a file which holds those courses treated as electives. The second step of the project removes the courses from the three grade files that are duplicate courses, those courses that the student has taken more than once. The third step of the project removes the students from the files who are not classified as freshmen, sophomores, juniors, and seniors. The final step of the project evaluates the courses and prints the report.

B. CHOICE OF IMPLEMENTATION

Dr. Goldston presented the CSC 350 class with three ways to implement the project. The first method was to upload the project, transmit it to TUCC, and receive a

print-out of the results at the Computer Center at Pembroke State University on the bar-hasp machine. The second method was to use the on-line editor job submission system called WYLBUR, which is maintained at TUCC. The third method was to use WYLBUR in conjunction with a personal computer, modem, and telecommunications software. Since neither I nor my other group members owned a modem, and we all wanted to save ourselves the trouble of uploading to TUCC every time we corrected a mistake, we chose to use the on-line editor called WYLBUR. In order to do so, we had to become familiar with the commands that were offered by WYLBUR. WYLBUR is not very complex to learn; however, at times it is very nerve racking to use. WYLBUR is similar in design to the MS-DOS editor EDLIN. Any changes to the project were done line by line. The cursor cannot be used to scroll through the text, like it can in the PC WRITE editor. However, the one advantage to using this system was that the job could be submitted and the results received at the terminal. Throughout the semester in which we worked on our class project, we used this system in this way. I am still using the WYLBUR editor to submit my jobs; however, I have found it much faster to use another editor to make my changes, upload the changes to TUCC, and then submit my job through WYLBUR.

C. INTERNAL STRUCTURE

The internal structure used to store the data in a program is very important. It determines the amount of memory needed for the program, the degree of complexity of the code, and the run time of the program. There were two main internal structures that the group had debated on using: arrays and linked lists. We finally decided to store our data within the program using arrays. In the first three programs, a record is read from a file, processed, and then written back out to a file. Only one record type was declared to hold the student data temporarily and that record type had a fixed amount of memory. In one of the first three programs, an array was also used to hold the basic studies courses and another array to hold the major courses. The array for the basic studies courses used approximately 3,630 bytes of memory and the array for the major courses used 966 bytes of memory. If a linked list had been used instead of the arrays, the basic studies courses would have used 660 more bytes of memory and the major courses would have used 644 more bytes of memory. Also, by using the array structure for the courses, we were able to perform a binary search on the arrays. In the final program of the project, a record was read from the files, processed, and then printed. For

the basic studies evaluation, a record was used to hold the class information and a couple of small arrays were used to hold the major information and the classes that could be considered as electives. If linked lists had been used within the final program of the project, four extra bytes of memory per class would have been used for each student processed. By implementing arrays instead of linked lists, the code was less complex because there was no need to keep track of pointers. Also, by using arrays, the run time was reduced slightly, since linked lists with pointers take longer to run. The choice to use arrays as the internal structure for our data, opposed to a linked list, saved about 1,300 bytes of memory in one program and about 160 bytes of memory per student in the final program. By using arrays, the run time was reduced and the code was less complex.

D. SEARCH AND SORT ROUTINES

The type of search and sort routines that are used during a program are very important aspects to consider when writing a program. There are various sort routines that are available to use, and each one has different rules and regulations governing them. However, we chose to use the sort routine that was provided by the operating system. All that was needed was to specify which file was to be

sorted, and the locations of the fields that the files were to be sorted by. By using the utility sort provided by the system, we reduced the amount of code that was written considerably, since the sort routine was called eight times in the project. The amount of comparisons made in a search routine can add considerably to the run time of a program. There are two main types of search routines that we considered. The first is a linear search, which starts at the beginning of the array and compares each item in the array until it finds what it is looking for. The second type of search routine is a binary search which compares the item you are looking for to the element in the middle of the list. the routine then looks at the bottom or the top half of the list for the item. The process of looking at the middle item in the list and then reducing the list by half is repeated until the item is found or there are no more elements to compare it to. On average, a linear search makes about N/2 successful searches, where N is the number of elements in the array. A binary search, however, makes an average of lg N + 1 searches or 2lg N - 3 searches, depending on which form of the binary search was chosen. we chose to use a form of the binary search called the 'Forgetful version', which takes an average of lg N + 1 comparisons to find an item in an array. The searches were

made on the two arrays which held the basic studies courses and the major courses. If a linear search had been used, an average of 81 comparisons would have been made. If the other version of binary search had been used, an average of 13.6 comparisons would have been made. By using the forgetful version of the binary search, the average number of comparisons was reduced to 8.3 . By using the utilities offered by the system and using the most efficient search routine, the amount and complexity of the code was reduced and the run time of the project was reduced considerably.

VI. ANOTHER ALTERNATIVE

A. INTRODUCTION

Although creating a program to write a report can be fairly easy, there are other alternatives to this decision. One alternative would be to restructure the entire computer system at Pembroke State University, and model it into a database management system. If a full-scale system was implemented, creating a report would be an extremely easy task to accomplish. A database management system would not be able to evaluate the courses that a student had taken; but, it would be able to print all of the courses that the student had taken. The task of evaluating all of the courses would still need to be done by a person in authority, such as the student's counselor. However, a

program could be written to evaluate the courses, such as the one I had written, with possibly less code than I had used. In order to create a listing of all of the courses, the user would only have to define the fields that he or she would want to appear on the report, give any special sorting instructions to the computer, and then send the request to the printer. In a matter of seconds, or perhaps minutes, the list of courses would be completed and printed with very little trouble involved.

B. DATABASES

Before a database management system could be implemented, though, there would need to be a database. In this case, it would be fairly easy to construct the database, because the data is already in memory. Databases are little more than a collection of related records that have been stored on some type of computer-based medium. As simple as the concept of a database may seem, there are two major considerations to be made when creating a database. The database should have a sound internal structure and be independent from any programming language.

1. Internal Structure

In creating the database, much thought must be given to how the information will be structured in memory. There are three main models in use today: the relational model,

the network model, and the hierarchial model. The relational model is, essentially, a collection of tables. Within each table, there is a field, or column, that can be found within another table, creating a "relationship" between the tables. This key field is usually a variable that changes little or not at all, such as a social security number. The network model is perceived to be a collection of records and the relationships between the records. The user must first locate one record within a file and then locate another record in another file based on the relationship between the records. The hierarchial model can be perceived as a collection of hierarchies. This model is similar to the network model, but much more restrictive in its structuring. Illustrations of the three models follow.

THE RELATIONAL MODEL:

STUDENT FILE :	Student ID	Name	Address	Major

GRADE FILE :	Student ID	Dept.	Course #	Grade

PERSONAL INFORMATION:	Student ID	Home Address	Phone #

database, is the type of languages that will be used to manipulate the database. The database itself should not be designed to be dependent on any language. In this way, a program to manipulate the database can be written in any language that best fits the application. For instance, if I had designed a database in such a way that the name field within a record was only as long as the actual name contained in it, I would not be able to use PASCAL to manipulate my database. The PASCAL language does not always support variable length records. In doing so, my database would become dependent on those languages, such as COBOL, that do support variable length records. Therefore, the best design should be one that is totally independent of any language.

C. DATABASE MANAGEMENT SYSTEMS

Having a good, well-structured database is not enough; there must be some way to manipulate the data stored within the database. The need to get at the information within the databases caused the database management system to be created. A database management system is, generally, a system which helps to coordinate and manage the database. A full-scale database management system should provide at least eight basic functions : data storage, retrieval, and update, a user-accessible catalog,

transaction support, concurrency control services, recovery services, authorization services, support for data communication, and integrity services. Besides these eight functions, many companies also include services to promote data independence and utility services in their database management systems.

VII. CONCLUSION

While working on this project, I had several small setbacks and one major setback to achieving my goals. As I have explained earlier, my program is set up to evaluate only one year of basic studies and Math and Computer Science majors. My original intention had been to evaluate the student's basic studies courses based upon the catalog year of their entrance date into Pembroke State University, and to evaluate those students who held a major in Math or Computer Science in the same way. Because of the missing data in the student files, I am unable to achieve this goal.

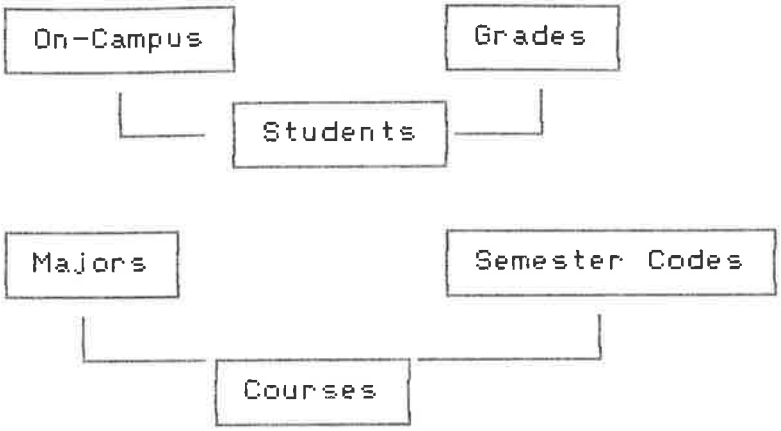
Throughout the time spent on this project, I learned many things. I learned a new programming language, how to adjust to a different version of a familiar programming language, how to work with another editor, and how to work within a different type of computing environment. But, besides these academic lessons, I learned some of life's

little lessons. One lesson learned was that of setting and achieving goals. Another lesson I learned was that I was a mere mortal and therefore prone to make errors, of which I made plenty. But, I think the most valuable lesson of all, was that if you own a puppy, not to place your disks within reach of that puppy. I survived the chewed up shoes, but almost did not survive the chewed up disk. Because of an earlier mistake on my part, no backup disk, I learned to make backups and saved myself from total ruination. All in all, it has been an experience I am not likely to forget.

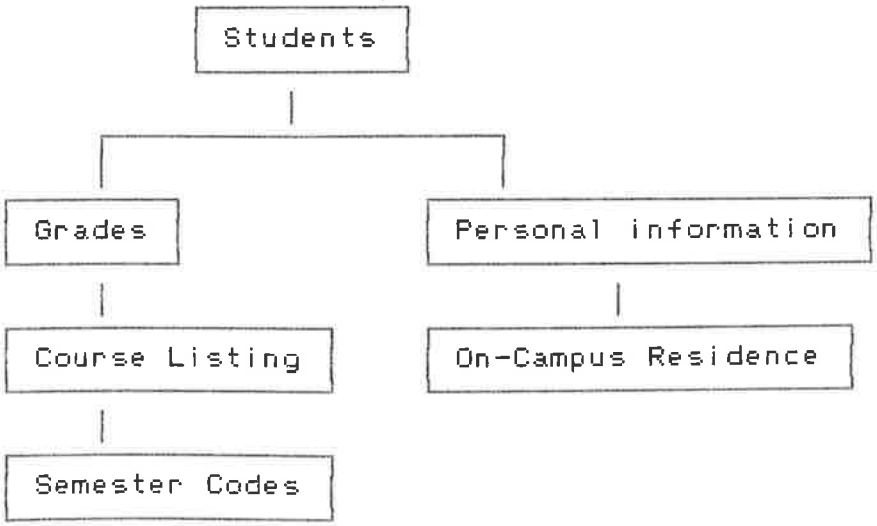
MAJOR : | Dept. | Course # | Credit Hours | Semester |

ON-CAMPUS RESIDENCE : | Student ID | Dorm | Room # |

THE NETWORK MODEL:



THE HIERARCHIAL MODEL:



2. Language Independence

Another consideration to be made when designing the

ACKNOWLEDGEMENTS

I would like to thank all of the people who I have "bugged" during the duration of this project, because without them I would not have been able to complete this project. I would also like to give some credit to the two other members of my group from CSC 350, who participated in part of the research , design, and coding of my project. These two people are Phil Saunders and John Hayes. I would also like to thank Dr. Goldston and Mr. Thomas Jackson for their help, guidance, and patience.

APPENDIX A

This appendix contains pages duplicated from the catalogs that Pembroke State University distributed in 1980-1981 and 1982-1983. This illustrates the differences between the requirements for the Humanities section of the basic studies requirements from onw year to the next.

V. CURRICULA AND DEGREES

Pembroke State University operates on the traditional two semester system and offers an extensive summer program designed to permit the academic acceleration of regular university students and to serve the needs of public school teachers. The Summer Session is divided into two terms. Information concerning enrollment procedures and costs may be obtained from the Office for Academic Affairs.

The University offers various liberal arts programs leading to the Bachelor of Arts or Bachelor of Science degree, programs for teacher certification in several fields, and special two-year curricula for students who intend to transfer to professional schools. Candidates who successfully complete the university requirements in the following fields receive the Bachelor of Arts degree:

- Art
- Economics
- English
- History
- Music
- Philosophy and Religion
- Political Science
- Sociology

Candidates who successfully complete the university requirements in the following fields receive the Bachelor of Science degree:

- Art Education
- Biology
- Business Administration
- Business Education
- Chemistry
- Elementary Education
- Home Economics
- Mathematics
- Music Education
- Physical Education
- Psychology
- Reading Education
- Special Education

Candidates who hold the Associate in Applied Science Degree and who successfully complete the additional university requirements receive the Bachelor of Science in Applied Science degree.

The Division of Professional Services of the State Department of Public Instruction has approved the following teacher certification programs at Pembroke:

- Elementary School
  - K-3 (Early Childhood Certificate)
  - 4-9 (Intermediate Grades Certificate)
- Secondary School
  - Business Education
  - English
  - Home Economics
  - Mathematics
  - Science
    - Biology
  - Social Studies
  - Reading Education
  - Special Education
  - Special Subjects (Grades 1-12)
    - Art
    - Music
    - Physical Education and Health

MINIMUM BASIC STUDIES REQUIREMENTS

Graduation from Pembroke State University is based upon successful completion of the Basic Studies Program which is required for all degrees and upon successful completion of a specialized program for a major.

1980-1981

B.A. OR B.S. DEGREE

A student must complete forty-five (45) semester hours of course credit in 100-level and 200-level designated courses in the three basic disciplines which are the:

I. Humanities
II. Social and Behavioral Sciences
III. Natural Sciences and Mathematics

The manner in which these forty-five semester hours of credit is to be distributed is indicated below:

I. Humanities—A minimum of twenty-four (24) semester hours from the areas of:

A. English (6 semester hours of Composition required)
B. Fine Arts (Art, Music, Dramatics)
C. Foreign Language
D. History
E. Philosophy and Religion

Credit must be earned in three of the five areas under the Humanities with a minimum of six semester hours in each of the three areas selected.
All full-time students must enroll in Composition I during their first semester of full-time study and continue consecutively thereafter until they successfully complete both CMA 105 and CMA 106, the six-hour required sequence.
An English proficiency test is required of certain students, see page 50.
Six semester hours of 200-level literature and six semester hours of 100 or 200-level history are required for teacher certification.

II. Social and Behavioral Sciences—A minimum of nine (9) semester hours from the areas of:

A. Business Administration
B. Economics
C. Geography
D. Political Science
E. Psychology
F. Sociology

Credit must be earned in three of the six areas under the Social and Behavioral Sciences.

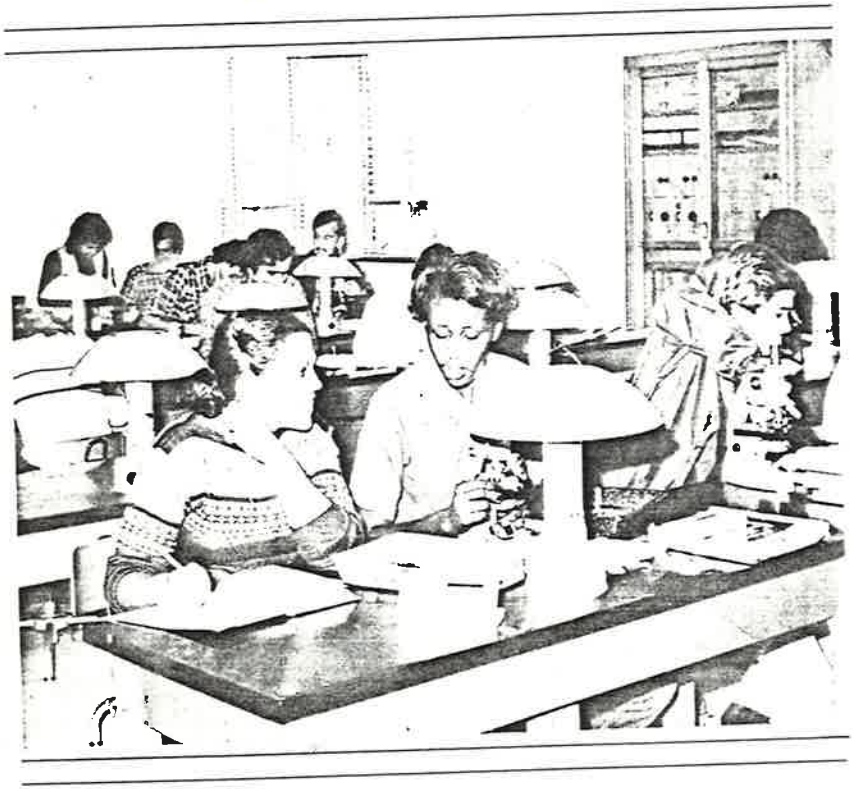
III. Natural Sciences and Mathematics—A minimum of twelve (12) semester hours, with at least 3 semester hours in each area.

A. Biological Science
B. Mathematics
C. Physical Science or Geology

Credit must be earned in all three areas under the Natural Sciences and Mathematics.

REQUIREMENTS FOR MAJORS, MINORS AND SPECIAL PROGRAMS

A major field of study consists of not less than thirty semester hours of course credit earned in that field, at least fifteen of which must be in courses numbered above 299. Detailed requirements for majors have been established and are printed in the department section of the catalog.



1982-1983

BASIC STUDIES REQUIREMENTS
(50 Hours Total)

A. **Basic Skills** (6 hours)

CMA 105 Composition I
CMA 106 Composition II — 6 hours

A full-time student must enroll in CMA 105 - Composition I - immediately.

If the student does not successfully complete CMA 105, the student must enroll in CMA 105 again in the next semester (and in each semester following until CMA 105 is successfully completed.)

A full-time student who has successfully completed CMA 105 must immediately enroll in CMA 106 - Composition II.

If the student does not successfully complete CMA 106, the student must enroll in CMA 106 again in the next semester (and in each semester following until CMA 106 is successfully completed.)

A student who has received credit for CMA 105 and CMA 106 is exempt from the requirements above; there is an English proficiency requirement, however, for certain students. Please see REQUIREMENTS FOR GRADUATION, page 51.

B. **Humanities** (18 hours)

(1) Fine Arts — 3 hours

Choice of *one* course from the following:

ART 205	Art Appreciation
CMA 250	Introduction to the Theatre
MUS 230	Music Appreciation
MUS 295	Music History and Literature I

(2) Literature — 3 hours

Choice of *one* course from the following:

CMA 205	World Literature I
CMA 206	World Literature II

Six semester hours of 200-level literature courses are required by Pembroke State University for any teacher certification candidate.

APPENDIX B

This appendix contains pages duplicated from the catalogs that Pembroke State University distributed in 1982-1983, 1983-1984, 1987-1988, and 1988-1989. These illustrate the differences between the requirements for the Humanities section of the basic studies requirements from one year to the next. This also illustrates the addition of a major in the Mathematics department.

1982-1983

BASIC STUDIES REQUIREMENTS
(50 Hours Total)

A. **Basic Skills** (6 hours)

CMA 105	Composition I	
CMA 106	Composition II	6 hours

A full-time student must enroll in CMA 105 - Composition I - immediately.

If the student does not successfully complete CMA 105, the student must enroll in CMA 105 again in the next semester (and in each semester following until CMA 105 is successfully completed.)

A full-time student who has successfully completed CMA 105 must immediately enroll in CMA 106 - Composition II.

If the student does not successfully complete CMA 106, the student must enroll in CMA 106 again in the next semester (and in each semester following until CMA 106 is successfully completed.)

A student who has received credit for CMA 105 and CMA 106 is exempt from the requirements above; there is an English proficiency requirement, however, for certain students. Please see REQUIREMENTS FOR GRADUATION, page 51.

B. **Humanities** (18 hours)

(1) Fine Arts — 3 hours

Choice of *one* course from the following:

ART 205	Art Appreciation
CMA 250	Introduction to the Theatre
MUS 230	Music Appreciation
MUS 295	Music History and Literature I

(2) Literature — 3 hours

Choice of *one* course from the following:

CMA 205	World Literature I
CMA 206	World Literature II

Six semester hours of 200-level literature courses are required by Pembroke State University for any teacher certification candidate.

1983-1984

BASIC STUDIES REQUIREMENTS

(50 Hours Total)

A. **Basic Skills** (6 hours)

CMA 105 Composition I
CMA 106 Composition II — 6 hours

A full-time student must enroll in CMA 105 - Composition I — immediately and must earn a "C" grade or better before enrolling in CMA 106.

If the student does not successfully complete CMA 105, the student must enroll in CMA 105 again in the next semester (and in each semester following until CMA 105 is successfully completed.)

A full-time student who has successfully completed CMA 105 must immediately enroll in CMA 106 - Composition II and must earn a "C" grade or better.

If the student does not successfully complete CMA 106, the student must enroll in CMA 106 again in the next semester (and in each semester following until CMA 106 is successfully completed.)

A student who has received credit for CMA 105 and CMA 106 is exempt from the requirements above.

B. **Humanities** (18 hours)

(1) Fine Arts — 3 hours
Choice of *one* course from the following:

ART 205 Art Appreciation
CMA 250 Introduction to the Theatre
MUS 230 Music Appreciation
MUS 295 Music History and Literature I

(2) Literature — 3 hours
Choice of *one* course from the following:

CMA 203 Introduction to Literature
CMA 205 World Literature I
CMA 206 World Literature II

Six semester hours of 200-level literature courses are required by Pembroke State University for any teacher certification candidate.

(3) History — 3 hours
Choice of *one* course from the following:

HST 101 American Civilization to 1865

Chairman prior to registering for Basic Studies courses. All majors choose an advisor and are urged to consult with them periodically in order to plan and carry out their program of study.

Most non-majors fulfill the Basic Studies requirement in mathematics by taking one of MAT 101, 105, 107 or 108. Well-prepared students may select MAT 109 or 221 for this purpose.

Requirements for a Bachelor of Science Degree in Mathematics: Major in Mathematics

	Sem. Hrs.
Orientation Requirement (p. 104)	1
Basic Studies Requirements (p. 60)*	50
Major Requirements	
MAT 107 and 108, or MAT 109, or equivalent courses in secondary school; also MAT 220, 221, 222, 315, 316, 325, 431, and nine additional semester hours of advanced mathematics.	33-39
Electives	38-44
Total	128

Requirements for a Bachelor of Science Degree in Mathematics with Certification by the State to Teach Mathematics at the Secondary Level

	Sem. Hrs.
Orientation Requirement (p. 104)	1
Basic Studies Requirements (p. 60)*	50
Major Requirements	
MAT 107 and 108, or MAT 109, or equivalent courses in secondary school; also MAT 220, 221, 222, 250, 315, 316 325, 411, 431, and six additional semester hours of advanced mathematics	35-41
Professional Education Requirements	
EDN 202, 227, 308, 419, 437, 445, 448, & MAT 400	25
Electives	11-17
Total	128

Requirements for a Bachelor of Science Degree in Mathematics: Major in Computer Science

	Sem. Hrs.
Orientation Requirement (p. 104)	1
Basic Studies Requirements (p. 60)*	50
Major Requirements	
MAT 107 and 108, or MAT 109, or equivalent courses in secondary school; also MAT 221, 222, 315, 316; and three semester hours selected from MAT 317, 325, 328, 329; CSC 100**; CSC 200 or 201; also CSC 250, 270, 350, 420, 450; CSC 210 or MAT 330; and three semester hours selected from CSC 370, 400, 410, or MAT 327	45-52
Electives	25-32
Total	128

*Students who plan a major in mathematics should consult with the Department Chairman prior to registering for Basic Studies courses.
**CSC 100 may be waived at the discretion of the Chairman of the Department based on the work or academic experience of the student.

Requirements for a Bachelor of Science Degree in Computer Science

Orientation Requirement (p. 104)	1
Basic Studies Requirements (p. 60)*	50
Major Requirements	
CSC 155, 215, 255, 275, 325, 355, 385, 455	
CSC 305 (grade of 'B' or better)	
CSC 415 or CSC 435	
MAT 109 or equivalent, MAT 221, 222, 315, 328	
one of CSC 445, MAT 327, 330	
one additional 400 level CSC course	
an additional 12 hours from one of the following categories:	
I. CSC 415, 425, 435, 445, 495, 496, MAT 327	
II. MAT 316, 322, 325, 330, 415, 431	
III. BUS 227, 228, 307, 308 CMA 101	
IV. CHM 100, 101, PHY 150, 151, 256, 320	62-66
Electives	11-15
Total	128

Requirements for a Minor in Mathematics

	Sem. Hrs.
MAT 221, 222, 315, 316; and three additional hours selected from advanced mathematics courses (300 or above).	18

Most non-majors fulfill the Basic Studies requirement in mathematics by taking one of MAT 101, 105, 107 or 108. Well-prepared students may select MAT 109 or 221 for this purpose.

Requirements for a Bachelor of Science Degree in Mathematics: Major in Mathematics

	Sem. Hrs.
Orientation Requirement (p. 108)	1
Basic Studies Requirements (p. 66)*	50
Major Requirements	
MAT 107 and 108, or MAT 109, or equivalent courses in secondary school; also MAT 220, 221, 222, 315, 316, 325, 431, and twelve additional semester hours of advanced mathematics	36-42
Electives	35-41
Total	128

Requirements for a Bachelor of Science Degree in Mathematics with Certification by the State to Teach Mathematics at the Secondary Level

	Sem. Hrs.
Orientation Requirement (p. 108)	1
Basic Studies Requirements (p. 66)*	50
Major Requirements	
MAT 107 and 108, or MAT 109, or equivalent courses in secondary school; also MAT 220, 221, 222, 315, 316 325, 411, 431, and nine additional semester hours of advanced mathematics	36-42
Professional Education Requirements	
EDN 202, 227, 308, 419, 437, 445, 448; MAT 250, 400	27
Electives	8-14
Total	128

*Students who plan a major in mathematics should consult with the Department Chairman prior to registering for Basic Studies courses.

Requirements for a Bachelor of Science Degree in Mathematics: Major in Computer Science

	Sem. Hrs.
Orientation Requirement (p. 108)	1
Basic Studies Requirements (p. 66)*	50
Major Requirements	
MAT 107 and 108, or MAT 109, or equivalent courses in secondary school; also MAT 221, 222, 315, 316; and three semester hours selected from MAT 317, 325, 328, 329; CSC 155; CSC 200 or 201; also CSC 250, 270, 350, 420, 450; CSC 210 or MAT 330; and three semester hours selected from CSC 370, 400, 410, or MAT 327	45-52
Electives	25-32
Total	128

*Students who plan a major in mathematics should consult with the Department Chairman prior to registering for Basic Studies courses.

1988-1989

New degree

Requirements for a Bachelor of Science Degree in Computer Science

Orientation Requirement (p. 108)	1
Basic Studies Requirements (p. 66)*	50
Major Requirements	
CSC 155, 215, 255, 275, 325, 355, 385, 455 CSC 305 (grade of 'B' or better) CSC 415 or CSC 435 MAT 109 or equivalent, MAT 221, 222, 315, 328 one of CSC 445, MAT 327, 330 one additional 400 level CSC course an additional 12 hours from one of the following categories: I. CSC 415, 425, 435, 445, 495, 496, MAT 327 II. MAT 316, 322, 325, 330, 415, 431 III. BUS 227, 228, 307, 308 CMA 101 IV. CHM 100, 101, PHY 150, 151, 256, 320	62-66
Electives	11-15
Total	128

Requirements for a Concentration in Mathematics for the B.S. in Middle Grades Education (6-9).

	Sem. Hrs.
MAT 101, 102, 107 and 108 or 109, 201, 300, CSC 100	19-21
Two courses from: MAT 118, 210, 220, 221, CSC 200, 201, 405 one of which must be a MAT course.	6-8
Total	25-29

Requirements for a Minor in Mathematics

	Sem. Hrs.
MAT 221, 222, 315, 316; and three additional hours selected from advanced mathematics courses (300 or above).	18

Requirements for a Minor in Computer Science

	Sem. Hrs.
CSC 100; CSC 200 or 201; CSC 250, 270; CSC 350 or 370; and six additional hours selected from CSC 200, 201, 210, 350, 370, 400, 410	18-21

*Students who plan a major in Computer Science should consult with the Department Chairman prior to registering for Basic Studies courses.

APPENDIX C

This appendix contains pages duplicated from the catalogs that Pembroke State University distributed in 1988-1989. These pages were used to illustrate a point concerning dual fulfillment of courses. There are ten courses that a student needs to satisfy the major and which can also apply towards basic studies.

PHYSICAL SCIENCE

Chairman: JOSE J. D'ARRUDA

JOHN E. REISSNER
HAROLD J. TEAGUE
TODD THORNTON
JOHN S. WALLINGFORD
PETER WISH

The Physical Science Department currently offers a Bachelor of Science Degree with a major in Chemistry, including a Biomedical Concentration and a Concentration in Medical Technology. Students who have completed the program have been successful at entering professional schools, gaining employment in government and industry, as well as pursuing graduate studies in chemistry.

The Physical Science Department also offers a pre-engineering program. This program has been approved by the Subcommittee on Engineering programs at North Carolina A & T State University, North Carolina State University, and the University of North Carolina at Charlotte.

Requirements for a Bachelor of Science Degree in Chemistry

	Sem. Hrs.
Orientation Requirement (p. 108)	1
Basic Studies Requirements (p. 66)*	50
Major Requirements	
CHM 100, 101; 200, 201; 300, 301; 410; 498	28
Elective in Chemistry (above 299)	3
PHY 150, 151 or 200, 201	6
MAT 107, 108 or equivalent; 221, 222	8-14
Electives	26-32
Total	128

*Students who plan a major in Chemistry should request an advisor in the Physical Science Department and consult with that advisor before registering for Basic Studies courses.

Requirements for a Bachelor of Science Degree in Chemistry with a Biomedical Concentration

First Year

Fall			Spring		
CHM 100,	General Chemistry	4	CHM 101,	General Chemistry	4
CMA 105,	Composition I	3	CMA 106,	Composition II	3
MAT 107,	College Algebra	3	MAT 108,	Plane Trigonometry	3
BIO 100,	Principles of Biology	4	BIO 102,	General Zoology	4
Basic Studies	(Physical Edn)	1	Basic Studies	(Physical Edn)	1
ORI 100,	University Orientation	1			
		16			15

Second Year

Fall			Spring		
CHM 200,	Qualitative Analysis	4	CHM 201,	Quantitative Analysis	4
PHY 150,	Introductory Physics	3	PHY 151,	Introductory Physics	3
PHY 206,	Physics Laboratory	1	PHY 207,	Physics Laboratory	1
BIO 211,	Human Anat. Physiol.	4	BIO 212,	Human Anat. Physiol.	4
MAT 210,	Intro. to Statistics	4	Basic Studies	(Humanities)	3
			Basic Studies	(Social Science)	3
		16			18

Third Year

Fall			Spring		
CHM 300,	Organic Chemistry	4	CHM 301,	Organic Chemistry	4
BIO 221,	Comparative Anatomy	4	CHM 498,	Literature Seminar	1
BIO 371,	Cell Biology	4	BIO 315,	Microbiology	4
Basic Studies	(Humanities)	6	Basic Studies	(Humanities,	6
				Advised Electives	3
		18			18

Fourth Year

Fall			Spring		
CHM 311,	Biochemistry	3	CHM 327,	Instrumental Analysis	3
BIO 427,	Principles of Genetics	4	Basic Studies	(Social Science)	6
Basic Studies	(Social Science)	3		Advised Electives	4
Basic Studies	(Humanities)	3			
	Advised Electives	1			
		14			13

Area 4. Philosophy and Religion

AIS	201	American Indian Culture
PHI	100	Introduction to Philosophy
PHI	101	Introduction to Logic
PHI(REL)	102	Perspectives on Man
PHI	202	Philosophy of Religion
PHI	204	Introduction to Ethics
PHI	205	Social and Political Philosophy
PHI	211	American Philosophy
REL	105	Survey of Old Testament
REL	106	Survey of New Testament
REL	130	Introduction to Religion
REL	209	Religion in America
REL(AIS)	213	American Indian Traditions
REL	214	Introduction to Religious Ethics
REL	216	Religions of the Far East
REL	218	Religions of the Near East

Area 5. (Chancellor's Scholars only)

CSP	200	Great Cultural Epochs I —will receive credit for any course in Section (5) of Humanities.
CSP	201	Great Cultural Epochs II —will receive credit for any course in Section (5) of Humanities.

C. **Social Science** (12 hours)

Each student must earn three hours in **four** out of **five** of the following disciplines, chosen from the courses listed below:

(1) Economics

ECN	201	Principles of Economics I

(2) Geography

GGY	101	Introduction to Geography
GGY	102	World Regional Geography
GGY	200	Cultural Geography
GGY(ECN)	206	Economic Geography

(3) Political Science

PLS	100	Introduction to Political Science
PLS	101	Introduction to American National Government

(4) Psychology

PSY	101	Introductory Psychology

(5) Sociology

SOC	101	Introduction to Modern Sociology
SOC	105	Introduction to Cultural Anthropology
SOC	201	Sociological Concepts

Chancellor's Scholars may substitute these courses for courses in any two disciplines:

CSP	100	Current World Problems —will receive credit for any course in Social Science.
CSP	275	Individual and Collective Man —will receive credit for any course in Social Science.

D. **Natural Sciences and Mathematics** (12 hours)

(1) Biology

BIO	100	Principles of Biology

(2) Physical Science

Choice of *one* course from the following:

CHM	100	General Chemistry
PHS	110	Physical Science I
PHS	111	Physical Science II
PHY	100	Elementary Physics I
PHY	150	College Physics I
PHY	200	University Physics I

Chancellor's Scholars only:

CSP	250	Structures of the Universe —will receive credit for any course in Section (2) of Natural Science and Mathematics.

(3) Mathematics

Choice of *one* course from the following:

MAT	101	Mathematics for Elementary School Teachers
MAT	105	Introduction to College Mathematics
MAT	107	College Algebra
MAT	108	Plane Trigonometry
MAT	109	College Algebra and Trigonometry
MAT	221	Calculus I

(4) Divisional Elective

Choice of one additional course to complete the 12 hour requirement.

BIO	100	Principles of Biology
BIO	100L	Laboratory Investigations and Experiences in General Biology
BIO	101	General Botany
BIO	102	General Zoology
BIO	103	Basic Human Biology
BIO	201	Economic Botany
BIO	207	History of Biology
BIO	210	Conservation and Environment
CHM	100, 101	General Chemistry
CSC	200	Intro. to Computer Programming—FORTRAN
CSC	201	Intro. to Computer Programming—COBOL
CSC	210	Introduction to Statistics

GLY 100	Introduction to Physical Geology
GLY 115	Introduction to Earth Science
MAT 101, 102	Mathematics for Elementary School Teachers
MAT 105	Introduction to College Mathematics
MAT 107	College Algebra
MAT 108	Plane Trigonometry
MAT 109	College Algebra and Trigonometry
MAT 118	Finite Math
MAT 210	Introduction to Statistics
MAT 221, 222	Calculus I, II
PHS 110, 111	Physical Science I, II
PHS 116	Exploring Man's Energy Choices
PHS 156	Astronomy
PHY 100, 101	Elementary Physics I, II
PHY 115	Electronics
PHY 150, 151	College Physics I, II
PHY 200, 201	University Physics I, II

Chancellor's Scholars only:

CSP 240	Practices and Trends in Computer Applications —will receive credit for any course in Section (4) of Natural Science and Mathematics.

E. **Physical Education** (2 hours)

Each student is required to complete *two* of the following courses:

PED 131	Archery
PED 132	Badminton
PED 133	Beginning Golf
PED 134	Beginning Swimming
PED 135	Beginning Tennis
PED 137	Bowling
PED 138	Folk Dancing
PED 139	Racquetball
PED 140	Intermediate Swimming
PED 141	Physical Conditioning
PED 142	Social Dance
PED 143	Stunts and Tumbling
PED 144	Trampoline
PED 145	Volleyball
PED 146	Weight Training
PED 149	Scuba Diving
PED 171	Intermediate Tennis
PED 172	Advanced Lifesaving
PED 175	Athletic Ballet
PED 176	Intermediate Golf
PED 177	Intermediate Weight Lifting
PED 178	Advanced Physical Conditioning
PED 179	Aerobic Fitness
PED 180	The Art of Self Defense

REQUIREMENTS FOR A DOUBLE MAJOR

A student may elect to earn majors in two separate disciplines on the condition that the student meets all requirements for each major. The student who completes requirements for more than one major will receive only one degree, but at the time of initial graduation, the record will indicate both majors.

REQUIREMENTS FOR A SECOND BACCALAUREATE DEGREE

A student with a bachelor's degree may receive a second baccalaureate degree if it is a different degree and a different major by fulfilling the following requirements:

(1) The student must meet all the requirements for the second degree and major.
(2) The student must complete a minimum of 30 hours in residence beyond the requirements for the first degree.

REQUIREMENTS FOR A MINOR

A recognized minor should ordinarily consist of 18 to 21 semester hours of courses. With the approval of the department granting the minor, up to six hours of the courses counted toward a minor may be used to satisfy Basic Studies, major requirements, or requirements of an additional minor. The award of a minor will require formal approval of the department concerned. Successful completion of a minor will be noted on the student's official transcript. Student participation in minor programs will be optional.

The following Departments offer minors: [A list of minors appears on p. 1]

American Indian Studies
Art
Business Admin. and Economics
Communicative Arts
Geology and Geography
Health, Physical Education and Recreation
History
Mathematics and Computer Science
Music
Philosophy and Religion
Political Science
Psychology
Sociology and Social Work

In addition, two interdepartmental minors are available:

World Studies (See p. 291)
Personnel and Organizational Leadership (See p. 288)

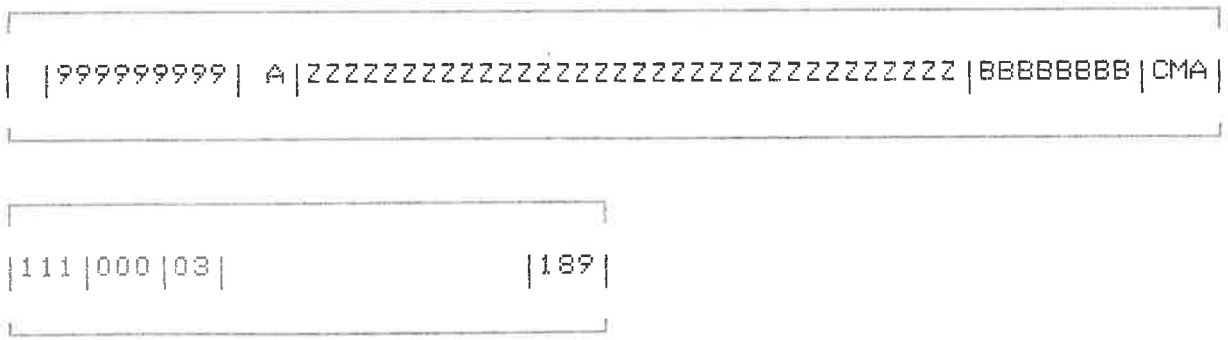
CHANCELLOR'S SCHOLARS PROGRAM

The Chancellor's Scholars Program is designed to recognize outstanding students and to promote the scholarly growth of the students selected for the program by providing interdisciplinary educational opportunities not necessarily available in the general curriculum programs. These distinctive opportunities include: small interdisciplinary seminars; an intellectually

APPENDIX D

FILE DESCRIPTION

EPSJWG.CSC350.GRADE is a file that is used in one of the programs of this project. It resides at TUCC. The file is 80 bytes long, and contains the social security number, grade,student name,professor name, department code,course number, section number, credit hours, and course semester code for every grade ever given to every student. For this project, there are approximately 675 records in the file. A layout of this file is printed below.



The first field used contains the social security number of the student (999999999). The second field contains the grade that the student received for the course (A). The third field contains the student's name (ZZZ...). The fourth field contains the professor's name (BBB...). The fifth field contains the department code for the class (CMA). The sixth field contains the course number of the class (111). The seventh field contains the section number for the course (000). The eigth field

contains the number of credit hours that the student received for the course (03). The eigth field contains blanks, because this information was not available to me. The ninth field contains the semester code for the semester that the course was taken in (189).

The file EPSJWG.CSC350.DUMDBAS contains all of the information about the student, such as the home address, the phone number, the high school that he or she graduated from, etc.. EPSJWG.CSC350.DUMDBAS contains 320 bytes per record. For the project, there is an estimated 65 records. A format of the file is printed below.

```
| |999999999|ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ|W|F|012089|

|121264|M|U|CU|NC|1|ZZZZ|TX|2|Y|S|D|0|BBBBBBBB|CSC|BS|999|

|888|10|3200|3500|3500|3500|CCCCCC|FFFFFFFF|1111|22|Y|Y|

|88|1|N|4|A|E|N|24000|24000|00000|99999|8888|77777|66666|

|55555|44444|3333|Y|222222|111111|S|J|J|ZZZZZZZZZZZZZZZZZZ|

|AAAAAAAAAAAAAAAA|BBBBBBBBBBBBBBBB|CC|09876|1234567|C|
```

The first field used contains the social security number of the student (999999999). The second field used contains the student's name (ZZZ..). The third field that was used in the program is the entrance date (012089). The marital status was also used in the printout of the report (M). The field that contains the student's disposition is used during the evaluation (1). The field that contains a character to indicate whether the student is a veteran or not is used on the printout (Y). The student's advisor's name is used in the program (BBB...). The field that contains the student's major (CSC) and degree (BS) are used to help evaluate the courses. SAT verbal (999) and SAT math (888) scores are used for the printout of the student. The field that contains teacher certification (Y) is used on the printout. Other fields that are printed on the report include the year graduated from high school (88), the transfer hours passed (2400), the cumulative hours attempted (77777), the cumulative hours passed (66666), and the cumulative gpa (3333).

The file EPSGPC.BSTUDY was created by the group. Each record contains the department code,course number,the number of locations, and five location codes for every course that could be considered a basic studies course. A format of the file is listed below.

999	888	1	B5b	B2			

The first field contains the department code (999). The second field contains the course number (888). The third field contains a number which indicates how many areas within the basic studies outline where the course can apply (1). The fourth through the eigth field contain the address codes for the course (B5b).

The file EPSGPC.MAJOR was created by me to hold a listing of the courses which could be considered as major courses. Each record contains the department code and the course number of the course. A file format is printed below.

999	888

APPENDIX E

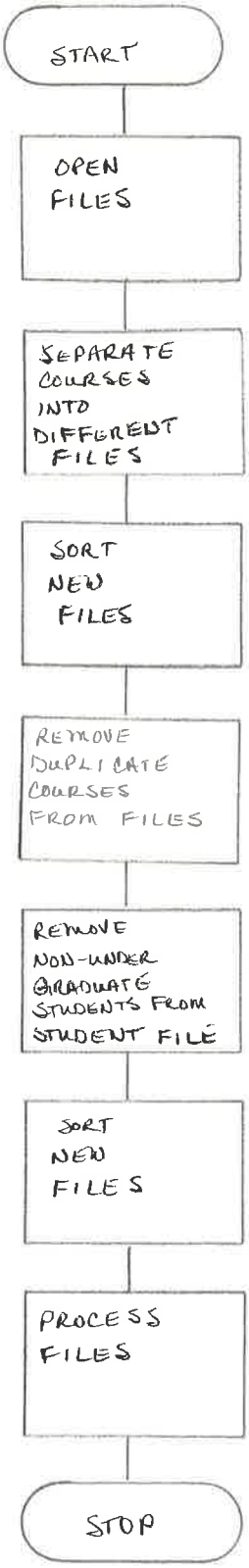
This appendix contains an example of the records for HST 210 and AIS 210 as they appear in the file which holds the basic studies courses.

AIS	210	B3	B5C			
HST	210	B3	B5C			

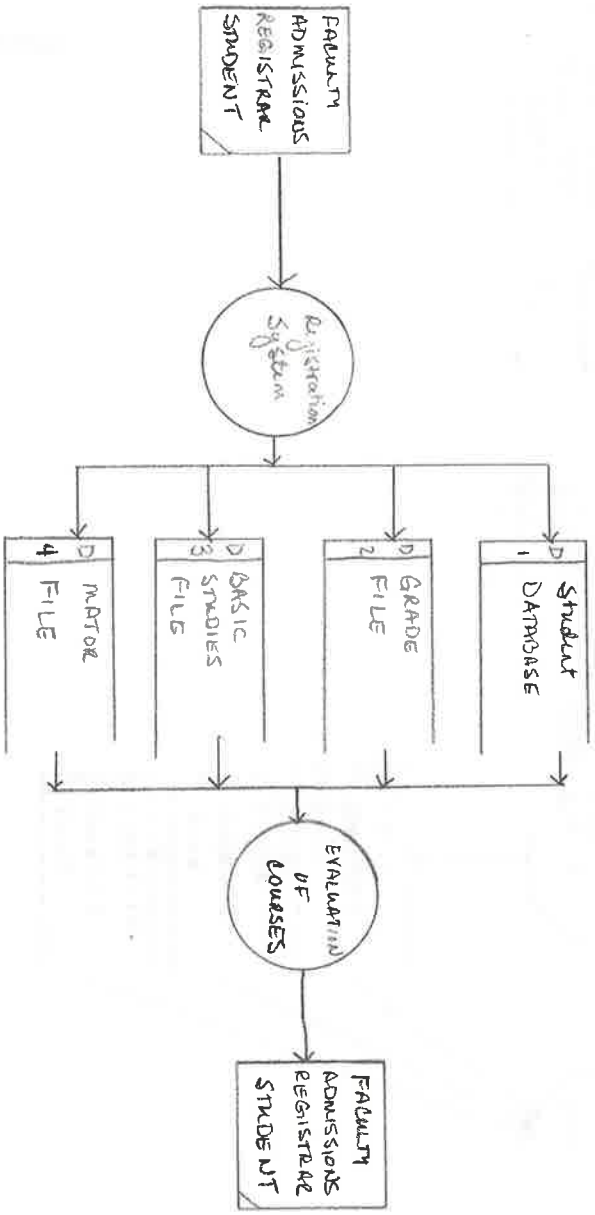
APPENDIX F

This appendix contains the data flow diagrams, system flowcharts, and the HIPO charts that were created during the design phase of the project.

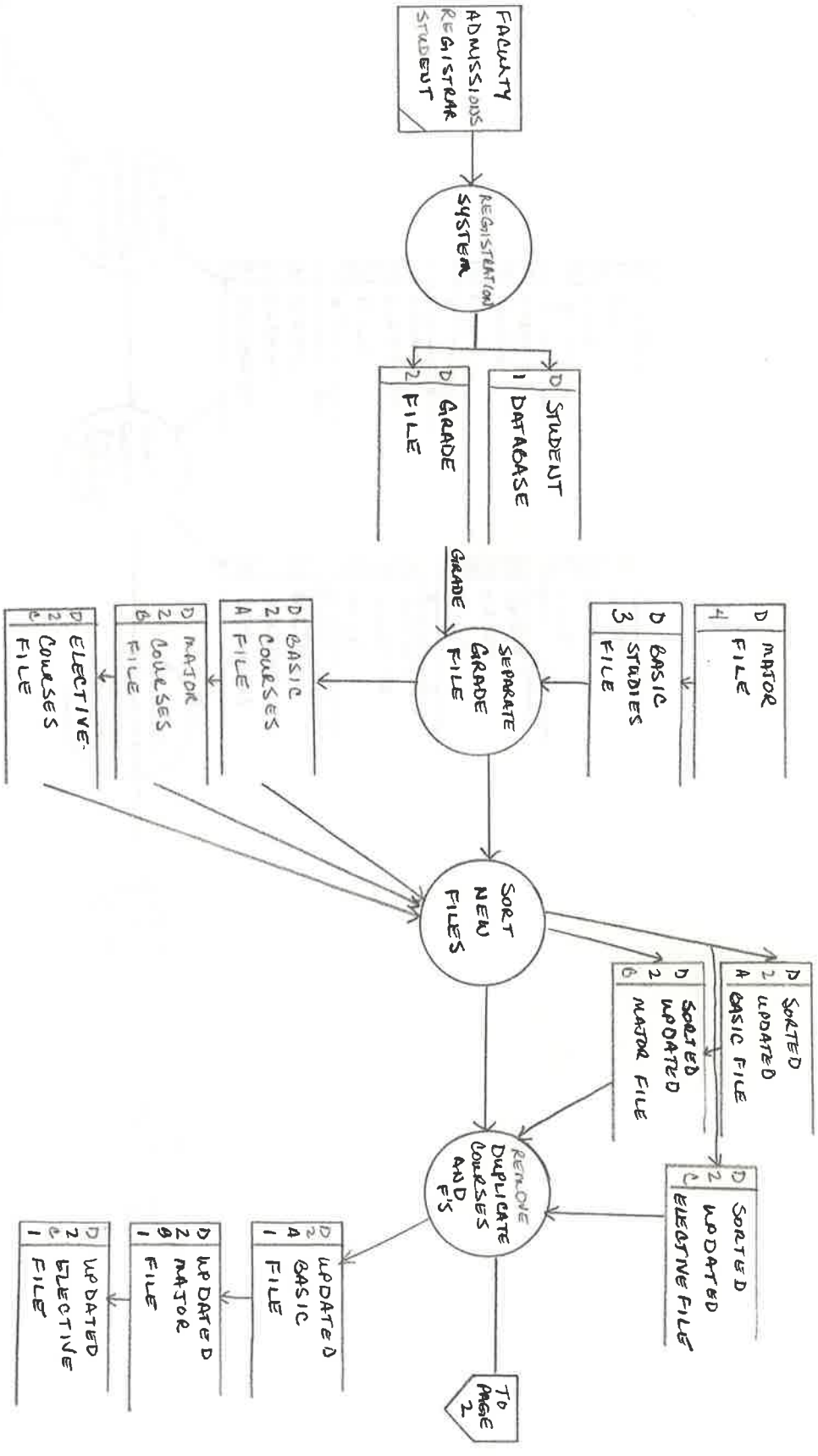
SYSTEM FLOWCHART



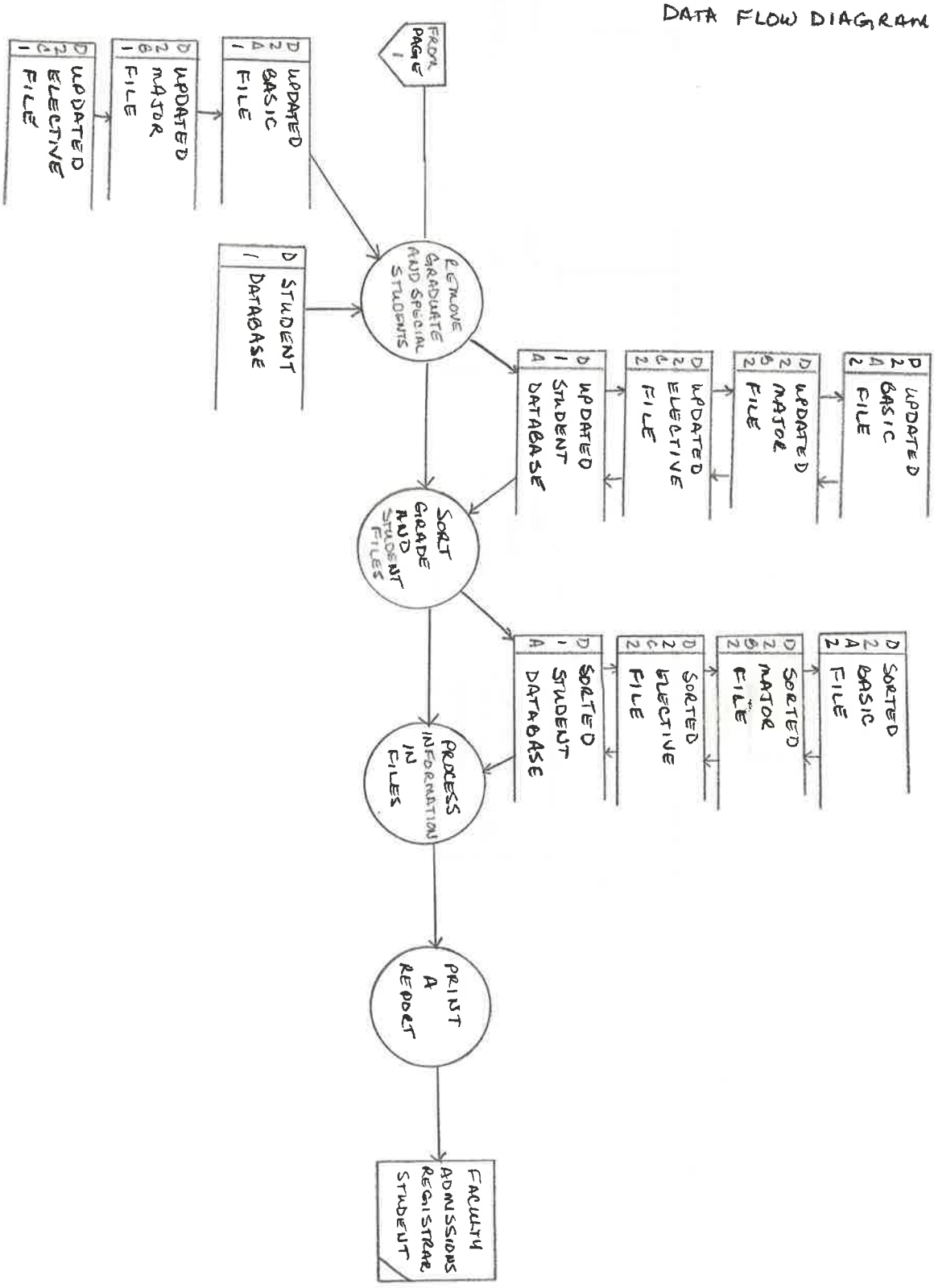
DATA FLOW DIAGRAM



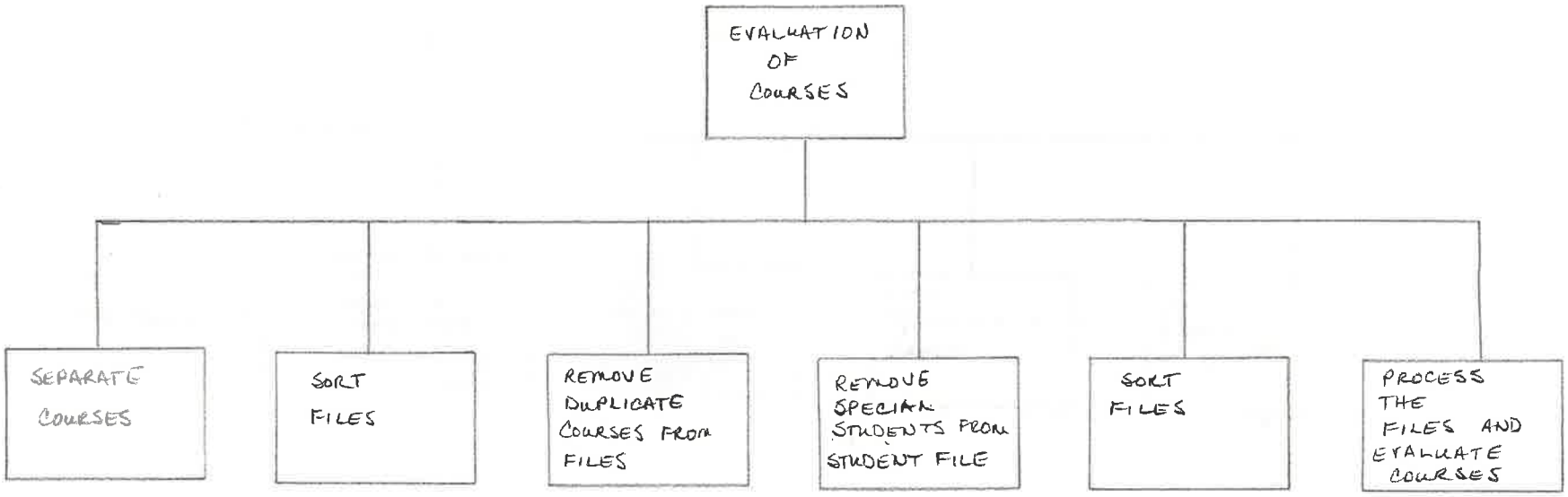
HIGH LEVEL
DATA FLOW DIAGRAM



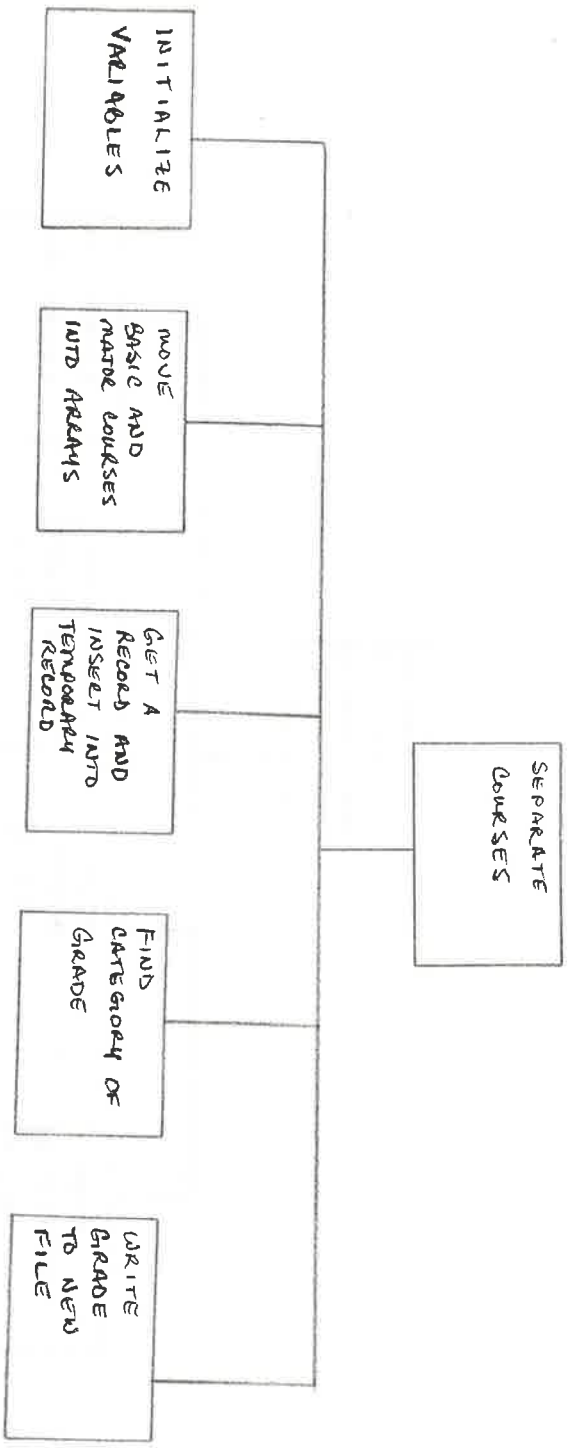
HIGH LEVEL
DATA FLOW DIAGRAM



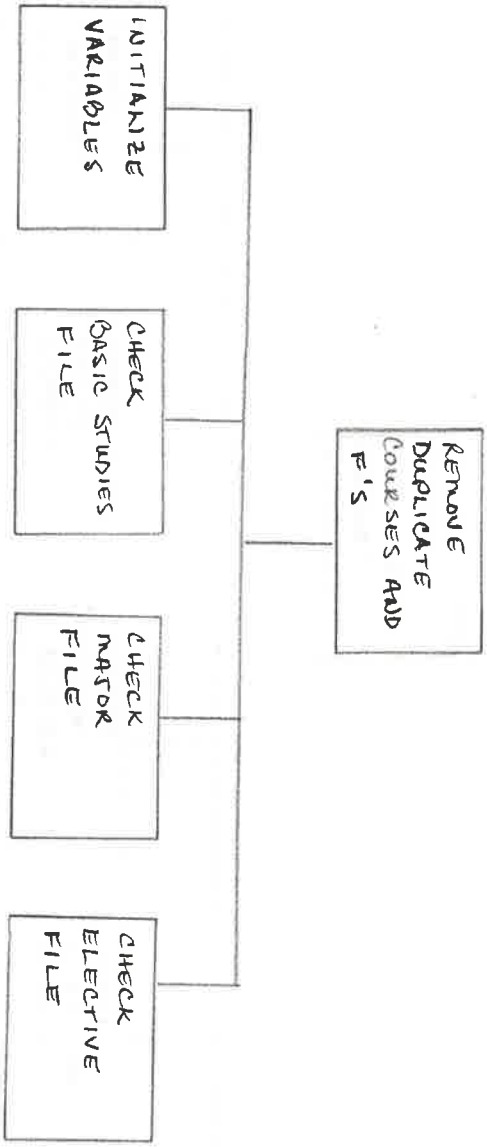
HIPO CHART


EVALUATION OF COURSES
SEPARATE COURSES
SORT FILES
REMOVE DUPLICATE COURSES FROM FILES
REMOVE SPECIAN STUDENTS FROM STUDENT FILE
SORT FILES
PROCESS THE FILES AND EVALUATE COURSES

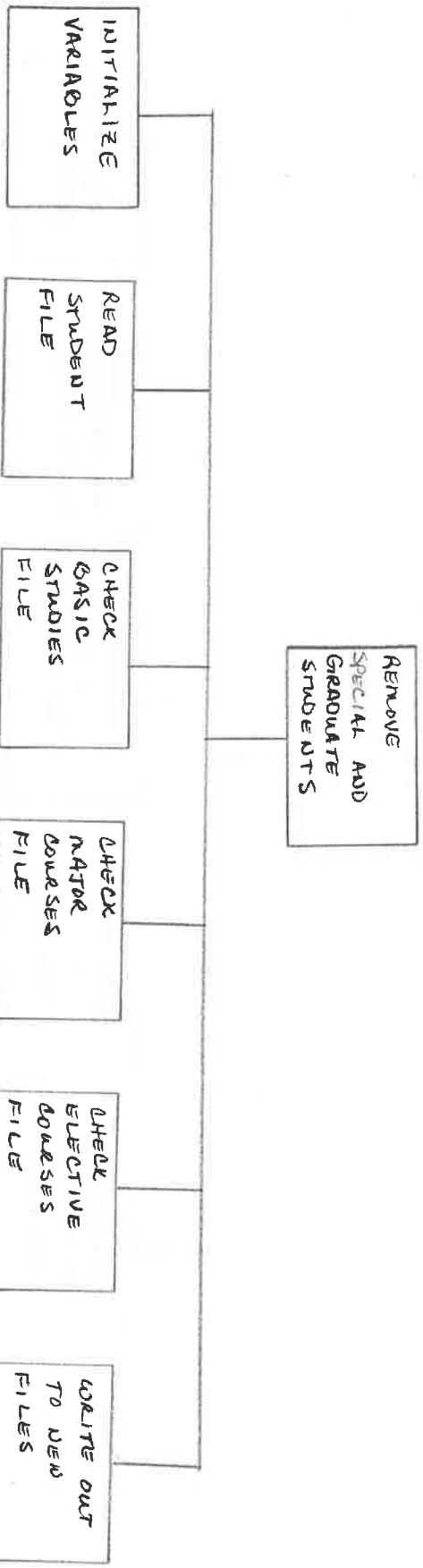
HIPO CHART FOR PROGRAM NONBAS



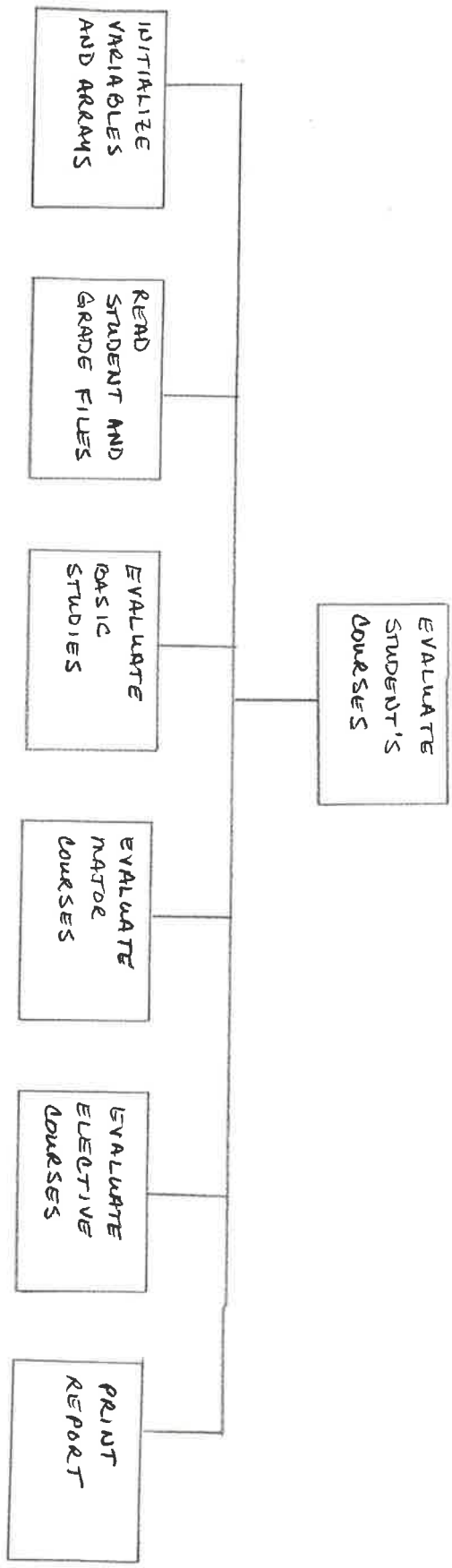
HIPO CHART FOR
PROGRAM REMDUPE



HIPO CHART FOR
PROGRAM READGR



HIPO CHART FOR
PROGRAM PROCESS



APPENDIX G

This appendix contains the program that was written to test the results of my research.

BIBLIOGRAPHY

Borland International. Turbo Pascal 4.0 . Scotts Valley, California, 1987.

Davis, William S. Systems Analysis and Design. Reading, Massachusetts: Addison-Wesley Publishing Co., 1983.

Duke University Computation Center. PASCALW Document No. LS-350-2: 1984.

Jamsa, Kris. DOS The Complete Reference. Berkley, California: McGraw-Hill, 1987.

Kruse, Robert L. Data Structures and Program Design. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1987.

Laurie, Peter. Databases. New York: Chapman and Hall/Meuthen, 1985.

North Carolina Educational Computing Service. Job Control Language User's Guide. Research Triangle Park, North Carolina, 1984.

Reges, Stuart. Building Pascal Programs. Boston: Little, Brown, and Co., 1987.

Rosen, Kenneth H. Elementary Number Theory and Its Applications. New York: Addison-Weslet Publishing Co., 1988.

Triangle Universities Computation Center. TUCC WYLBUR User Guide. Triangle Park, North Carolina, 1982.

Warren, Gene and Dr. Monika Brown. Pembroke State University 1988-1989 Catalog.

```
PROGRAM NONBAS (GRADES,REQUIRED,CRSFILE,REQ_MAJ,EXTRA_FILE,
                MAJOR_FILE);
(*$ S 40000,X 0 *)
CONST
   MAXLIST = 161;
   MAXMAJOR = 165;

TYPE
   COURSEREC = PACKED ARRAY[1..6] OF CHAR;
   GRADEREC = PACKED ARRAY[1..22] OF CHAR;
   GRADEARRAY = ARRAY[1..MAXLIST] OF GRADEREC;
   STRING1 = PACKED ARRAY[1..1] OF CHAR;
   STRING2 = PACKED ARRAY[1..2] OF CHAR;
   STRING3 = PACKED ARRAY [1..3] OF CHAR;
   STRING6 = PACKED ARRAY[1..6] OF CHAR;
   STRING8 = PACKED ARRAY [1..8] OF CHAR;
   STRING9 = PACKED ARRAY [1..9] OF CHAR;
   STRING13 = PACKED ARRAY [1..13] OF CHAR;
   STRING22 = PACKED ARRAY [1..22] OF CHAR;
   STRING34 = PACKED ARRAY [1..34] OF CHAR;
   GRADE_REC = RECORD
                  SSAN : STRING9;
                  GRADE : STRING2;
                  NAME : STRING34;
                  DEPT : STRING3;
                  COURSE : STRING3;
                  HOUR : STRING2;
                  SEMESTER : STRING3;
               END;
   MAJOR_ARRAY = ARRAY[1..MAXMAJOR] OF COURSEREC;

VAR
   GRADES : FILE OF CHAR;
   REQUIRED : FILE OF CHAR;
   CRSFILE : FILE OF CHAR;
   REQ_MAJ : FILE OF CHAR;
   EXTRA_FILE : FILE OF CHAR;
   MAJOR_FILE : FILE OF CHAR;
   MAJOR : MAJOR_ARRAY;
   MAJ_SIZE : INTEGER;
   BSTUDY : GRADEARRAY;
   CODE : INTEGER;
   ARRAY_SIZE : INTEGER;
   T_GRADE : GRADE_REC;


PROCEDURE GET_NEXT_REC (VAR T_GRADE : GRADE_REC);
VAR
   BLANK :STRING1;
   SSAN : STRING9;
   GRADE : STRING2;
   NAME : STRING34;
   FILLER8 : STRING8;
   DEPT : STRING3;
   COURSE : STRING3;
   FILLER2 : STRING2;
   HOUR : STRING2;
   FILLER13 : STRING13;
   SEMESTER : STRING3;

BEGIN
   READ(GRADES,BLANK,SSAN,GRADE,NAME,FILLER8,DEPT,COURSE);
   READLN(GRADES,FILLER2,HOUR,FILLER13,SEMESTER);
   T_GRADE.SSAN:= SSAN;
   T_GRADE.GRADE:= GRADE;
   T_GRADE.NAME:= NAME;
   T_GRADE.DEPT:= DEPT;
   T_GRADE.COURSE:= COURSE;
   T_GRADE.HOUR:= HOUR;
   T_GRADE.SEMESTER:= SEMESTER;
END;

PROCEDURE INIT_ARRAYS (VAR BSTUDY : GRADEARRAY;
                       VAR ARRAY_SIZE : INTEGER;
                       VAR MAJOR : MAJOR_ARRAY;
                       VAR MAJ_SIZE : INTEGER);

VAR
   TEMP_STUDY : STRING22;
   COUNT : INTEGER;
```

```
  READLN(GRADES,FILLER2,HOUR,FILLER13,SEMESTER);
  T_GRADE.SSAN:= SSAN;
  T_GRADE.GRADE:= GRADE;
  T_GRADE.NAME:= NAME;
  T_GRADE.DEPT:= DEPT;
  T_GRADE.COURSE:= COURSE;
  T_GRADE.HOUR:= HOUR;
  T_GRADE.SEMESTER:= SEMESTER;
END;

PROCEDURE INIT_ARRAYS (VAR BSTUDY : GRADEARRAY;
                       VAR ARRAY_SIZE : INTEGER;
                       VAR MAJOR : MAJOR_ARRAY;
                       VAR MAJ_SIZE : INTEGER);

VAR
   TEMP_STUDY : STRING22;
   COUNT : INTEGER;
   TEMP_MAJOR : STRING6;
   DEPT : STRING3;
   COURSE : STRING3;
   RPT : STRING1;
   LOC1 : STRING3;
   LOC2 : STRING3;
   LOC3 : STRING3;
   LOC4 : STRING3;
   LOC5 : STRING3;

BEGIN
  FOR COUNT:= 1 TO MAXLIST DO
       BSTUDY[COUNT]:= '                      ';
  FOR COUNT:= 1 TO MAXMAJOR DO
       MAJOR[COUNT]:= '      ';
  COUNT:= 1;
  TEMP_STUDY:= '                      ';
  WHILE NOT EOF(REQUIRED) DO
    BEGIN
       READ(REQUIRED,DEPT,COURSE,RPT,LOC1,LOC2,LOC3);
       READLN(REQUIRED,LOC4,LOC5);
       TEMP_STUDY:= DEPT;
       STRCONCAT(TEMP_STUDY,COURSE);
       STRCONCAT(TEMP_STUDY,RPT);
       STRCONCAT(TEMP_STUDY,LOC1);
       STRCONCAT(TEMP_STUDY,LOC2);
       STRCONCAT(TEMP_STUDY,LOC3);
       STRCONCAT(TEMP_STUDY,LOC4);
       STRCONCAT(TEMP_STUDY,LOC5);
       BSTUDY[COUNT]:= TEMP_STUDY;
       TEMP_STUDY:= '                      ';
       COUNT:= COUNT + 1;
    END;
  ARRAY_SIZE:= COUNT;
  COUNT:= 1;
  WHILE NOT EOF(REQ_MAJ) DO
    BEGIN
      READ (REQ_MAJ,DEPT,COURSE);
      TEMP_MAJOR:= DEPT;
      STRCONCAT(TEMP_MAJOR,COURSE);
      MAJOR[COUNT]:= TEMP_MAJOR;
      TEMP_MAJOR:= '      ';
      COUNT:= COUNT + 1;
    END;
  MAJ_SIZE:= COUNT;
END;

PROCEDURE FIND_COURSE (VAR LIST : GRADEARRAY;
                       TARGET : COURSEREC;
                       ARRAY_SIZE : INTEGER;
                       VAR GOT_IT : BOOLEAN;
                       VAR LOCATION : INTEGER);

VAR
  TOP,BOTTOM,MID : INTEGER;
  CHECK : COURSEREC;

BEGIN
  TOP:= ARRAY_SIZE;
  BOTTOM:= 1;
  WHILE TOP > BOTTOM DO
    BEGIN
      MID:= (TOP + BOTTOM) DIV 2;
      SUBSTR(LIST[MID],6,1,CHECK);
      IF TARGET > CHECK THEN
         BOTTOM:= MID + 1
```

```
                        TARGET : COURSEREC;
                        ARRAY_SIZE : INTEGER;
                        VAR GOT_IT : BOOLEAN;
                        VAR LOCATION : INTEGER);

VAR
  TOP,BOTTOM,MID : INTEGER;
  CHECK : COURSEREC;

BEGIN
  TOP:= ARRAY_SIZE;
  BOTTOM:= 1;
  WHILE TOP > BOTTOM DO
    BEGIN
      MID:= (TOP + BOTTOM) DIV 2;
      SUBSTR(LIST[MID],6,1,CHECK);
      IF TARGET > CHECK THEN
         BOTTOM:= MID + 1
      ELSE
         TOP:= MID;
    END;
  IF TOP = 0 THEN
     GOT_IT:= FALSE
  ELSE
    BEGIN
      SUBSTR(LIST[TOP],6,1,CHECK);
      GOT_IT:= (TARGET = CHECK);
    END;
  LOCATION:= TOP;
END;

PROCEDURE CHECK_MAJOR (VAR MAJOR : MAJOR_ARRAY;
                       SUBJECT : COURSEREC;
                       MAJ_SIZE : INTEGER;
                       VAR GRADE_FOUND : BOOLEAN;
                       VAR LOCATION : INTEGER);

VAR
  TOP,BOTTOM,MID : INTEGER;
  CHECK : COURSEREC;

BEGIN
  TOP:= MAJ_SIZE;
  BOTTOM:= 1;
  WHILE TOP > BOTTOM DO
    BEGIN
      MID:= (TOP + BOTTOM) DIV 2;
      CHECK:= MAJOR[MID];
      IF SUBJECT > CHECK THEN
        BOTTOM:= MID + 1
      ELSE
        TOP:= MID;
    END;
  IF TOP = 0 THEN
     GRADE_FOUND:= FALSE
  ELSE
     BEGIN
       SUBSTR(MAJOR[TOP],6,1,CHECK);
       GRADE_FOUND:= (SUBJECT = CHECK);
     END;
  LOCATION:= TOP;
END;

PROCEDURE CHECK_GRADE ( VAR CRSFILE : FILE OF CHAR;
                        VAR T_GRADE : GRADE_REC;
                        ARRAY_SIZE : INTEGER;
                        MAJ_SIZE : INTEGER;
                        VAR MAJOR_FILE : FILE OF CHAR;
                        VAR EXTRA_FILE : FILE OF CHAR);

VAR
  GRADE_FOUND : BOOLEAN;
  LOCATION : INTEGER
  TEMP_RPT : INTEGER;
  TEMP_LOC : STRING3;
  COUNT : INTEGER;


PROCEDURE WRITE_BASIC (COUNT : INTEGER;
                       T_GRADE : GRADE_REC;
                       VAR CRSFILE : FILE OF CHAR);

VAR
  TEMP_LOC : STRING3;
```

```
                        VAR MAJOR_FILE : FILE OF CHAR;
                        VAR EXTRA_FILE : FILE OF CHAR);

VAR
  GRADE_FOUND : BOOLEAN;
  LOCATION : INTEGER
  TEMP_RPT : INTEGER;
  TEMP_LOC : STRING3;
  COUNT : INTEGER;


PROCEDURE WRITE_BASIC (COUNT : INTEGER;
                       T_GRADE : GRADE_REC;
                       VAR CRSFILE : FILE OF CHAR);

VAR
  TEMP_LOC : STRING3;
  TEMP_RPT : INTEGER;
  RPT : STRING1;

BEGIN
  SUBSTR(BSTUDY[COUNT],3,8,TEMP_LOC);
  SUBSTR(BSTUDY[COUNT],1,7,RPT);
  WRITE(CRSFILE,T_GRADE.SSAN,T_GRADE,NAME);
  WRITE(CRSFILE,T_GRADE.DEPT,T_GRADE.COURSE);
  WRITE(CRSFILE,T_GRADE.HOUR,T_GRADE.GRADE);
  WRITE(CRSFILE,T_GRADE.SEMESTER,RPT,TEMP_LOC);
  TEMP_RPT:= TRUNC(STOR(RPT));
  IF TEMP_RPT > 1 THEN
    BEGIN
      SUBSTR(BSTUDY[COUNT],3,11,TEMP_LOC);
      WRITE(CRSFILE,TEMP_LOC);
      IF TEMP_RPT > 2 THEN
        BEGIN
          SUBSTR(BSTUDY[COUNT]3,14,TEMP_LOC);
          WRITE(CRSFILE,TEMP_LOC);
          IF TEMP_RPT > 3 THEN
            BEGIN
              SUBSTR(BSTUDY[COUNT],3,17,TEMP_LOC);
              WRITE(CRSFILE,TEMP_LOC);
              IF TEMP_RPT > 4 THEN
                BEGIN
                  SUBSTR(BSTUDY[COUNT],3,20,TEMP_LOC);
                  WRITELN(CRSFILE,TEMP_LOC);
                END
              ELSE
                WRITELN(CRSFILE);
            END
          ELSE
            WRITELN(CRSFILE);
        END
      ELSE
        WRITELN(CRSFILE);
    END
  ELSE
    WRITELN(CRSFILE);
END;

PROCEDURE WRITE_MAJOR (COUNT : INTEGER;
                       VAR MAJOR_FILE : FILE OF CHAR;
                       T_GRADE : COURSEREC);

BEGIN
  WRITE(MAJOR_FILE,T_GRADE.SSAN,T_GRADE.NAME);
  WRITE(MAJOR_FILE,T_GRADE.DEPT,T_GRADE.COURSE);
  WRITE(MAJOR_FILE,T_GRADE.HOUR,T_GRADE.GRADE);
  WRITELN(MAJOR_FILE,T_GRADE.SEMESTER);
END;

PROCEDURE WRITE_EXTRA (COUNT : INTEGER;
                       VAR EXTRA_FILE : FILE OF CHAR;
                       T_GRADE : COURSEREC);

BEGIN
  WRITE(EXTRA_FILE,T_GRADE.SSAN,T_GRADE.NAME);
  WRITE(EXTRA_FILE,T_GRADE.DEPT,T_GRADE.COURSE);
  WRITE(EXTRA_FILE,T_GRADE.HOUR,T_GRADE.GRADE);
  WRITELN(EXTRA_FILE,T_GRADE.SEMESTER);
END;


BEGIN
```

```
  WRITE(MAJOR_FILE,T_GRADE.HOUR,T_GRADE.GRADE);
  WRITELN(MAJOR_FILE,T_GRADE.SEMESTER);
END;

PROCEDURE WRITE_EXTRA (COUNT : INTEGER;
                       VAR EXTRA_FILE : FILE OF CHAR;
                       T_GRADE : COURSEREC);

BEGIN
  WRITE(EXTRA_FILE,T_GRADE.SSAN,T_GRADE.NAME);
  WRITE(EXTRA_FILE,T_GRADE.DEPT,T_GRADE.COURSE);
  WRITE(EXTRA_FILE,T_GRADE.HOUR,T_GRADE.GRADE);
  WRITELN(EXTRA_FILE,T_GRADE.SEMESTER);
END;



BEGIN
  GRADE_FOUND:= FALSE;
  WHILE NOT EOF(GRADES) DO
    BEGIN
      SUBJECT:= T_GRADE.DEPT;
      STRCONCAT(SUBJECT,T_GRADE.COURSE);
      FIND_COURSE(BSTUDY,SUBJECT,ARRAY_SIZE,GRADE_FOUND,LOCATION);
      IF GRADE_FOUND THEN
         BEGIN
           COUNT:= LOCATION;
           WRITE_BASIC(COUNT,T_GRADE,CRSFILE);
         END
      ELSE
       BEGIN
         CHECK_MAJOR(MAJOR,SUBJECT,MAJ_SIZE,GRADE_FOUND,LOCATION);
         IF GRADE_FOUND THEN
           BEGIN
             COUNT:= LOCATION;
             WRITE_MAJOR(COUNT,MAJOR_FILE,T_GRADE);
           END
         ELSE
           WRITE_EXTRA(EXTRA_FILE,LOCATION,T_GRADE);
       END;
      GET_NEXT_REC(T_GRADE);
    END;
END;

BEGIN
  RESET(GRADES);
  RESET(REQUIRED);
  RESET(REQ_MAJOR);
  REWRITE(CRSFILE);
  REWRITE(MAJOR_FILE);
  REWRITE(EXTRA_FILE);
  INIT_ARRAYS(BSTUDY,ARRAY_SIZE,MAJOR,MAJ_SIZE);
  GET_NEXT_REC(T_GRADE);
  CHECK_GRADE(CRSFILE,T_GRADE,ARRAY_SIZE,MAJOR,MAJ_SIZE,
              MAJOR_FILE,EXTRA_FILE);
END.


PROGRAM REMDUPE(CLSFILE,CRSFILE,MAJ_FILE,
       MAJOR_FILE,EXTRA_FILE,EX_FILE);

TYPE
  STRING1 = PACKED ARRAY[1..1] OF CHAR;
  STRING2 = PACKED ARRAY[1..2] OF CHAR;
  STRING3 = PACKED ARRAY[1..3] OF CHAR;
  STRING9 = PACKED ARRAY[1..9] OF CHAR;
  STRING34 = PACKED ARRAY[1..34] OF CHAR;
  STRING72 = PACKED ARRAY[1..72] OF CHAR;
  CLASSREC = RECORD
               SSAN : STRING9;
               NAME : STRING34;
               DEPT : STRING3;
               COURSE : STRING3;
               HOUR : STRING2;
               GRADE : STRING2;
               SEMESTER : STRING3;
               RPT : STRING1;
               LOC1 : STRING3;
               LOC2 : STRING3;
               LOC3 : STRING3;
               LOC4 : STRING3;
               LOC5 : STRING3;
             END;
```

```
  STRING72 = PACKED ARRAY[1..72] OF CHAR;
  CLASSREC = RECORD
               SSAN : STRING9;
               NAME : STRING34;
               DEPT : STRING3;
               COURSE : STRING3;
               HOUR : STRING2;
               GRADE : STRING2;
               SEMESTER : STRING3;
               RPT : STRING1;
               LOC1 : STRING3;
               LOC2 : STRING3;
               LOC3 : STRING3;
               LOC4 : STRING3;
               LOC5 : STRING3;
             END;

VAR
  CLSFILE : FILE OF CHAR;
  CRSFILE : FILE OF CHAR;
  MAJ_FILE : FILE OF CHAR;
  MAJOR_FILE : FILE OF CHAR;
  EX_FILE : FILE OF CHAR;
  EXTRA_FILE : FILE OF CHAR;


PROCEDURE CHECK_BASIC (VAR CLSFILE : FILE OF CHAR;
                       VAR CRSFILE : FILE OF CHAR);

VAR
  SSAN : STRING9;
  NAME : STRING34;
  DEPT : STRING3;
  COURSE : STRING3;
  HOUR : STRING2;
  GRADE : STRING2;
  SEMESTER : STRING3;
  RPT : STRING1;
  LOC1 : STRING3;
  LOC2 : STRING3;
  LOC3 : STRING3;
  LOC4 : STRING3;
  LOC5 : STRING3;
  TEMP_COURSE : STRING3;
  TEMP_HOUR : STRING2;
  TEMP_SSN : STRING9;
  TEMP_DEPT : STRING3;
  FIRST : BOOLEAN;
  COUNT : INTEGER;

BEGIN
  READ (CLSFILE,SSAN,NAME,DEPT,COURSE,HOUR,GRADE);
  READLN(CLSFILE,SEMESTER,RPT,LOC1,LOC2,LOC3,LOC4,LOC5);
  TEMP_COURSE:= COURSE;
  TEMP_HOUR:= HOUR;
  TEMP_SSN:= SSAN;
  TEMP_DEPT:= DEPT;
  FIRST:= TRUE;
  COUNT:= 1;
  WHILE NOT EOF(CLSFILE) DO
    BEGIN
      SUBSTR(GRADE,1,1,TEMP_GRADE);
      IF (TEMP_COURSE = COURSE) AND (TEMP_DEPT = DEPT) AND
         (TEMP_HOUR = HOUR) AND TEMP_SSN =SSAN) AND NOT(FIRST) THEN
        BEGIN
          READ(CLSFILE,SSAN,NAME,DEPT,COURSE,HOUR,GRADE);
          READLN (CLSFILE,SEMESTER,RPT,LOC1,LOC2,LOC3,LOC4,LOC5);
        END
      ELSE
        BEGIN
          IF (TEMP_GRADE = 'A') OR (TERMP_GRADE = 'B') OR
             (TEMP_GRADE = 'C') OR TEMP_GRADE = 'D') THEN
              BEGIN
                WRITE(CRSFILE,SSAN,NAME,DEPT,COURSE,HOUR,GRADE);
                WRITELN(CRSFILE,SEMESTER,RPT,LOC1,LOC2,LOC3,LOC4,LOC5);
                TEMP_COURSE:= COURSE;
                TEMP_DEPT:= DEPT;
                TEMP_HOUR:= HOUR;
                TEMP_SSN:= SSAN;
              END;
          COUNT:= COUNT + 1;
          IF COUNT > 1 THEN
            FIRST:= FALSE;
```

```
          READ(CLSFILE,SSAN,NAME,DEPT,COURSE,HOUR,GRADE);
          READLN (CLSFILE,SEMESTER,RPT,LOC1,LOC2,LOC3,LOC4,LOC5);
        END
      ELSE
        BEGIN
          IF (TEMP_GRADE = 'A') OR (TERMP_GRADE = 'B') OR
             (TEMP_GRADE = 'C') OR TEMP_GRADE = 'D') THEN
              BEGIN
                WRITE(CRSFILE,SSAN,NAME,DEPT,COURSE,HOUR,GRADE);
                WRITELN(CRSFILE,SEMESTER,RPT,LOC1,LOC2,LOC3,LOC4,LOC5);
                TEMP_COURSE:= COURSE;
                TEMP_DEPT:= DEPT;
                TEMP_HOUR:= HOUR;
                TEMP_SSN:= SSAN;
              END;
          COUNT:= COUNT + 1;
          IF COUNT > 1 THEN
            FIRST:= FALSE;
          READ(CLSFILE,SSAN,NAME,DEPT,COURSE,HOUR,GRADE);
          READLN(CLSFILE,SEMESTER,RPT,LOC1,LOC2,LOC3,LOC4,LOC5);
        END;
    END;
END;

PROCEDURE CHECK_MAJOR (VAR MAJ_FILE : FILE OF CHAR;
                       VAR MAJOR_FILE : FILE OF CHAR);
VAR
  SSAN : STRING9;
  NAME : STRING34;
  DEPT : STRING3;
  COURSE : STRING3;
  SEMESTER : STRING3;
  HOUR : STRING2;
  GRADE : STRING2;
  TEMP_GRADE : STRING2;
  TEMP_DEPT : STRING3;
  TEMP_COURSE : STRING3;
  TEMP_SSN : STRING9;
  TEMP_HOUR : STRING2;
  COUNT : INTEGER;
  FIRST : BOOLEAN;

BEGIN
  READ (MAJ_FILE,SSAN,NAME,DEPT,COURSE,HOUR);
  READLN(MAJ_FILE,GRADE,SEMESTER);
  TEMP_COURSE:= COURSE;
  TEMP_HOUR:= HOUR;
  TEMP_SSN:= SSAN;
  TEMP_DEPT:= DEPT;
  FIRST:= TRUE;
  COUNT:= 1;
  WHILE NOT EOF(MAJ_FILE) DO
    BEGIN
      SUBSTR(GRADE,1,1,TEMP_GRADE);
      IF (TEMP_COURSE = COURSE) AND (TEMP_DEPT = DEPT) AND
         (TEMP_HOUR = HOUR) AND TEMP_SSN = SSAN) AND NOT(FIRST) THEN
          BEGIN
            READ(MAJ_FILE,SSAN,NAME,DEPT,COURSE,HOUR,GRADE);
            READLN(MAJ_FILE,SEMESTER);
          END
      ELSE
        BEGIN
          IF (TEMP_GRADE = 'A') OR (TEMP_GRADE = 'B') OR
             (TEMP_GRADE = 'C') OR (TEMP_GRADE = 'D') THEN
               BEGIN
                 WRITE(MAJOR_FILE,SSAN,NAME,DEPT,COURSE,HOUR);
                 WRITELN(MAJOR_FILE,GRADE,SEMESTER);
                 TEMP_COURSE:= COURSE;
                 TEMP_DEPT:= DEPT;
                 TEMP_HOUR:= HOUR;
                 TEMP_SSN:= SSAN;
               END;
          COUNT:= COUNT + 1;
          IF COUNT > 1 THEN
            FIRST:= FALSE;
          READ(MAJ_FILE,SSAN,NAME,DEPT,COURSE,HOUR);
          READLN(MAJ_FILE,GRADE,SEMESTER);
        END;
    END;
END;

PROCEDURE CHECK_EXTRA (VAR EXTRA_FILE : FILE OF CHAR;
                       VAR EX_FILE    : FILE OF CHAR);
VAR
```

```
                TEMP_COURSE:= COURSE;
                TEMP_DEPT:= DEPT;
                TEMP_HOUR:= HOUR;
                TEMP_SSN:= SSAN;
              END;
          COUNT:= COUNT + 1;
          IF COUNT > 1 THEN
            FIRST:= FALSE;
          READ(MAJ_FILE,SSAN,NAME,DEPT,COURSE,HOUR);
          READLN(MAJ_FILE,GRADE,SEMESTER);
        END;
    END;
END;

PROCEDURE CHECK_EXTRA (VAR EXTRA_FILE : FILE OF CHAR;
                       VAR EX_FILE    : FILE OF CHAR);
VAR
  SSAN : STRING9;
  NAME : STRING34;
  DEPT : STRING3;
  COURSE : STRING3;
  SEMESTER : STRING3;
  HOUR : STRING2;
  GRADE : STRING2;
  TEMP_GRADE : STRING2;
  TEMP_DEPT : STRING3;
  TEMP_COURSE : STRING3;
  TEMP_SSN : STRING9;
  TEMP_HOUR : STRING2;
  COUNT : INTEGER;
  FIRST : BOOLEAN;

BEGIN
  READ (EXTRA_FILE,SSAN,NAME,DEPT,COURSE,HOUR);
  READLN(EXTRA_FILE,GRADE,SEMESTER);
  TEMP_COURSE:= COURSE;
  TEMP_HOUR:= HOUR;
  TEMP_SSN:= SSAN;
  TEMP_DEPT:= DEPT;
  FIRST:= TRUE;
  COUNT:= 1;
  WHILE NOT EOF(EXTRA_FILE) DO
    BEGIN
      SUBSTR(GRADE,1,1,TEMP_GRADE);
      IF (TEMP_COURSE = COURSE) AND (TEMP_DEPT = DEPT) AND
         (TEMP_HOUR = HOUR) AND TEMP_SSN = SSAN) AND NOT(FIRST) THEN
          BEGIN
            READ(EXTRA_FILE,SSAN,NAME,DEPT,COURSE,HOUR,GRADE);
            READLN(EXTRA_FILE,SEMESTER);
          END
      ELSE
        BEGIN
          IF (TEMP_GRADE = 'A') OR (TEMP_GRADE = 'B') OR
             (TEMP_GRADE = 'C') OR (TEMP_GRADE = 'D') THEN
               BEGIN
                 WRITE(EX_FILE,SSAN,NAME,DEPT,COURSE,HOUR);
                 WRITELN(EX_FILE,GRADE,SEMESTER);
                 TEMP_COURSE:= COURSE;
                 TEMP_DEPT:= DEPT;
                 TEMP_HOUR:= HOUR;
                 TEMP_SSN:= SSAN;
               END;
          COUNT:= COUNT + 1;
          IF COUNT > 1 THEN
            FIRST:= FALSE;
          READ(EXTRA_FILE,SSAN,NAME,DEPT,COURSE,HOUR);
          READLN(EXTRA_FILE,GRADE,SEMESTER);
        END;
    END;
END;



BEGIN
  RESET(CLSFILE);
  RESET(MAJ_FILE);
  RESET(EXTRA_FILE);
  REWRITE(CRSFILE);
  REWRITE(MAJOR_FILE);
  REWRITE(EX_FILE);
  CHECK_BASIC(CLSFILE,CRSFILE);
  CHECK_MAJOR(MAJ_FILE,MAJOR_FILE);
  CHECK_EXTRA(EXTRA_FILE,EX_FILE);
```

```
          READ(EXTRA_FILE,SSAN,NAME,DEPT,COURSE,HOUR);
          READLN(EXTRA_FILE,GRADE,SEMESTER);
        END;
    END;
END;



BEGIN
  RESET(CLSFILE);
  RESET(MAJ_FILE);
  RESET(EXTRA_FILE);
  REWRITE(CRSFILE);
  REWRITE(MAJOR_FILE);
  REWRITE(EX_FILE);
  CHECK_BASIC(CLSFILE,CRSFILE);
  CHECK_MAJOR(MAJ_FILE,MAJOR_FILE);
  CHECK_EXTRA(EXTRA_FILE,EX_FILE);
END.


PROGRAM READGR (CRSFILE,DUMMY,CLSFILE,NEWFILE,
                MAJOR_FILE,MAJ_FILE,EX_FILE,EXTRA_FILE);

TYPE
  STRING1 = PACKED ARRAY[1..1] OF CHAR;
  STRING2 = PACKED ARRAY[1..2] OF CHAR;
  STRING3 = PACKED ARRAY[1..3] OF CHAR;
  STRING4 = PACKED ARRAY[1..4] OF CHAR;
  STRING5 = PACKED ARRAY [1..5] OF CHAR;
  STRING6 = PACKED ARRAY [1..6] OF CHAR;
  STRING7 = PACKED ARRAY [1..7] OF CHAR;
  STRING8 = PACKED ARRAY [1..8] OF CHAR;
  STRING9 = PACKED ARRAY [1..9] OF CHAR;
  STRING10 = PACKED ARRAY [1..10] OF CHAR;
  STRING11 = PACKED ARRAY[1..11] OF CHAR;
  STRING13 = PACKED ARRAY[1..13] OF CHAR;
  STRING14 = PACKED ARRAY[1..14] OF CHAR;
  STRING20 = PACKED ARRAY[1..20] OF CHAR;
  STRING21 = PACKED ARRAY [1..21] OF CHAR;
  STRING25 = PACKED ARRAY[1..25] OF CHAR;
  STRING28 = PACKED ARRAY[1..28] OF CHAR;
  STRING34 = PACKED ARRAY [1..34] OF CHAR;
  STRING39 = PACKED ARRAY [1..39] OF CHAR;
  STRING95 = PACKED ARRAY[1..95] OF CHAR;

VAR
  DUMMY : FILE OF CHAR;
  NEWFILE : FILE OF CHAR;
  CRSFILE : FILE OF CHAR;
  CLSFILE : FILE OF CHAR;
  SSAN,SSN : STRING9;
  NAME,NOME : STRING34;
  ENT_DATE : STRING6;
  MAR_STAT : CHAR;
  DISP : CHAR;
  VET,REPT : CHAR;
  ADVISOR,ADV : STRING8;
  MAJOR,DEP,CRS,SEM,LC1,LC2,LC3,LC4,LC5 : STRING3;
  DEGREE,HR,GRD, : STRING2;
  SAT_VERB : STRING3;
  SAT_MATH : STRING3;
  TEACH_CERT : CHAR;
  YEAR_GRAD : STRING2;
  CLASS : STRING1;
  TRANS_HOUR : STRING5;
  CUM_ATT : STRING5;
  CUM_PASS : STRING5;
  CUM_GPA : STRING4;
  BLANK : CHAR;
  FILLER : CHAR;
  FILLER7 : STRING7;
  FILLER2 : STRING2;
  FILLER5 : STRING5;
  FILLER6 :STRING6;
  FILLER10 : STRING10;
  FILLER11 : STRING11;
  FILLER13 : STRING13;
  FILLER14 : STRING14;
  FILLER20 : STRING20;
  FILLER25 : STRING25;
  FILLER39 : STRING39;
  DISP_NUMBER : REAL;
  FILLER95 : STRING95;
```

```
  CUM_PASS : STRING5;
  CUM_GPA : STRING4;
  BLANK : CHAR;
  FILLER : CHAR;
  FILLER7 : STRING7;
  FILLER2 : STRING2;
  FILLER5 : STRING5;
  FILLER6 :STRING6;
  FILLER10 : STRING10;
  FILLER11 : STRING11;
  FILLER13 : STRING13;
  FILLER14 : STRING14;
  FILLER20 : STRING20;
  FILLER25 : STRING25;
  FILLER39 : STRING39;
  DISP_NUMBER : REAL;
  FILLER95 : STRING95;
  FILLER8 : STRING8;
  FILLER12 : STRING12;
  M_SSN,E_SSN : STRING9;
  M_NAME,E_NAME : STRING34;
  M_DEPT,E_DEPT : STRING3;
  M_CRS,E_CRS :STRING3;
  M_HOUR,E_HOUR : STRING2;
  M_GRADE,E_GRADE : STRING2;
  M_SEM,E_SEM : STRING3;

PROCEDURE CHECK_BASIC (VAR CLSFILE : FILE OF CHAR;
                       SSAN: STRING9;
                       VAR CRSFILE : FILE OF CHAR;
                       VAR SSN: STRING9;
                       NOME : STRING34; ADVISOR : STRING8;
                       REPT : CHAR; DEP,CRS,SEM,LC1,LC2,LC3,LC4,LC5 : STRING3;
                       HR,GRD : STRING2; MAJOR : STRING3);

BEGIN
  WHILE (SSN = SSAN) AND (NOT EOF(CRSFILE)) DO
    BEGIN
      WRITE(CLSFILE,SSN,NOME,ADVISOR);
      WRITE(CLSFILE,MAJOR,DEP,CRS,HR,GRD,SEM);
      WRITELN(CLSFILE,REPT,LC1,LC2,LC3,LC4,LC5);
      IF NOT EOF(CRSFILE) THEN
         BEGIN
           READ(CRSFILE,SSN,NOME,DEP,CRS,HR,GRD,SEM);
           READLN(CRSFILE,REPT,LC1,LC2,LC3,LC4,LC5);
         END
      ELSE
        SSN:= '         ';
    END;
END;

PROCEDURE CHECK_MAJOR (VAR MAJOR_FILE : FILE OF CHAR;
                       VAR MAJ_FILE : FILE OF CHAR;
                       SSAN : STRING9;
                       VAR M_SSN : STRING9;
                       M_NAME : STRING34;M_DEPT : STRING3;
                       ADVISOR : STRING8; M_CRS : STRING3;
                       M_SEM : STRING3; M_HOUR : STRING2;
                       M_GRADE : STRING2; MAJOR : STRING3);

BEGIN
  WHILE (M_SSN = SSAN) AND (NOT EOF(MAJOR_FILE)) DO
     BEGIN
       WRITE (MAJ_FILE,M_SSN,M_NAME,ADVISOR);
       WRITE(MAJ_FILE,MAJOR,M_DEPT,M_CRS,M_HOUR);
       WRITELN(MAJ_FILE,M_GRADE,M_SEM);
       IF NOT EOF(MAJOR_FILE) THEN
         BEGIN
           READ(MAJOR_FILE,M_SSN,M_NAME,M_DEPT);
           READ(MAJOR_FILE,M_CRS,M_HOUR,M_GRADE);
           READLN(MAJOR_FILE,M_SEM);
         END
       ELSE
         M_SSN:= '         ';
     END;
END;


PROCEDURE CHECK_EXTRA (VAR EX_FILE : FILE OF CHAR;
                       VAR EXTRA_FILE : FILE OF CHAR;
                       SSAN : STRING9;
                       VAR E_SSN : STRING9;
```

```
        WRITE(MAJ_FILE,MAJOR,M_DEPT,M_CRS,M_HOUR);
        WRITELN(MAJ_FILE,M_GRADE,M_SEM);
        IF NOT EOF(MAJOR_FILE) THEN
          BEGIN
            READ(MAJOR_FILE,M_SSN,M_NAME,M_DEPT);
            READ(MAJOR_FILE,M_CRS,M_HOUR,M_GRADE);
            READLN(MAJOR_FILE,M_SEM);
          END
        ELSE
          M_SSN:= '         ';
      END;
END;


PROCEDURE CHECK_EXTRA (VAR EX_FILE : FILE OF CHAR;
                       VAR EXTRA_FILE : FILE OF CHAR;
                       SSAN : STRING9;
                       VAR E_SSN : STRING9;
                       E_NAME : STRING34;E_DEPT : STRING3;
                       ADVISOR : STRING8; E_CRS : STRING3;
                       E_SEM : STRING3; E_HOUR : STRING2;
                       E_GRADE : STRING2; MAJOR : STRING3);

BEGIN
  WHILE (E_SSN = SSAN) AND (NOT EOF(EX_FILE)) DO
    BEGIN
      WRITE (EXTRA_FILE,E_SSN,E_NAME,ADVISOR);
      WRITE(EXTRA_FILE,MAJOR,E_DEPT,E_CRS,E_HOUR);
      WRITELN(EXTRA_FILE,E_GRADE,E_SEM);
      IF NOT EOF(EX_FILE) THEN
        BEGIN
          READ(EX_FILE,E_SSN,E_NAME,E_DEPT);
          READ(EX_FILE,E_CRS,E_HOUR,E_GRADE);
          READLN(EX_FILE,E_SEM);
        END
      ELSE
        E_SSN:= '         ';
    END;
END;

BEGIN
  RESET(DUMMY);
  RESET(CRSFILE);
  REWRITE(NEWFILE);
  REWRITE(CLSFILE);
  RESET(MAJOR_FILE);
  REWRITE(MAJ_FILE);
  RESET(EX_FILE);
  REWRITE(EXTRA_FILE);
  READ(DUMMY,BLANK,SSAN,NAME,FILLER2,ENT_DATE,FILLER6);
  READ(DUMMY,MAR_STAT,FILLER5,DISP,FILLER13,VET);
  READ(DUMMY,FILLER2,ADVISOR,FILLER6,MAJOR,DEGREE);
  READ(DUMMY,SAT_VERB,FILLER2,SAT_MATH,FILLER39);
  READ(DUMMY,TEACH_CERT,YEAR_GRAD,FILLER20,CLASS,FILLER11);
  READ(DUMMY,TRANS_HOUR,FILLER14,CUM_ATT);
  READLN(DUMMY,CUM_PASS,FILLER10,CUM_GPA,FILLER95);
  READ(CRSFILE,SSN,NOME,DEP,CRS,HR,GRD,SEM);
  READLN(CRSFILE,REPT,LC1,LC2,LC3,LC4,LC5);
  READ(MAJOR_FILE,M_SSN,M_NAME,M_DEPT,M_CRS);
  READLN(MAJOR_FILE,M_HOUR,M_GRADE,M_SEM);
  READ(EX_FILE,E_SSN,E_NAME,E_DEPT,E_CRS);
  READLN(EX_FILE,E_HOUR,E_GRADE,E_SEM);
  WHILE NOT EOF(DUMMY) DO
    BEGIN
      DISP_NUMBER:= TRUNC(STOR(CLASS));
      IF DISP_NUMBER < 5 THEN
        BEGIN
          WRITE(NEWFILE,SSAN,NAME,ENT_DATE,MAR_STAT,DISP);
          WRITE(NEWFILE,VET,ADVISOR,MAJOR,DEGREE,SAT_VERB);
          WRITE(NEWFILE,SAT_MATH,TEACH_CERT,YEAR_GRAD,CLASS);
          WRITELN(NEWFILE,TRANS_HOUR,CUMM_ATT,CUM_PASS,CUM_GPA);
          IF NOT EOF(CRSFILE) THEN
              CHECK_BASIC(CLSFILE,SSAN,CRSFILE,SSN,NOME,
                          ADVISOR,REPT,DEP,CRS,SEM,LC1,LC2,
                          LC3,LC4,LC5,HR,GRAD,MAJOR);
          IF NOT EOF(MAJOR_FILE) THEN
            CHECK_MAJOR(MAJOR_FILE,MAJ_FILE,SSAN,M_SSN,M_NAME,
                        M_DEPT,ADVISOR,M_CRS,M_SEM,M_HOUR,
                        M_GRADE,MAJOR);
          IF NOT EOF(EX_FILE) THEN
             CHECK_EXTRA(EX_FILE,EXTRA_FILE,SSAN,E_SSN,E_NAME,
                         E_DEPT,ADVISOR,E_CRS,E_SEM,E_HOUR,
                         E_GRADE,MAJOR);
        END;
```

```
        BEGIN
          WRITE(NEWFILE,SSAN,NAME,ENT_DATE,MAR_STAT,DISP);
          WRITE(NEWFILE,VET,ADVISOR,MAJOR,DEGREE,SAT_VERB);
          WRITE(NEWFILE,SAT_MATH,TEACH_CERT,YEAR_GRAD,CLASS);
          WRITELN(NEWFILE,TRANS_HOUR,CUMM_ATT,CUM_PASS,CUM_GPA);
          IF NOT EOF(CRSFILE) THEN
              CHECK_BASIC(CLSFILE,SSAN,CRSFILE,SSN,NOME,
                          ADVISOR,REPT,DEP,CRS,SEM,LC1,LC2,
                          LC3,LC4,LC5,HR,GRAD,MAJOR);
          IF NOT EOF(MAJOR_FILE) THEN
            CHECK_MAJOR(MAJOR_FILE,MAJ_FILE,SSAN,M_SSN,M_NAME,
                        M_DEPT,ADVISOR,M_CRS,M_SEM,M_HOUR,
                        M_GRADE,MAJOR);
          IF NOT EOF(EX_FILE) THEN
             CHECK_EXTRA(EX_FILE,EXTRA_FILE,SSAN,E_SSN,E_NAME,
                         E_DEPT,ADVISOR,E_CRS,E_SEM,E_HOUR,
                         E_GRADE,MAJOR);
        END;
      READ(DUMMY,BLANK,SSAN,NAME,FILLER2,ENT_DATE,FILLER6);
      READ(DUMMY,MAR_STAT,FILLER5,DISP,FILLER13,VET);
      READ(DUMMY,FILLER2,ADVISOR,FILLER6,MAJOR,DEGREE);
      READ(DUMMY,SAT_VERB,FILLER2,SAT_MATH,FILLER39);
      READ(DUMMY,TEACH_CERT,YEAR_GRAD,FILLER20,CLASS,FILLER11);
      READ(DUMMY,TRANS_HOUR,FILLER14,CM_ATT);
      READLN(DUMMY,CUM_PASS,FILLER10,CUM_GPA,FILLER95);
    END;
  WRITE(NEWFILE,SSAN,NAME,ENT_DATE,MAR_STAT,DISP);
  WRITE(NEWFILE,VET,ADVISOR,MAJOR,DEGREE,SAT_VERB);
  WRITE(NEWFILE,SAT_MATH,TEACH_CERT,YEAR_GRAD,CLASS);
  WRITELN(NEWFILE,TRANS_HOUR,CUMM_ATT,CUM_PASS,CUM_GPA);
  IF NOT EOF(CRSFILE) THEN
     CHECK_BASIC(CLSFILE,SSAN,CRSFILE,SSN,NOME,
                 ADVISOR,REPT,DEP,CRS,SEM,LC1,LC2,
                 LC3,LC4,LC5,HR,GRAD,MAJOR);
  IF NOT EOF(MAJOR_FILE) THEN
     CHECK_MAJOR(MAJOR_FILE,MAJ_FILE,SSAN,M_SSN,M_NAME,
                 M_DEPT,ADVISOR,M_CRS,M_SEM,M_HOUR,
                 M_GRADE,MAJOR);
  IF NOT EOF(EX_FILE) THEN
     CHECK_EXTRA(EX_FILE,EXTRA_FILE,SSAN,E_SSN,E_NAME,
                 E_DEPT,ADVISOR,E_CRS,E_SEM,E_HOUR,
                 E_GRADE,MAJOR);
END.
```

```
PROGRAM PROCESS(OUTPUT,GRADES,NEWFILE,MAJ_FILE,EX_FILE);
TYPE
  STR1 = PACKED ARRAY[1..1] OF CHAR;
  CLASS_REC    = RECORD
                   ADVISOR     : PACKED ARRAY[1..8] OF CHAR;
                   COURSE      : PACKED ARRAY[1..3] OF CHAR;
                   DEPT        : PACKED ARRAY[1..3] OF CHAR;
                   GRADE       : PACKED ARRAY[1..2] OF CHAR;
                   HOURS       : PACKED ARRAY[1..2] OF CHAR;
                   LOC1        : PACKED ARRAY[1..3] OF CHAR;
                   LOC2        : PACKED ARRAY[1..3] OF CHAR;
                   LOC3        : PACKED ARRAY[1..3] OF CHAR;
                   LOC4        : PACKED ARRAY[1..3] OF CHAR;
                   LOC5        : PACKED ARRAY[1..3] OF CHAR;
                   MAJOR       : PACKED ARRAY[1..3] OF CHAR;
                   NAME        : PACKED ARRAY[1..34] OF CHAR;
                   RPT         : PACKED ARRAY [1..1] OF CHAR;
                   SEMESTER    : PACKED ARRAY[1..3] OF CHAR;
                   SSAN        : PACKED ARRAY[1..9] OF CHAR;
                 END;

  COURSE_REC   = RECORD
                   COURSE_NUM  : PACKED ARRAY[1..3] OF CHAR;
                   CRSE_HOURS  : PACKED ARRAY[1..2] OF CHAR;
                   DEPT_CODE   : PACKED ARRAY[1..3] OF CHAR;
                   CRSE_GRADE  : PACKED ARRAY[1..2] OF CHAR;
                   LOC1        : PACKED ARRAY[1..3] OF CHAR;
                   LOC2        : PACKED ARRAY[1..3] OF CHAR;
                   LOC3        : PACKED ARRAY[1..3] OF CHAR;
                   LOC4        : PACKED ARRAY[1..3] OF CHAR;
                   LOC5        : PACKED ARRAY[1..3] OF CHAR;
                   RPT         : PACKED ARRAY[1..1] OF CHAR;
                   SEMESTER    : PACKED ARRAY[1..3] OF CHAR;
                 END;

  STUDENT_INFO = RECORD
                    ADVISOR       : PACKED ARRAY[1..8] OF CHAR;
                    CLASS         : STR1;
                    CUM_ATMP      : PACKED ARRAY[1..5] OF CHAR;
                    CUM_GPA       : PACKED ARRAY[1..5] OF CHAR;
                    CUM_PASS      : PACKED ARRAY[1..5] OF CHAR;
                    DEGREE        : PACKED ARRAY[1..2] OF CHAR;
                    DISPOSITION   : CHAR;
                    ENTERED       : PACKED ARRAY[1..6] OF CHAR;
                    HS_GRAD       : PACKED ARRAY[1..2] OF CHAR;
                    MAJOR         : PACKED ARRAY[1..3] OF CHAR;
                    MARRIED       : CHAR;
                    NAME          : PACKED ARRAY[1..34] OF CHAR;
                    SAT_MATH      : PACKED ARRAY[1..3] OF CHAR;
                    SAT_VERB      : PACKED ARRAY[1..3] OF CHAR;
                    SSAN          : PACKED ARRAY[1..9] OF CHAR;
                    TEACHER_CERT  : CHAR;
                    TRAN_PASS     : PACKED ARRAY[1..5] OF CHAR;
                    VETERAN       : CHAR;
                  END;

  BAS_SKILL    = RECORD
                   CLASS1 : COURSE_REC;
                   CLASS2 : COURSE_REC;
                   SKILLS_TOT : INTEGER;
                 END;

  HUMAN        = RECORD
                   FINE_ARTS : COURSE_REC;
                   ARTS_TOTAL : INTEGER;
                   LITERATURE : COURSE_REC;
                   LIT_TOTAL : INTEGER;
                   HISTORY : COURSE_REC;
                   HISTORY_TOTAL : INTEGER;
                   PHIL_REL : COURSE_REC;
                   PHIL_REL_TOTAL : INTEGER;
                   HUM_ELECTIVES : RECORD
                                     AREA1 : COURSE_REC;
                                     AREA1_TOT : INTEGER;
                                     AREA2 : COURSE_REC;
                                     AREA2_TOT : INTEGER;
                                     AREA3 : COURSE_REC;
                                     AREA3_TOT : INTEGER;
                                     AREA4 : COURSE_REC;
                                     AREA4_TOT : INTEGER;
                                     AREA5 : COURSE_REC;
```

```
                    FINE_ARTS : COURSE_REC;
                    ARTS_TOTAL : INTEGER;
                    LITERATURE : COURSE_REC;
                    LIT_TOTAL : INTEGER;
                    HISTORY : COURSE_REC;
                    HISTORY_TOTAL : INTEGER;
                    PHIL_REL : COURSE_REC;
                    PHIL_REL_TOTAL : INTEGER;
                    HUM_ELECTIVES : RECORD
                                      AREA1 : COURSE_REC;
                                      AREA1_TOT : INTEGER;
                                      AREA2 : COURSE_REC;
                                      AREA2_TOT : INTEGER;
                                      AREA3 : COURSE_REC;
                                      AREA3_TOT : INTEGER;
                                      AREA4 : COURSE_REC;
                                      AREA4_TOT : INTEGER;
                                      AREA5 : COURSE_REC;
                                      AREA5_TOT : INTEGER;
                                    END;
                    ELEC_TOTAL : INTEGER;
                    HUMAN_TOTAL : INTEGER;
                  END;

SOCIAL       = RECORD
                    ECONOMICS : COURSE_REC;
                    ECON_TOTAL : INTEGER;
                    GEOGRAPHY : COURSE_REC;
                    GEOGR_TOTAL : INTEGER;
                    POLITICAL_SCIENCE : COURSE_REC;
                    POL_SCI_TOTAL : INTEGER;
                    PSYCHOLOGY : COURSE_REC;
                    PSY_TOTAL : INTEGER;
                    SOCIOLOGY :  COURSE_REC;
                    SOC_TOTAL : INTEGER;
                    SOC_SCI_TOTAL : INTEGER;
                  END;

SCIENCE      = RECORD
                    BIOLOGY : COURSE_REC;
                    BIO_TOTAL : INTEGER;
                    PHYSICAL_SCIENCE : COURSE_REC;
                    PHY_SCI_TOTAL : INTEGER;
                    MATH : COURSE_REC;
                    MATH_TOTAL : INTEGER;
                    MATH_ELECTIVES : COURSE_REC;
                    ELEC_TOTAL : INTEGER;
                    SCI_MATH_TOTAL : INTEGER;
                  END;

PHYSICAL     = RECORD
                    P_E :ARRAY[1..2] OF COURSE_REC;
                    P_E_TOTAL : INTEGER;
                  END;

STUDENT_REC = RECORD
                    BASIC_SKILLS : BAS_SKILL;
                    HUMANITIES : HUMAN;
                    SOCIAL_SCIENCES : SOCIAL;
                    SCIENCE_MATHEMATICS : SCIENCE;
                    PHYSICAL_EDUCATION : PHYSICAL;
                  END;
CL_RECORD = RECORD
                    SSAN : PACKED ARRAY[1..9] OF CHAR;
                    NAME : PACKED ARRAY[1..34] OF CHAR;
                    DEPT : PACKED ARRAY[1..3] OF CHAR;
                    COURSE : PACKED ARRAY[1..3] OF CHAR;
                    HOUR : PACKED ARRAY[1..2] OF CHAR;
                    GRADE : PACKED ARRAY [1..2] OF CHAR;
                    SEMESTER : PACKED ARRAY[1..3] OF CHAR;
                  END;
MATH_RECORD = RECORD
                    REQ_107 : ARRAY[1..2] OF CL_RECORD;
                    REQ_220 : CL_RECORD;
                    REQ_221 : CL_RECORD;
                    REQ_222 : CL_RECORD;
                    REQ_315 : CL_RECORD;
                    REQ_316 : CL_RECORD;
                    REQ_325 : CL_RECORD;
                    REQ_431 : CL_RECORD;
                    REQ_EX  : ARRAY[1..5] OF CL_RECORD;
                    ELEC_TOTAL : INTEGER;
                    REQ_COUNT : INTEGER;
                  END;
```

```
                COURSE : PACKED ARRAY[1..3] OF CHAR;
                HOUR : PACKED ARRAY[1..2] OF CHAR;
                GRADE : PACKED ARRAY [1..2] OF CHAR;
                SEMESTER : PACKED ARRAY[1..3] OF CHAR;
              END;
  MATH_RECORD = RECORD
                  REQ_107 : ARRAY[1..2] OF CL_RECORD;
                  REQ_220 : CL_RECORD;
                  REQ_221 : CL_RECORD;
                  REQ_222 : CL_RECORD;
                  REQ_315 : CL_RECORD;
                  REQ_316 : CL_RECORD;
                  REQ_325 : CL_RECORD;
                  REQ_431 : CL_RECORD;
                  REQ_EX  : ARRAY[1..5] OF CL_RECORD;
                  ELEC_TOTAL : INTEGER;
                  REQ_COUNT : INTEGER;
                END;
  CSC_RECORD = RECORD
                  REQ_107 : ARRAY[1..2] OF CL_RECORD;;
                  REQ_221 : CL_RECORD;
                  REQ_222 : CL_RECORD;
                  REQ_315 : CL_RECORD;
                  REQ_316 : CL_RECORD;
                  REQ_317 : CL_RECORD;
                  REQ_155 : CL_RECORD;
                  REQ_200 : CL_RECORD;
                  REQ_250 : CL_RECORD;
                  REQ_270 : CL_RECORD;
                  REQ_350 : CL_RECORD;
                  REQ_420 : CL_RECORD;
                  REQ_450 : CL_RECORD;
                  REQ_210 : CL_RECORD;
                  REQ_370 : CL_RECORD;
               END;


VAR
  CONTINUE      : BOOLEAN;
  GRADES        : FILE OF CHAR;
  NEWFILE       : FILE OF CHAR;
  GRADE_DATA    : CLASS_REC;
  GRAND_TOTAL   : INTEGER;
  LOC           : CHAR;
  READ_REC      : BOOLEAN;
  REPORT        : STUDENT_REC;
  STUDENT_DATA  : STUDENT_INFO;
  MAJOR_DATA    : CL_RECORD;
  EX_DATA       : CL_RECORD;
  MAJ_ARRAY     : ARRAY[1..30] OF MAJOR_DATA
  EX_ARRAY      : ARRAY [1..50] OF EX_DATA;
  MAJ_REPORT    : MAJ_ARRAY;
  EX_REPORT     : EX_ARRAY;
  OVERALL_TOT   : INTEGER;
  MAJOR_TOT     : INTEGER;
  EXTRA_TOT     : INTEGER;
  COUNT         : INTEGER;
  BASIC_ARRAY   : ARRAY[1..50] OF CLASS_REC;
  EXTRA_BASIC   : BASIC_ARRAY;
  BASIC_COUNT   : INTEGER;
  MAT_REP       : MATH_RECORD;
  CSC_REP       : CSC_RECORD;
  MAJOR_COUNT : INTEGER;
  EXTRA_COUNT : INTEGER;
  B_COUNT : INTEGER;
  MAJ_FILE : FILE OF CHAR;
  EX_FILE : FILE OF CHAR;



PROCEDURE PART_A(GRADE_DATA : CLASS_REC;
                 VAR AREA_A : BAS_SKILL;
                 VAR GRAND_TOTAL : INTEGER);

VAR
  TEMP_HRS : INTEGER;
  CHR      : STR1;

  BEGIN
    SUBSTR(GRADE_DATA.GRADE,1,1,CHR);
    IF (CHR <= "C") AND (GRADE_DATA.COURSE = '105')
      THEN
        BEGIN
```

```
  EX_FILE : FILE OF CHAR);


PROCEDURE PART_A(GRADE_DATA : CLASS_REC;
                 VAR AREA_A : BAS_SKILL;
                 VAR GRAND_TOTAL : INTEGER);

VAR
  TEMP_HRS : INTEGER;
  CHR      : STR1;

  BEGIN
    SUBSTR(GRADE_DATA.GRADE,1,1,CHR);
    IF (CHR <= "C") AND (GRADE_DATA.COURSE = '105')
      THEN
        BEGIN
          TEMP_HRS := TRUNC(STOR(GRADE_DATA.HOURS));
          GRAND_TOTAL := GRAND_TOTAL + TEMP_HRS;
          AREA_A.SKILLS_TOT := AREA_A.SKILLS_TOT + TEMP_HRS;
          AREA_A.CLASS1.CRSE_HOURS := GRADE_DATA.HOURS;
          AREA_A.CLASS1.COURSE_NUM := GRADE_DATA.COURSE;
          AREA_A.CLASS1.DEPT_CODE := GRADE_DATA.DEPT;
          AREA_A.CLASS1.CRSE_GRADE := GRADE_DATA.GRADE;
          AREA_A.CLASS1.SEMESTER := GRADE_DATA.SEMESTER;
          AREA_A.CLASS1.LOC1 := GRADE_DATA.LOC1;
          AREA_A.CLASS1.LOC2 := GRADE_DATA.LOC2;
          AREA_A.CLASS1.LOC3 := GRADE_DATA.LOC3;
          AREA_A.CLASS1.LOC4 := GRADE_DATA.LOC4;
          AREA_A.CLASS1.LOC5 := GRADE_DATA.LOC5;
          AREA_A.CLASS1.RPT := GRADE_DATA.RPT;
        END;
    IF (CHR <= "C") AND (GRADE_DATA.COURSE = '106')
      THEN
        BEGIN
          TEMP_HRS := TRUNC(STOR(GRADE_DATA.HOURS));
          GRAND_TOTAL := GRAND_TOTAL + TEMP_HRS;
          AREA_A.SKILLS_TOT := AREA_A.SKILLS_TOT + TEMP_HRS;
          AREA_A.CLASS2.CRSE_HOURS := GRADE_DATA.HOURS;
          AREA_A.CLASS2.COURSE_NUM := GRADE_DATA.COURSE;
          AREA_A.CLASS2.DEPT_CODE := GRADE_DATA.DEPT;
          AREA_A.CLASS2.CRSE_GRADE := GRADE_DATA.GRADE;
          AREA_A.CLASS2.SEMESTER := GRADE_DATA.SEMESTER;
          AREA_A.CLASS2.LOC1 := GRADE_DATA.LOC1;
          AREA_A.CLASS2.LOC2 := GRADE_DATA.LOC2;
          AREA_A.CLASS2.LOC3 := GRADE_DATA.LOC3;
          AREA_A.CLASS2.LOC4 := GRADE_DATA.LOC4;
          AREA_A.CLASS2.LOC5 := GRADE_DATA.LOC5;
          AREA_A.CLASS2.RPT := GRADE_DATA.RPT;
        END;
  END;


PROCEDURE PART_E(GRADE_DATA : CLASS_REC;
                 VAR AREA_E : PHYSICAL;
                 VAR GRAND_TOTAL : INTEGER;
                 COUNT : INTEGER;
                 VAR EXTRA_BASIC : BASIC_ARRAY);

  VAR
    TEMP_HRS : INTEGER;

  BEGIN
    IF AREA_E.P_E_TOTAL = 0
      THEN
        BEGIN
          TEMP_HRS := TRUNC(STOR(GRADE_DATA.HOURS));
          GRAND_TOTAL := GRAND_TOTAL + TEMP_HRS;
          AREA_E.P_E_TOTAL := AREA_E.P_E_TOTAL + TEMP_HRS;
          AREA_E.P_E[1].CRSE_HOURS := GRADE_DATA.HOURS;
          AREA_E.P_E[1].COURSE_NUM := GRADE_DATA.COURSE;
          AREA_E.P_E[1].DEPT_CODE := GRADE_DATA.DEPT;
          AREA_E.P_E[1].CRSE_GRADE := GRADE_DATA.GRADE;
          AREA_E.P_E[1].SEMESTER := GRADE_DATA.SEMESTER;
          AREA_E.P_E[1].LOC1 := GRADE_DATA.LOC1;
          AREA_E.P_E[1].LOC2 := GRADE_DATA.LOC2;
          AREA_E.P_E[1].LOC3 := GRADE_DATA.LOC3;
          AREA_E.P_E[1].LOC4 := GRADE_DATA.LOC4;
          AREA_E.P_E[1].LOC5 := GRADE_DATA.LOC5;
          AREA_E.P_E[1].RPT := GRADE_DATA.RPT;
        END
      ELSE
        IF AREA_E.P_E_TOTAL = 1
```

```
        THEN
          BEGIN
            TEMP_HRS := TRUNC(STOR(GRADE_DATA.HOURS));
            GRAND_TOTAL := GRAND_TOTAL + TEMP_HRS;
            AREA_E.P_E_TOTAL := AREA_E.P_E_TOTAL + TEMP_HRS;
            AREA_E.P_E[1].CRSE_HOURS := GRADE_DATA.HOURS;
            AREA_E.P_E[1].COURSE_NUM := GRADE_DATA.COURSE;
            AREA_E.P_E[1].DEPT_CODE := GRADE_DATA.DEPT;
            AREA_E.P_E[1].CRSE_GRADE := GRADE_DATA.GRADE;
            AREA_E.P_E[1].SEMESTER := GRADE_DATA.SEMESTER;
            AREA_E.P_E[1].LOC1 := GRADE_DATA.LOC1;
            AREA_E.P_E[1].LOC2 := GRADE_DATA.LOC2;
            AREA_E.P_E[1].LOC3 := GRADE_DATA.LOC3;
            AREA_E.P_E[1].LOC4 := GRADE_DATA.LOC4;
            AREA_E.P_E[1].LOC5 := GRADE_DATA.LOC5;
            AREA_E.P_E[1].RPT := GRADE_DATA.RPT;
          END
      ELSE
         IF AREA_E.P_E_TOTAL = 1
           THEN
             BEGIN
               TEMP_HRS := TRUNC(STOR(GRADE_DATA.HOURS));
               GRAND_TOTAL := GRAND_TOTAL + TEMP_HRS;
               AREA_E.P_E_TOTAL := AREA_E.P_E_TOTAL + TEMP_HRS;
               AREA_E.P_E[2].CRSE_HOURS := GRADE_DATA.HOURS;
               AREA_E.P_E[2].COURSE_NUM := GRADE_DATA.COURSE;
               AREA_E.P_E[2].DEPT_CODE := GRADE_DATA.DEPT;
               AREA_E.P_E[2].CRSE_GRADE := GRADE_DATA.GRADE;
               AREA_E.P_E[2].SEMESTER := GRADE_DATA.SEMESTER;
               AREA_E.P_E[2].LOC1 := GRADE_DATA.LOC1;
               AREA_E.P_E[2].LOC2 := GRADE_DATA.LOC2;
               AREA_E.P_E[2].LOC3 := GRADE_DATA.LOC3;
               AREA_E.P_E[2].LOC4 := GRADE_DATA.LOC4;
               AREA_E.P_E[2].LOC5 := GRADE_DATA.LOC5;
               AREA_E.P_E[2].RPT := GRADE_DATA.RPT;
             END
         ELSE
           EXTRA_BASIC[COUNT]:= GRADE_DATA;
  END;

PROCEDURE PRINT(STUDENT_DATA : STUDENT_INFO;
                REPORT       : STUDENT_REC;
                GRAND_TOTAL  : INTEGER;
                MAJOR_COUNT : INTEGER;
                EXTRA_COUNT : INTEGER);
TYPE
  STR3 = PACKED ARRAY[1..3] OF CHAR;
  STR9 = PACKED ARRAY[1..9] OF CHAR;
VAR
  TRANSFER : BOOLEAN;

FUNCTION TRANSFER_STUDENT(DISPOSITION : CHAR) : BOOLEAN;
BEGIN
    IF (DISPOSITION = 'I') OR (DISPOSITION = 'T') OR
       (DISPOSITION = 'E') OR (DISPOSITION = 'F') OR
       (DISPOSITION = 'Q') THEN
         TRANSFER_STUDENT:= TRUE
    ELSE
         TRANSFER_STUDENT:= FALSE;
END;

PROCEDURE CONVERT_SEMESTER(SEMESTER : STR3;
                           VAR SEM_TAKEN : STR9);
VAR
  QUARTER : CHAR;
  TEMP_STR : PACKED ARRAY[1..2] OF CHAR;
BEGIN
  QUARTER:= SEMESTER[1];
  CASE QUARTER OF
    '1' : SEM_TAKEN := '  FALL ';
    '2' : SEM_TAKEN := 'SPRING ';
    '3' : SEM_TAKEN := 'SUMMER ';
    '4' : SEM_TAKEN := 'SUMMER ';
  END;
  SUBSTR(SEMESTER,2,2,TEMP_STR);
  STRINSERT(SEM_TAKEN,TEMP_STR,8);
END;

PROCEDURE PRINT_HEADER(STUD : STUDENT_INFO);
TYPE
  STR3 = PACKED ARRAY[1..3] OF CHAR;
  STR6 = PACKED ARRAY[1..6] OF CHAR;
  STR7 = PACKED ARRAY[1..7] OF CHAR;
  STR8 = PACKED ARRAY[1..8] OF CHAR;
```

```
    QUARTER:= SEMESTER[1];
    CASE QUARTER OF
      '1' : SEM_TAKEN := '  FALL ';
      '2' : SEM_TAKEN := 'SPRING ';
      '3' : SEM_TAKEN := 'SUMMER ';
      '4' : SEM_TAKEN := 'SUMMER ';
    END;
    SUBSTR(SEMESTER,2,2,TEMP_STR);
    STRINSERT(SEM_TAKEN,TEMP_STR,8);
  END;

  PROCEDURE PRINT_HEADER(STUD : STUDENT_INFO);
  TYPE
    STR3 = PACKED ARRAY[1..3] OF CHAR;
    STR6 = PACKED ARRAY[1..6] OF CHAR;
    STR7 = PACKED ARRAY[1..7] OF CHAR;
    STR8 = PACKED ARRAY[1..8] OF CHAR;
    STR9 = PACKED ARRAY[1..9] OF CHAR;
    STR11 = PACKED ARRAY[1..11] OF CHAR;
  VAR
    NEW_CLASS : STR9;
    NEW_DATE  : STR8;
    RESULT    : STR3;
    SOC_SEC   : STR11;
    STATUS    : STR7;

  PROCEDURE CONVERT_DATE(ENTERED : STR6;
                         VAR NEW_DATE : STR8);
  BEGIN
     NEW_DATE := ENTERED;
     STRINSERT(NEW_DATE,'/',3);
     STRINSERT(NEW_DATE,'/',6);
  END;

  PROCEDURE CONVERT_BOOLEAN(INPUT : CHAR;
                            VAR RESULT : STR3);
  BEGIN
    IF INPUT = '1'
      THEN
        RESULT := 'YES'
     ELSE
        RESULT := ' NO';
  END;


  PROCEDURE CONVERT_TEACH (INPUT : CHAR;
                           VAR RESULT : STR3);

  BEGIN
    IF INPUT = 'Y' THEN
      RESULT:= 'YES'
    ELSE
      RESULT:= 'NO';
  END;

  PROCEDURE CONVERT_VET (INPUT : CHAR;
                         VAR RESULT : STR3);

  BEGIN
    IF INPUT = '0' THEN
      RESULT:= 'YES'
    ELSE
      RESULT:= 'NO';
  END;

  PROCEDURE CONVERT_MARITAL(INPUT : CHAR;
                            VAR STATUS : STR7);
  BEGIN
     STATUS:= "  ";
     STATUS[1]:= INPUT;
  END;

  PROCEDURE CONVERT_SSAN(SSAN : STR9;
                         VAR SOC_SEC : STR11);
  BEGIN
    SOC_SEC := SSAN;
    STRINSERT(SOC_SEC,'-',4);
    STRINSERT(SOC_SEC,'-',7);
  END;

  PROCEDURE CONVERT_CLASS(CLASS : STR1;
                          VAR NEW_CLASS : STR9);

  VAR
```

```
                        VAR STATUS : STR7);
BEGIN
   STATUS:= "  ";
   STATUS[1]:= INPUT;
END;

PROCEDURE CONVERT_SSAN(SSAN : STR9;
                       VAR SOC_SEC : STR11);
BEGIN
  SOC_SEC := SSAN;
  STRINSERT(SOC_SEC,'-',4);
  STRINSERT(SOC_SEC,'-',7);
END;

PROCEDURE CONVERT_CLASS(CLASS : STR1;
                        VAR NEW_CLASS : STR9);

VAR
  TEMP : CHAR;

BEGIN
  TEMP:= CLASS[1];
  IF TEMP = '1' THEN
    NEW_CLASS:= 'FRESHMAN'
  ELSE
    IF TEMP = '2' THEN
      NEW_CLASS:= 'SOPHOMORE'
    ELSE
      IF TEMP = '3' THEN
        NEW_CLASS:= 'JUNIOR'
      ELSE
        IF TEMP = '4' THEN
          NEW_CLASS:= 'SENIOR'
        ELSE
          NEW_CLASS:= '         ';
END;

BEGIN
  WRITELN('BASIC STUDIES EVALUATION':52);
  WRITELN;
  CONVERT_SSAN(STUD.SSAN,SOC_SEC);
  WRITE(STUD.NAME,'   ',SOC_SEC);
  WRITE('  MAJOR: ',STUD.MAJOR);
  WRITELN('  DEGREE: ',STUD.DEGREE);
  WRITE('ADVISOR: ',STUD.ADVISOR);
  CONVERT_CLASS(STUD.CLASS,NEW_CLASS);
  WRITE('  CLASS: ',NEW_CLASS);
  CONVERT_DATE(STUD.ENTERED,NEW_DATE);
  WRITE('  ENTRANCE DATE: ',NEW_DATE);
  CONVERT_VET(STUD.VETERAN,RESULT);
  WRITELN('  VETERAN: ',RESULT);
  WRITE('DISPOSITION: ',STUD.DISPOSITION);
  CONVERT_TEACH(STUD.TEACHER_CERT,RESULT);
  WRITE('  TEACHER CERTIFICATION: ',RESULT);
  CONVERT_MARITAL(STUD.MARRIED,STATUS);
  WRITELN('    MARITAL STATUS: ',STATUS);
  WRITE('YEAR GRADUATED H.S.: 19',STUD.HS_GRAD);
  WRITE('  SAT VERBAL: ',STUD.SAT_VERB);
  WRITELN('  SAT MATH: ',STUD.SAT_MATH);
  WRITELN;
  WRITE('CATEGORY','CLASS':13,'DATE TAKEN':16);
  WRITELN('GRADE':11,'CREDIT HOURS':15,'TOTALS':15);
  WRITELN;
END;

PROCEDURE PRINT_BASIC(AREA1 : BAS_SKILL);
CONST
    HRS = 3;
VAR
    I : INTEGER;
    TEMP_SKILLS_TOTAL : INTEGER;
    SEM_TAKEN : STR9;
    TEMP_CLASS : COURSE_REC;
BEGIN
  I := 1;
  TEMP_SKILLS_TOTAL := AREA1.SKILLS_TOTAL;
  IF AREA1.SKILLS_TOT > 0 THEN
    BEGIN
      WRITELN('BASIC SKILLS: REQUIRED - 6 HOURS');
      WHILE TEMP_SKILLS_TOTAL > 0 DO
        BEGIN
          IF TEMP_SKILLS_TOTAL = 3 THEN
            TEMP_CLASS:= AREA1.CLASS1
          ELSE
```

```
CONST
    HRS = 3;
VAR
    I : INTEGER;
    TEMP_SKILLS_TOTAL : INTEGER;
    SEM_TAKEN : STR9;
    TEMP_CLASS : COURSE_REC;
BEGIN
  I := 1;
  TEMP_SKILLS_TOTAL := AREA1.SKILLS_TOTAL;
  IF AREA1.SKILLS_TOT > 0 THEN
    BEGIN
      WRITELN('BASIC SKILLS: REQUIRED - 6 HOURS');
      WHILE TEMP_SKILLS_TOTAL > 0 DO
        BEGIN
          IF TEMP_SKILLS_TOTAL = 3 THEN
            TEMP_CLASS:= AREA1.CLASS1
          ELSE
            TEMP_CLASS:= AREA1.CLASS2;
          WRITE(TEMP_CLASS.DEPT_CODE:18,' ');
          WRITE(TEMP_CLASS.COURSE_NUM);
          CONVERT_SEMESTER(TEMP_CLASS.SEMESTER,SEM_TAKEN);
          WRITE(SEM_TAKEN:14,TEMP_CLASS.CRSE_GRADE:10);
          WRITELN(TEMP_CLASS.CRSE_HOURS:10);
          I := I + 1;
          TEMP_SKILLS_TOTAL := TEMP_SKILLS_TOTAL - HRS;
        END;
  WRITELN('TOTAL BASIC SKILLS':58,AREA1.SKILLS_TOTAL:18);
  WRITELN;
  END
  ELSE
    BEGIN
      WRITELN('** NO BASIC SKILLS **');
      WRITELN;
    END;
END;

PROCEDURE PRINT_HUMANITIES(AREA_B : HUMAN);
CONST
  HRS = 3;
VAR
  SEM_TAKEN : STR9;
  COUNT : INTEGER;
BEGIN
  IF AREA_B.HUMAN_TOTAL > 0
    THEN
      BEGIN
        WRITELN('HUMANITIES: REQUIRED - 18 HOURS');
        IF AREA_B.ARTS_TOTAL > 0
          THEN
            BEGIN
              WRITE('  (1)FINE ARTS - (3 HOURS)');
              WRITELN('COMPLETED     ':48,HRS);
              WRITE(AREA_B.FINE_ARTS.DEPT_CODE:18,' ',
                    AREA_B.FINE_ARTS.COURSE_NUM);
              CONVERT_SEMESTER(AREA_B.FINE_ARTS.SEMESTER,
                               SEM_TAKEN);
              WRITE('      ',SEM_TAKEN);
              WRITELN(AREA_B.FINE_ARTS.CRSE_GRADE:10,
                      AREA_B.FINE_ARTS.CRSE_HOURS:10);
            END
          ELSE
            WRITELN('** NO FINE ARTS **');
        IF AREA_B.LIT_TOTAL > 0
          THEN
            BEGIN
              WRITE(' (2)LITERATURE - (3 HOURS)');
              WRITELN('COMPLETED     ':47,HRS);
              WRITE(AREA_B.LITERATURE.DEPT_CODE:18,' ',
                    AREA_B.LITERATURE.COURSE_NUM);
              CONVERT_SEMESTER(AREA_B.LITERATURE.SEMESTER,
                               SEM_TAKEN);
              WRITE('      ',SEM_TAKEN);
              WRITELN(AREA_B.LITERATURE.CRSE_GRADE:10,
                      AREA_B.LITERATURE.CRSE_HOURS:10);
            END
          ELSE
            WRITELN('** NO LITERATURE **');
        IF AREA_B.HISTORY_TOTAL > 0
          THEN
            BEGIN
              WRITE(' (3)HISTORY - (3 HOURS)');
              WRITELN('COMPLETED     ':50,HRS);
              WRITE(AREA_B.HISTORY.DEPT_CODE:18,' ',
```

```
      BEGIN
        WRITE(' (2)LITERATURE - (3 HOURS)');
        WRITELN('COMPLETED      ':47,HRS);
        WRITE(AREA_B.LITERATURE.DEPT_CODE:18,' ',
              AREA_B.LITERATURE.COURSE_NUM);
        CONVERT_SEMESTER(AREA_B.LITERATURE.SEMESTER,
                         SEM_TAKEN);
        WRITE('      ',SEM_TAKEN);
        WRITELN(AREA_B.LITERATURE.CRSE_GRADE:10,
                AREA_B.LITERATURE.CRSE_HOURS:10);
      END
    ELSE
      WRITELN('** NO LITERATURE **');
  IF AREA_B.HISTORY_TOTAL > 0
    THEN
      BEGIN
        WRITE(' (3)HISTORY - (3 HOURS)');
        WRITELN('COMPLETED      ':50,HRS);
        WRITE(AREA_B.HISTORY.DEPT_CODE:18,' ',
              AREA_B.HISTORY.COURSE_NUM);
        CONVERT_SEMESTER(AREA_B.HISTORY.SEMESTER,SEM_TAKEN);
        WRITE('      ',SEM_TAKEN);
        WRITELN(AREA_B.HISTORY.CRSE_GRADE:10,
                AREA_B.HISTORY.CRSE_HOURS:10);
      END
    ELSE
      WRITELN('** NO HISTORY **');
  IF AREA_B.PHIL_REL_TOTAL > 0
    THEN
      BEGIN
        WRITE(' (4)PHILOSOPHY AND RELIGION - (3 HOURS)');
        WRITELN('COMPLETED      ':34,HRS);
        WRITE(AREA_B.PHIL_REL.DEPT_CODE:18,' ',
              AREA_B.PHIL_REL.COURSE_NUM);
        CONVERT_SEMESTER(AREA_B.PHIL_REL.SEMESTER,
                         SEM_TAKEN);
        WRITE('      ',SEM_TAKEN);
        WRITELN(AREA_B.PHIL_REL.CRSE_GRADE:10,
                AREA_B.PHIL_REL.CRSE_HOURS:10);
      END
    ELSE
      WRITELN('** NO PHILOSOPHY OF RELIGION **');
  IF AREA_B.ELEC_TOTAL > 0
    THEN
      BEGIN
        WRITE('  (5)HUMANITIES ELECTIVES - ',
              '(6 HOURS)');
        WRITELN(' ':23,'COMPLETED      ',
                AREA_B.ELEC_TOTAL);
        IF AREA_B.HUM_ELECTIVES.AREA1_TOT > 0 THEN
          BEGIN
            WITH AREA_B.HUM_ELECTIVES.AREA1 DO
              BEGIN
              WRITE('       (A)FINE ARTS',' ':42);
              WRITELN('COMPLETED      ',
                      AREA_B.HUM_ELECTIVES.AREA1_TOT);
              WRITE(' ':15,DEPT_CODE,' ');
              WRITE(COURSE_NUM,' ':5);
              CONVERT_SEMESTER(AREA_B.HUM_ELECTIVES.AREA1.SEMESTER,
                  SEM_TAKEN);
              WRITE(SEM_TAKEN,' ':9,CRSE_GRADE);
              END;
          END;
        IF AREA_B.HUM_ELECTIVES.AREA2_TOT > 0 THEN
          BEGIN
            WRITE('       (B)LITERATURE',' ':41);
            WRITELN(COMPLETED      ',
                    AREA_B.HUM_ELECTIVES.AREA2_TOT);
            IF AREA_B.HUM_ELECTIVES.AREA2_TOT = 3 THEN
              BEGIN
                WITH AREA_B.HUM_ELECTIVES.AREA2 DO
                BEGIN
                 WRITELN('COMPLETED      ',
                       AREA_B.HUM_ELECTIVES.AREA1_TOT);
                 WRITE(' ':15,DEPT_CODE,' ');
                 WRITE(COURSE_NUM,' ':5);
                 CONVERT_SEMESTER(AREA_B.HUM_ELECTIVES.AREA1.SEMESTER,
                    SEM_TAKEN);
                 WRITE(SEM_TAKEN,' ':9,CRSE_GRADE);
                END;
              END
            ELSE
            BEGIN
              WITH AREA_B.HUM_ELECTIVES.AREA2 DO
```

```
      IF AREA_B.HUM_ELECTIVES.AREA2_TOT = 3 THEN
        BEGIN
          WITH AREA_B.HUM_ELECTIVES.AREA2 DO
          BEGIN
           WRITELN('COMPLETED      ',
                AREA_B.HUM_ELECTIVES.AREA1_TOT);
           WRITE(' ':15,DEPT_CODE,' ');
           WRITE(COURSE_NUM,' ':5);
           CONVERT_SEMESTER(AREA_B.HUM_ELECTIVES.AREA1.SEMESTER,
              SEM_TAKEN);
           WRITE(SEM_TAKEN,' ':9,CRSE_GRADE);
          END;
        END
      ELSE
      BEGIN
        WITH AREA_B.HUM_ELECTIVES.AREA2 DO
        BEGIN
        COUNT:= 1;
        WHILE COUNT <= 2 DO
          BEGIN
            WRITELN('COMPLETED      ',
                 AREA_B.HUM_ELECTIVES.AREA1_TOT);
            WRITE(' ':15,DEPT_CODE);
            WRITE(COURSE_NUM,' ':5);
            CONVERT_SEMESTER(AREA_B.HUM_ELECTIVES.AREA1.SEMESTER,
               SEM_TAKEN);
            WRITE(SEM_TAKEN,' ':9,CRSE_GRADE);
            COUNT:= COUNT + 1;
          END;
        END;
      END;
    END;
  IF AREA_B.HUM_ELECTIVES.AREA3_TOT > 0 THEN
    BEGIN
      WRITE('       (C)HISTORY',' ':44);
      WRITELN('COMPLETED      ',
        AREA_B.HUM_ELECTIVES.AREA3_TOT);
      WITH AREA_B.HUM_ELECTIVES.AREA3 DO
      BEGIN
      WRITELN('COMPLETED      ',
       AREA_B.HUM_ELECTIVES.AREA1_TOT);
      WRITE(' ':15,DEPT_CODE);
      WRITE(COURSE_NUM,' ':5);
      CONVERT_SEMESTER(AREA_B.HUM_ELECTIVES.AREA1.SEMESTER,
           SEM_TAKEN);
      WRITE(SEM_TAKEN,' ':9,CRSE_GRADE);
      END;
    END;
  IF AREA_B.HUM_ELECTIVES.AREA4_TOT > 0 THEN
    BEGIN
      WRITE('       (D)PHILOSOPHY AND RELIGION',
         ' ':28);
      WRITELN('COMPLETED      ',AREA_B.HUM_ELECTIVES.AREA4_TOT);
      WITH AREA_B.HUM_ELECTIVES.AREA4 DO
      BEGIN
      WRITELN('COMPLETED      ',AREA_B.HUM_ELECTIVES.AREA1_TOT);
      WRITE(' ':15,DEPT_CODE,' ');
      WRITE(COURSE_NUM,' ':5);
      CONVERT_SEMESTER(AREA_B.HUM_ELECTIVES.AREA1.SEMESTER,
          SEM_TAKEN);
      WRITE(SEM_TAKEN,' ':9,CRSE_GRADE);
      END;
    END;
  IF AREA_B.HUM_ELECTIVES.AREA5_TOT > 0 THEN
    BEGIN
      WRITE('       (E)CHANCELLOR"S SCHOLARS',
         ' ':30);
      WRITELN('COMPLETED      ',AREA_B.HUM_ELECTIVES.AREA5_TOT);
      IF AREA_B.HUM_ELECTIVES.AREA5_TOT = 3 THEN
        BEGIN
          WITH AREA_B.HUM_ELECTIVES.AREA5 DO
          BEGIN
          WRITELN('COMPLETED      ',AREA_B.HUM_ELECTIVES.AREA1_TOT);
          WRITE(' ':15,DEPT_CODE,' ');
          WRITE(COURSE_NUM,' ':5);
          CONVERT_SEMESTER(AREA_B.HUM_ELECTIVES.AREA1.SEMESTER,
            SEM_TAKEN);
          WRITE(SEM_TAKEN,' ':9,CRSE_GRADE);
          END;
        END
      ELSE
        BEGIN
        COUNT:= COUNT + 1;
        WHILE COUNT <= 2 DO
```

```
                    WRITE('      (E)CHANCELLOR"S SCHOLARS',
                      ' ':30);
                    WRITELN('COMPLETED      ',AREA_B.HUM_ELECTIVES.AREA5_TOT);
                    IF AREA_B.HUM_ELECTIVES.AREA5_TOT = 3 THEN
                      BEGIN
                        WITH AREA_B.HUM_ELECTIVES.AREA5 DO
                        BEGIN
                        WRITELN('COMPLETED      ',AREA_B.HUM_ELECTIVES.AREA1_TOT);
                        WRITE(' ':15,DEPT_CODE,' ');
                        WRITE(COURSE_NUM,' ':5);
                        CONVERT_SEMESTER(AREA_B.HUM_ELECTIVES.AREA1.SEMESTER,
                          SEM_TAKEN);
                        WRITE(SEM_TAKEN,' ':9,CRSE_GRADE);
                        END;
                      END
                    ELSE
                      BEGIN
                      COUNT:= COUNT + 1;
                      WHILE COUNT <= 2 DO
                        BEGIN
                         WITH AREA_B.HUM_ELECTIVES.AREA5 DO
                         BEGIN
                         WRITELN('COMPLETED      ',
                            AREA_B.HUM_ELECTIVES.AREA1_TOT);
                         WRITE(' ':15,DEPT_CODE,' ');
                         WRITE(COURSE_NUM,' ':5);
                         CONVERT_SEMESTER(AREA_B.HUM_ELECTIVES.AREA1.SEMESTER,
                            SEM_TAKEN);
                         WRITE(SEM_TAKEN,' ':9,CRSE_GRADE);
                         COUNT:= COUNT + 1;
                         END;
                        END;
                       END;
                  IF AREA_B.ELEC_TOTAL = 3 THEN
                    WRITELN('ONE HUMANITIES ELECTIVE IS NEEDED');
                END;
             END
            ELSE
              WRITELN('** NO HUMANITIES ELECTIVES **');
            WRITELN('TOTAL HUMANITIES':56,AREA_B.HUMAN_TOTAL:20);
            WRITELN;
          END
          ELSE
            WRITELN('** NO HUMANITIES **');
END;



PROCEDURE PRINT_SOC_SCI(AREA3 : SOCIAL);
CONST
  HRS = 03;
VAR
  CONTINUE : INTEGER;
  SEM_TAKEN : STR9;
BEGIN
  CONTINUE := 1;
  IF AREA3.SOC_SCI_TOTAL > 0
    THEN
      BEGIN
        WRITELN('SOCIAL SCIENCE: REQUIRED - 12 HOURS);
        IF (AREA3.ECON_TOTAL > 0) AND (CONTINUE <= 4 )
          THEN
            BEGIN
              WRITE('  (',CONTINUE,')ECONOMICS - (3 HOURS)');
              WRITELN('COMPLETED      ':48,HRS);
              WRITE(AREA3.ECONOMICS.DEPT_CODE:18,' ',
                    AREA3.ECONOMICS.COURSE_NUM);
              CONVERT_SEMESTER(AREA3.ECONOMICS.SEMESTER,
                               SEM_TAKEN);
              WRITE('      ',SEM_TAKEN);
              WRITELN(AREA3.ECONOMICS.CRSE_GRADE:10,
                      AREA3.ECONOMICS.CRSE_HOURS:10);
              CONTINUE := CONTINUE + 1;
            END
          ELSE
            WRITELN('** NO ECONOMICS **');
        IF (AREA3.GEOGR_TOTAL > 0) AND (CONTINUE <= 4 )
          THEN
            BEGIN
              WRITE('  (',CONTINUE,')GEOGRAPHY - (3 HOURS)');
              WRITELN('COMPLETED      ':48,HRS);
              WRITE(AREA3.GEOGRAPHY.DEPT_CODE:18,' ',
```

```
          WRITE(AREA3.ECONOMICS.DEPT_CODE:18,' ',
                AREA3.ECONOMICS.COURSE_NUM);
          CONVERT_SEMESTER(AREA3.ECONOMICS.SEMESTER,
                           SEM_TAKEN);
          WRITE('      ',SEM_TAKEN);
          WRITELN(AREA3.ECONOMICS.CRSE_GRADE:10,
                  AREA3.ECONOMICS.CRSE_HOURS:10);
          CONTINUE := CONTINUE + 1;
        END
      ELSE
        WRITELN('** NO ECONOMICS **');
    IF (AREA3.GEOGR_TOTAL > 0) AND (CONTINUE <= 4 )
      THEN
        BEGIN
          WRITE('  (',CONTINUE,')GEOGRAPHY - (3 HOURS)');
          WRITELN('COMPLETED      ':48,HRS);
          WRITE(AREA3.GEOGRAPHY.DEPT_CODE:18,' ',
                AREA3.GEOGRAPHY.COURSE_NUM);
          CONVERT_SEMESTER(AREA3.GEOGRAPHY.SEMESTER,
                           SEM_TAKEN);
          WRITE('      ',SEM_TAKEN);
          WRITELN(AREA3.GEOGRAPHY.CRSE_GRADE:10,
                  AREA3.GEOGRAPHY.CRSE_HOURS:10);
          CONTINUE := CONTINUE + 1;
        END
      ELSE
        WRITELN('** NO GEOGRAPHY **');
    IF (AREA3.POL_SCI_TOTAL > 0) AND (CONTINUE <= 4 )
      THEN
        BEGIN
          WRITE('  (',CONTINUE,')PHYSICAL SCIENCE - ',
                '(3 HOURS)');
          WRITELN('COMPLETED      ':41,HRS);
          WRITE(AREA3.POLITICAL_SCIENCE.DEPT_CODE:18,' ',
                AREA3.POLITICAL_SCIENCE.COURSE_NUM);
          CONVERT_SEMESTER(AREA3.POLITICAL_SCIENCE.SEMESTER,
                           SEM_TAKEN);
          WRITE('      ',SEM_TAKEN);
          WRITELN(AREA3.POLITICAL_SCIENCE.CRSE_GRADE:10,
                  AREA3.POLITICAL_SCIENCE.CRSE_HOURS:10);
          CONTINUE := CONTINUE + 1;
        END
      ELSE
        WRITELN('** NO PHYSICAL SCIENCE **');
    IF (AREA3.PSY_TOTAL > 0) AND (CONTINUE <= 4 )
      THEN
        BEGIN
          WRITE('  (',CONTINUE,')PSYCHOLOGY - (3 HOURS)');
          WRITELN('COMPLETED      ':47,HRS);
          WRITE(AREA3.PSYCHOLOGY.DEPT_CODE:18,' ',
                AREA3.PSYCHOLOGY.COURSE_NUM);
          CONVERT_SEMESTER(AREA3.PSYCHOLOGY.SEMESTER,
                           SEM_TAKEN);
          WRITE('      ',SEM_TAKEN);
          WRITELN(AREA3.PSYCHOLOGY.CRSE_GRADE:10,
                  AREA3.PSYCHOLOGY.CRSE_HOURS:10);
          CONTINUE := CONTINUE + 1;
        END
      ELSE
        WRITELN('** NO PSYCHOLOGY **');
    IF (AREA3.SOC_TOTAL > 0) AND (CONTINUE <= 4 )
      THEN
        BEGIN
          WRITE('  (',CONTINUE,')SOCIOLOGY - (3 HOURS)');
          WRITELN('COMPLETED      ':43,HRS);
          WRITE(AREA3.SOCIOLOGY.DEPT_CODE:18,' ',
                AREA3.SOCIOLOGY.COURSE_NUM);
          CONVERT_SEMESTER(AREA3.SOCIOLOGY.SEMESTER,
                           SEM_TAKEN);
          WRITE('      ',SEM_TAKEN);
          WRITELN(AREA3.SOCIOLOGY.CRSE_GRADE:10,
                  AREA3.SOCIOLOGY.CRSE_HOURS:10);
          CONTINUE := CONTINUE + 1;
        END
      ELSE
        WRITELN('** NO SOCIOLOGY **');
    WRITELN('TOTAL SOCIAL SCIENCE':60,AREA3.SOC_SCI_TOTAL:16);
    WRITELN;
  END
ELSE
  BEGIN
    WRITELN('** NO SOCIAL SCIENCES **');
    WRITLEN;
  END;
```

```
            WRITE(AREA3.SOCIOLOGY.DEPT_CODE:18,' ',
                  AREA3.SOCIOLOGY.COURSE_NUM);
            CONVERT_SEMESTER(AREA3.SOCIOLOGY.SEMESTER,
                             SEM_TAKEN);
            WRITE('      ',SEM_TAKEN);
            WRITELN(AREA3.SOCIOLOGY.CRSE_GRADE:10,
                    AREA3.SOCIOLOGY.CRSE_HOURS:10);
            CONTINUE := CONTINUE + 1;
          END
        ELSE
          WRITELN('** NO SOCIOLOGY **');
      WRITELN('TOTAL SOCIAL SCIENCE':60,AREA3.SOC_SCI_TOTAL:16);
      WRITELN;
    END
  ELSE
    BEGIN
      WRITELN('** NO SOCIAL SCIENCES **');
      WRITLEN;
    END;
END;

PROCEDURE PRINT_SCI_MATH(AREA4:SCIENCE);
CONST
  HRS = 03;
VAR
  SEM_TAKEN : STR9;
BEGIN
  IF AREA4.SCI_MATH_TOTAL > 0
    THEN
      BEGIN
        WRITELN('NATURAL SCIENCES AND MATHEMATICS: REQUIRED - ',
                '12 HOURS');
        IF (AREA4.BIO_TOTAL > 0)
          THEN
            BEGIN
              WRITE('  (1)BIOLOGY - (3 HOURS)');
              WRITELN('COMPLETED      ':36,HRS);
              WRITE(AREA4.BIOLOGY.DEPT_CODE:18,' ',
                    AREA4.BIOLOGY.COURSE_NUM);
              CONVERT_SEMESTER(AREA4.BIOLOGY.SEMESTER,SEM_TAKEN);
              WRITE('      ',SEM_TAKEN);
              WRITELN(AREA4.BIOLOGY.CRSE_GRADE:10,
                      AREA4.BIOLOGY.CRSE_HOURS:10);
            END
          ELSE
            WRITELN('** NO BIOLOGY **');
        IF (AREA4.PHY_SCI_TOTAL > 0)
          THEN
            BEGIN
              WRITE('  (2)PHYSICAL SCIENCE - (3 HOURS)');
              WRITELN('COMPLETED      ':27,HRS);
              WRITE(AREA4.PHYSICAL_SCIENCE.DEPT_CODE:18,' ',
                    AREA4.PHYSICAL_SCIENCE.COURSE_NUM);
              CONVERT_SEMESTER(AREA4.PHYSICAL_SCIENCE.SEMESTER,
                               SEM_TAKEN);
              WRITE('      ',SEM_TAKEN);
              WRITELN(AREA4.PHYSICAL_SCIENCE.CRSE_GRADE:10,
                      AREA4.PHYSICAL_SCIENCE.CRSE_HOURS:10);
            END
          ELSE
            WRITELN('** NO PYSICAL SCIENCE **');
        IF (AREA4.MATH_TOTAL > 0)
          THEN
            BEGIN
              WRITE('  (3)MATHEMATICS - (3 HOURS)');
              WRITELN('COMPLETED      ':32,HRS);
              WRITE(AREA4.MATH.DEPT_CODE:18,' ',
                    AREA4.MATH.COURSE_NUM);
              CONVERT_SEMESTER(AREA4.MATH.SEMESTER,SEM_TAKEN);
              WRITE('      ',SEM_TAKEN);
              WRITELN(AREA4.MATH.CRSE_GRADE:10,
                      AREA4.MATH.CRSE_HOURS:10);
            END
          ELSE
            WRITELN('** NO MATHEMATICS **');
        IF (AREA4.ELEC_TOTAL > 0)
          THEN
            BEGIN
              WRITE('  (4)DIVISIONAL ELECTIVES - (3 HOURS)');
              WRITELN('COMPLETED      ':23,HRS);
              WRITE(AREA4.MATH_ELECTIVES.DEPT_CODE:18,' ',
                    AREA4.MATH_ELECTIVES.COURSE_NUM);
              CONVERT_SEMESTER(AREA4.MATH_ELECTIVES.SEMESTER,
                               SEM_TAKEN);
```

```
                    AREA4.MATH.COURSE_NUM);
                CONVERT_SEMESTER(AREA4.MATH.SEMESTER,SEM_TAKEN);
                WRITE('      ',SEM_TAKEN);
                WRITELN(AREA4.MATH.CRSE_GRADE:10,
                        AREA4.MATH.CRSE_HOURS:10);
              END
            ELSE
              WRITELN('** NO MATHEMATICS **');
          IF (AREA4.ELEC_TOTAL > 0)
            THEN
              BEGIN
                WRITE('  (4)DIVISIONAL ELECTIVES - (3 HOURS)');
                WRITELN('COMPLETED     ':23,HRS);
                WRITE(AREA4.MATH_ELECTIVES.DEPT_CODE:18,' ',
                      AREA4.MATH_ELECTIVES.COURSE_NUM);
                CONVERT_SEMESTER(AREA4.MATH_ELECTIVES.SEMESTER,
                                 SEM_TAKEN);
                WRITE('      ',SEM_TAKEN);
                WRITELN(AREA4.MATH_ELECTIVES.CRSE_GRADE:10,
                        AREA4.MATH.CRSE_HOURS:10);
              END
            ELSE
              WRITELN('** NO DIVISIONAL ELECTIVES **');
          WRITELN('TOTAL SCIENCE AND MATHEMATICES':69,
                  AREA4.SCI_MATH_TOTAL:7);
          WRITELN;
        END
      ELSE
        BEGIN
          WRITELN('** NO NATURAL SCIENCE OR MATHEMATICS **');
          WRITELN;
        END;
END;

PROCEDURE PRINT_PE(AREA5 : PHYSICAL):
CONST
  HRS = 1;
VAR
  I : INTEGER;
  TEMP_PE_TOTAL : INTEGER;
  SEM_TAKEN : STR9;
BEGIN
  I := 1;
  TEMP_PE_TOTAL := AREA5.P_E_TOTAL;
  IF AREA5.P_E_TOTAL > 0
  THEN
    BEGIN
      WRITELN('PHYSICAL EDUCATION: REQUIRED - 2 HOURS');
      WHILE TEMP_PE_TOTAL > 0 DO
        BEGIN
          WRITE(AREA5.PE[I].DEPT_CODE:18,' ');
          WRITE(AREA5.PE[I].COURSE_NUM);
          CONVERT_SEMESTER(AREA5.PE[I].SEMESTER,SEM_TAKEN);
          WRITE(SEM_TAKEN:14,AREA5.PE[I].CRSE_GRADE:10);
          WRITELN(AREA5.PE[I].CRSE_HOURS:10);
          I := I + 1;
          TEMP_PE_TOTAL := TEMP_PE_TOTAL - HRS;
        END;
      WRITELN('TOTAL PHYSICAL EDUCATION':64,AREA5.P_E_TOTAL:12);
      WRITELN;
    END
  ELSE
    BEGIN
      WRITELN('** NO PHYSICAL EDUCATION **');
      WRITELN;
    END;
END;

PROCEDURE PRINT_TOTALS(STUDENT_DATA : STUDENT_INFO;
                       GRAND_TOTAL : INTEGER;
                       TRANSFER : BOOLEAN);

BEGIN
  IF TRANSFER
    THEN
      BEGIN
        WRITE('TOTAL BASIC STUDIES HOURS: ':27,GRAND_TOTAL:3);
        WRITELN('TRANSFER HOURS PASSED: ':40,
                STUDENT_DATA.TRAN_PASS);
      END
    ELSE
      WRITELN('TOTAL BASIC STUDIES HOURS:':27,GRAND_TOTAL:3);
  WRITELN;
```

```
END;

PROCEDURE PRINT_TOTALS(STUDENT_DATA : STUDENT_INFO;
                       GRAND_TOTAL : INTEGER;
                       TRANSFER : BOOLEAN);

BEGIN
  IF TRANSFER
    THEN
      BEGIN
        WRITE('TOTAL BASIC STUDIES HOURS: ':27,GRAND_TOTAL:3);
        WRITELN('TRANSFER HOURS PASSED: ':40,
                STUDENT_DATA.TRAN_PASS);
      END
    ELSE
      WRITELN('TOTAL BASIC STUDIES HOURS:':27,GRAND_TOTAL:3);
  WRITELN;
  WRITE('CUMULATIVE HOURS ATTEMPTED:  ':29,STUDENT_DATA.CUM_ATMP);
  WRITELN('CUMULATIVE GPA: ':31,STUDENT_DATA.CUM_GPA);
  WRITELN('CUMULATIVE HOURS PASSED: ':24,'    ',
          STUDENT_DATA.CUM_PASS);
END;

PROCEDURE PRINT_MAJOR(MAT_REP : MATH_RECORD;
                      CSC_REP : CSC_RECORD;
                      MAJOR_TOT : INTEGER;
                      MAJOR_COUNT : INTEGER);

PROCEDURE PRINT_MATH (MAT_REP   : MATH_RECORD;
                      MAJOR_TOT : INTEGER);
CONST
  ELEC_MAX = 12;
VAR
  COUNTER : INTEGER;
BEGIN
 IF MAJOR_TOT <> 0 THEN
 BEGIN
  IF (MAT_REP.REQ_107[1].SSAN = '         ') AND
     (MAT_REP.REQ_107[2].SSAN = '         ') THEN
    BEGIN
      WRITELN('MAT 107 AND MAT 108 OR MAT 109 HAS NOT BEEN FULFILLED.');
      WRITELN('CHECK BASIC STUDIES EVALUATION.');
    END
  IF MAT_REP.REQ_107[1].SSAN <> '         ' THEN
     BEGIN
       WITH MAT_REP.REQ_107[1] DO
       BEGIN
       WRITE(DEPT,'   ',COURSE,'   ',HOUR,'   ');
       WRITE(GRADE,'   ');
       CONVERT_SEMESTER(MAT_REP.REQ_107[1].SEMESTER,SEM_TAKEN);
       WRITELN(SEM_TAKEN);
       END;
     END;
   IF MAT_REP.REQ_107[2].SSAN <> '         ' THEN
    BEGIN
      WITH MAT_REP.REQ_107 DO
      BEGIN
      WRITE(DEPT,'   ',COURSE,'   ',HOUR,'   ');
      WRITE(GRADE,'   ');
      CONVERT_SEMESTER(MAT_REP.REQ_107[2].SEMESTER,SEM_TAKEN);
      WRITELN(SEM_TAKEN);
      END;
    END;
 IF MAT_REP.REQ_220.SSAN <> '         ' THEN
   BEGIN
     WITH MAT_REP.REQ_220 DO
     BEGIN
      WRITE(DEPT,'   ',COURSE,'   ',HOUR,'   ');
      WRITE(GRADE,'   ');
      CONVERT_SEMESTER(MAT_REP.REQ_220.SEMESTER,SEM_TAKEN);
      WRITELN(SEM_TAKEN);
     END
   ELSE
    WRITELN('MATH 220 HAS NOT BEEN FULFILLED.');
 IF MAT_REP.REQ_221.SSAN <> '         ' THEN
   BEGIN
     WITH MAT_REP.REQ_221 DO
     BEGIN
      WRITE(DEPT,'   ',COURSE,'   ',HOUR,'   ');
      WRITE(GRADE,'   ');
      CONVERT_SEMESTER(MAT_REP.REQ_221.SEMESTER,SEM_TAKEN);
      WRITELN(SEM_TAKEN);
     END
```

```
HUMAN          = RECORD
     BEGIN
      WRITE(DEPT,'   ',COURSE,'    ',HOUR,'    ');
      WRITE(GRADE,'    ');
      CONVERT_SEMESTER(MAT_REP.REQ_220.SEMESTER,SEM_TAKEN);
      WRITELN(SEM_TAKEN);
     END
  ELSE
   WRITELN('MATH 220 HAS NOT BEEN FULFILLED.');
IF MAT_REP.REQ_221.SSAN <> '          ' THEN
  BEGIN
    WITH MAT_REP.REQ_221 DO
    BEGIN
     WRITE(DEPT,'   ',COURSE,'    ',HOUR,'    ');
     WRITE(GRADE,'    ');
     CONVERT_SEMESTER(MAT_REP.REQ_221.SEMESTER,SEM_TAKEN);
     WRITELN(SEM_TAKEN);
    END
  ELSE
   WRITELN('MATH 221 HAS NOT BEEN FULFILLED.');
IF MAT_REP.REQ_222.SSAN <> '          ' THEN
  BEGIN
    WITH MAT_REP.REQ_222 DO
    BEGIN
     WRITE(DEPT,'   ',COURSE,'    ',HOUR,'    ');
     WRITE(GRADE,'    ');
     CONVERT_SEMESTER(MAT_REP.REQ_222.SEMESTER,SEM_TAKEN);
     WRITELN(SEM_TAKEN);
    END
  ELSE
   WRITELN('MATH 222 HAS NOT BEEN FULFILLED.');
IF MAT_REP.REQ_315.SSAN <> '          ' THEN
  BEGIN
    WITH MAT_REP.REQ_315 DO
    BEGIN
     WRITE(DEPT,'   ',COURSE,'    ',HOUR,'    ');
     WRITE(GRADE,'    ');
     CONVERT_SEMESTER(MAT_REP.REQ_315.SEMESTER,SEM_TAKEN);
     WRITELN(SEM_TAKEN);
    END
  ELSE
   WRITELN('MATH 315 HAS NOT BEEN FULFILLED.');
IF MAT_REP.REQ_316.SSAN <> '          ' THEN
  BEGIN
    WITH MAT_REP.REQ_316 DO
    BEGIN
     WRITE(DEPT,'   ',COURSE,'    ',HOUR,'    ');
     WRITE(GRADE,'    ');
     CONVERT_SEMESTER(MAT_REP.REQ_316.SEMESTER,SEM_TAKEN);
     WRITELN(SEM_TAKEN);
    END
  ELSE
   WRITELN('MATH 316 HAS NOT BEEN FULFILLED.');
IF MAT_REP.REQ_325.SSAN <> '          ' THEN
  BEGIN
    WITH MAT_REP.REQ_325 DO
    BEGIN
     WRITE(DEPT,'   ',COURSE,'    ',HOUR,'    ');
     WRITE(GRADE,'    ');
     CONVERT_SEMESTER(MAT_REP.REQ_325.SEMESTER,SEM_TAKEN);
     WRITELN(SEM_TAKEN);
    END
  ELSE
   WRITELN('MATH 325 HAS NOT BEEN FULFILLED.');
IF MAT_REP.REQ_431.SSAN <> '          ' THEN
  BEGIN
    WITH MAT_REP.REQ_431 DO
    BEGIN
     WRITE(DEPT,'   ',COURSE,'    ',HOUR,'    ');
     WRITE(GRADE,'    ');
     CONVERT_SEMESTER(MAT_REP.REQ_431.SEMESTER,SEM_TAKEN);
     WRITELN(SEM_TAKEN);
    END
  ELSE
   WRITELN('MATH 431 HAS NOT BEEN FULFILLED.');
IF MAT_REP.ELEC_TOTAL > 0 THEN
  BEGIN
    FOR COUNTER := 1 TO MAT_REP.REQ_COUNT DO
      BEGIN
        WITH MAT_REP.REQ_EX[COUNTER] DO
        BEGIN
        WRITE(DEPT,'   'COURSE,'    ',HOUR,'    ');
        WRITE(GRADE,'    ');
        CONVERT_SEMESTER(MAT_REP.REQ_EX[COUNTER].SEMESTER,
```

```
      WITH MAT_REP.REQ_431 DO
      BEGIN
       WRITE(DEPT,'   ',COURSE,'    ',HOUR,'    ');
       WRITE(GRADE,'    ');
       CONVERT_SEMESTER(MAT_REP.REQ_431.SEMESTER,SEM_TAKEN);
       WRITELN(SEM_TAKEN);
      END
    ELSE
     WRITELN('MATH 431 HAS NOT BEEN FULFILLED.');
  IF MAT_REP.ELEC_TOTAL > 0 THEN
    BEGIN
      FOR COUNTER := 1 TO MAT_REP.REQ_COUNT DO
        BEGIN
          WITH MAT_REP.REQ_EX[COUNTER] DO
          BEGIN
          WRITE(DEPT,'   'COURSE,'    ',HOUR,'    ');
          WRITE(GRADE,'    ');
          CONVERT_SEMESTER(MAT_REP.REQ_EX[COUNTER].SEMESTER,
                           SEM_TAKEN);
          WRITELN(SEM_TAKEN);
          END;
        END;
    IF MAT_REP.ELEC_TOTAL < ELEC_MAX THEN
      WRITELN('12 HOURS ARE NEEDED TO FULFILL THIS REQUIREMENT.');
    END
    ELSE
      WRITELN('12 HOURS OF ELECTIVES HAVE NOT BEEN FULFILLED.');
 END
 ELSE
   WRITELN('NO MAJOR CLASSES HAVE BEEN FULFILLED.');

PROCEDURE PRINT_CSC(CSC_REP : CSC_RECORD;
                    MAJOR_TOT : INTEGER);

BEGIN
  WRITELN('MAJOR CLASSES FULFILLED');
  IF MAJOR_TOT > 0 THEN
  BEGIN
  IF (CSC_REP.REQ_107[1].SSAN = '          ') AND
     (CSC_REP.REQ_107[2].SSAN = '          ') THEN
    BEGIN
     WRITE('MATH 107 AND MATH 108 OR MATH 109 HAS NOT BEEN FULFILLED.');
     WRITE('CHECK BASIC STUDIES EVALUATION.');
    END
  ELSE
  BEGIN
    IF CSC_REP.REQ_107[1].SSAN <>'          ' THEN
      BEGIN
       WITH CSC_REP.REQ_107[1] DO
       BEGIN
         WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
         WRITE(GRADE,'    ');
         CONVERT_SEMESTER(CSC_REP.REQ_107[1].SEMESTER,SEM_TAKEN);
         WRITELN(SEM_TAKEN);
       END;
      END;
    IF CSC_REP.REQ_107[2].SSAN <> '          ' THEN
    BEGIN
      WITH CSC_REP.REQ_107[2] DO
       BEGIN
         WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
         WRITE(GRADE,'    ');
         CONVERT_SEMESTER(CSC_REP.REQ_107[2].SEMESTER,SEM_TAKEN);
         WRITELN(SEM_TAKEN);
       END;
    END;
  END;
 IF CSC_REP.REQ_221.SSAN <> '          ' THEN
  BEGIN
    WITH CSC_REP.REQ_221 DO
    BEGIN
    WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
    WRITE(GRADE,'    ');
    CONVERT_SEMESTER(CSC_REP.REQ_221.SEMESTER,SEM_TAKEN);
    WRITELN(SEM_TAKEN);
    END;
  END
 ELSE
   WRITELN('MATH 221 HAS NOT BEEN FULFILLED.');
 IF CSC_REP.REQ_222.SSAN <> '          ' THEN
  BEGIN
    WITH CSC_REP.REQ_222 DO
    BEGIN
    WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
```

```
HUMAN          = RECORD
 BEGIN
   WITH CSC_REP.REQ_221 DO
   BEGIN
   WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
   WRITE(GRADE,'    ');
   CONVERT_SEMESTER(CSC_REP.REQ_221.SEMESTER,SEM_TAKEN);
   WRITELN(SEM_TAKEN);
   END;
 END
ELSE
  WRITELN('MATH 221 HAS NOT BEEN FULFILLED.');
IF CSC_REP.REQ_222.SSAN <> '          ' THEN
 BEGIN
   WITH CSC_REP.REQ_222 DO
   BEGIN
   WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
   WRITE(GRADE,'    ');
   CONVERT_SEMESTER(CSC_REP.REQ_222.SEMESTER,SEM_TAKEN);
   WRITELN(SEM_TAKEN);
   END;
 END
ELSE
  WRITELN('MATH 222 HAS NOT BEEN FULFILLED.');
IF CSC_REP.REQ_315.SSAN <> '          ' THEN
 BEGIN
   WITH CSC_REP.REQ_315 DO
   BEGIN
   WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
   WRITE(GRADE,'    ');
   CONVERT_SEMESTER(CSC_REP.REQ_315.SEMESTER,SEM_TAKEN);
   WRITELN(SEM_TAKEN);
   END;
 END
ELSE
  WRITELN('MATH 315 HAS NOT BEEN FULFILLED.');
IF CSC_REP.REQ_316.SSAN <> '          ' THEN
 BEGIN
   WITH CSC_REP.REQ_316 DO
   BEGIN
   WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
   WRITE(GRADE,'    ');
   CONVERT_SEMESTER(CSC_REP.REQ_316.SEMESTER,SEM_TAKEN);
   WRITELN(SEM_TAKEN);
   END;
 END
ELSE
  WRITELN('MATH 316 HAS NOT BEEN FULFILLED.');
IF CSC_REP.REQ_317.SSAN <> '          ' THEN
 BEGIN
   WITH CSC_REP.REQ_317 DO
   BEGIN
   WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
   WRITE(GRADE,'    ');
   CONVERT_SEMESTER(CSC_REP.REQ_317.SEMESTER,SEM_TAKEN);
   WRITELN(SEM_TAKEN);
   END;
 END
ELSE
  WRITELN('MATH 317 HAS NOT BEEN FULFILLED.');
IF CSC_REP.REQ_155.SSAN <> '          ' THEN
 BEGIN
   WITH CSC_REP.REQ_155 DO
   BEGIN
   WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
   WRITE(GRADE,'    ');
   CONVERT_SEMESTER(CSC_REP.REQ_155.SEMESTER,SEM_TAKEN);
   WRITELN(SEM_TAKEN);
   END;
 END
ELSE
  WRITELN('CSC 155 HAS NOT BEEN FULFILLED.');
IF CSC_REP.REQ_200.SSAN <> '          ' THEN
 BEGIN
   WITH CSC_REP.REQ_200 DO
   BEGIN
   WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
   WRITE(GRADE,'    ');
   CONVERT_SEMESTER(CSC_REP.REQ_200.SEMESTER,SEM_TAKEN);
   WRITELN(SEM_TAKEN);
   END;
 END
ELSE
  WRITELN('CSC 200 HAS NOT BEEN FULFILLED.');
```

```
    CONVERT_SEMESTER(CSC_REP.REQ_155.SEMESTER,SEM_TAKEN);
    WRITELN(SEM_TAKEN);
    END;
  END
ELSE
   WRITELN('CSC 155 HAS NOT BEEN FULFILLED.');
IF CSC_REP.REQ_200.SSAN <> '          ' THEN
  BEGIN
    WITH CSC_REP.REQ_200 DO
    BEGIN
    WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
    WRITE(GRADE,'    ');
    CONVERT_SEMESTER(CSC_REP.REQ_200.SEMESTER,SEM_TAKEN);
    WRITELN(SEM_TAKEN);
    END;
  END
ELSE
   WRITELN('CSC 200 HAS NOT BEEN FULFILLED.');
IF CSC_REP.REQ_250.SSAN<> '          ' THEN
  BEGIN
    WITH CSC_REP.REQ_250 DO
    BEGIN
    WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
    WRITE(GRADE,'    ');
    CONVERT_SEMESTER(CSC_REP.REQ_250.SEMESTER,SEM_TAKEN);
    WRITELN(SEM_TAKEN);
    END;
  END
ELSE
   WRITELN('CSC 250 HAS NOT BEEN FULFILLED.');
IF CSC_REP.REQ_270.SSAN <> '          ' THEN
  BEGIN
    WITH CSC_REP.REQ_270 DO
    BEGIN
    WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
    WRITE(GRADE,'    ');
    CONVERT_SEMESTER(CSC_REP.REQ_270.SEMESTER,SEM_TAKEN);
    WRITELN(SEM_TAKEN);
    END;
  END
ELSE
   WRITELN('CSC 270 HAS NOT BEEN FULFILLED.');
IF CSC_REP.REQ_350.SSAN <> '          ' THEN
  BEGIN
    WITH CSC_REP.REQ_350 DO
    BEGIN
    WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
    WRITE(GRADE,'    ');
    CONVERT_SEMESTER(CSC_REP.REQ_350.SEMESTER,SEM_TAKEN);
    WRITELN(SEM_TAKEN);
    END;
  END
ELSE
   WRITELN('CSC 350 HAS NOT BEEN FULFILLED.');
IF CSC_REP.REQ_420.SSAN <> '          ' THEN
  BEGIN
    WITH CSC_REP.REQ_420 DO
    BEGIN
    WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
    WRITE(GRADE,'    ');
    CONVERT_SEMESTER(CSC_REP.REQ_420.SEMESTER,SEM_TAKEN);
    WRITELN(SEM_TAKEN);
    END;
  END
ELSE
   WRITELN('CSC 420 HAS NOT BEEN FULFILLED.');
IF CSC_REP.REQ_450.SSAN <> '          ' THEN
  BEGIN
    WITH CSC_REP.REQ_450 DO
    BEGIN
    WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
    WRITE(GRADE,'    ');
    CONVERT_SEMESTER(CSC_REP.REQ_450.SEMESTER,SEM_TAKEN);
    WRITELN(SEM_TAKEN);
    END;
  END
ELSE
   WRITELN('CSC 450 HAS NOT BEEN FULFILLED.');
IF CSC_REP.REQ_210.SSAN <> '          ' THEN
  BEGIN
    WITH CSC_REP.REQ_210 DO
    BEGIN
    WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
```

```
    HUMAN          = RECORD
  IF CSC_REP.REQ_450.SSAN <>                    THEN
   BEGIN
     WITH CSC_REP.REQ_450 DO
     BEGIN
     WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
     WRITE(GRADE,'    ');
     CONVERT_SEMESTER(CSC_REP.REQ_450.SEMESTER,SEM_TAKEN);
     WRITELN(SEM_TAKEN);
     END;
   END
  ELSE
    WRITELN('CSC 450 HAS NOT BEEN FULFILLED.');
  IF CSC_REP.REQ_210.SSAN <>          ' THEN
   BEGIN
     WITH CSC_REP.REQ_210 DO
     BEGIN
     WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
     WRITE(GRADE,'    ');
     CONVERT_SEMESTER(CSC_REP.REQ_210.SEMESTER,SEM_TAKEN);
     WRITELN(SEM_TAKEN);
     END;
   END
  ELSE
    WRITELN('CSC 210 HAS NOT BEEN FULFILLED.');
  IF CSC_REP.REQ_370.SSAN <>          ' THEN
   BEGIN
     WITH CSC_REP.REQ_370 DO
     BEGIN
     WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
     WRITE(GRADE,'    ');
     CONVERT_SEMESTER(CSC_REP.REQ_370.SEMESTER,SEM_TAKEN);
     WRITELN(SEM_TAKEN);
     END;
   END
  ELSE
    WRITELN('CSC 370 HAS NOT BEEN FULFILLED.');
  END
  ELSE
    WRITELN('NO MAJOR CLASSES HAVE BEEN FULFILLED.');
END;

PROCEDURE PRINT_OTHER(MAJ_REPORT : MAJ_ARRAY;
                      MAJOR_TOT : INTEGER;
                      MAJOR_COUNT : INTEGER);
VAR
  COUNTER : INTEGER;
BEGIN
  IF MAJOR_TOT > 0 THEN
   BEGIN
     FOR COUNTER := 1 TO MAJOR_COUNT DO
       BEGIN
        WITH MAJ_REPORT[COUNTER] DO
        BEGIN
         WRITE(DEPT,'    ',COURSE,'    ',HOUR,'    ');
         WRITE(GRADE,'    ');
         CONVERT_SEMESTER(MAJ_REPORT[COUNTER].SEMESTER,
                  SEM_TAKEN);
         WRITELN(SEM_TAKEN);
        END;
       END;
   END
   ELSE
     WRITELN('NO MAJOR CLASSES HAVE BEEN FULFILLED.');
END;



BEGIN
  WRITELN('MAJOR CLASSES FULFILLED');
  IF STUDENT_DATA.MAJOR  = 'MAT' THEN
     PRINT_MATH(MAT_REP,MAJOR_TOT);
  ELSE
    IF STUDENT_DATA.MAJOR = 'CSC' THEN
      PRINT_CSC(CSC_REP,MAJOR_TOT)
    ELSE
      PRINT_OTHER(MAJ_REPORT,MAJOR_TOT,MAJOR_COUNT);
  WRITE(NUMBER OF HOURS FILLED FOR MAJOR : ',MAJOR_TOT);
END;

PROCEDURE PRINT_EXTRA(EX_REPORT : EX_ARRAY;
                      EXTRA_TOT : INTEGER;
                      EXTRA_COUNT : INTEGER);
```

```
BEGIN
  WRITELN('MAJOR CLASSES FULFILLED');
  IF STUDENT_DATA.MAJOR  = 'MAT' THEN
     PRINT_MATH(MAT_REP,MAJOR_TOT);
  ELSE
    IF STUDENT_DATA.MAJOR = 'CSC' THEN
      PRINT_CSC(CSC_REP,MAJOR_TOT)
    ELSE
      PRINT_OTHER(MAJ_REPORT,MAJOR_TOT,MAJOR_COUNT);
  WRITE(NUMBER OF HOURS FILLED FOR MAJOR : ',MAJOR_TOT);
END;

PROCEDURE PRINT_EXTRA(EX_REPORT : EX_ARRAY;
                      EXTRA_TOT : INTEGER;
                      EXTRA_COUNT : INTEGER);

VAR
  COUNTER : INTEGER;
BEGIN
  IF EXTRA_TOT > 0 THEN
    BEGIN
      FOR COUNTER := 1 TO EXTRA_COUNT DO
        BEGIN
         WITH EX_REPORT[COUNTER] DO
         BEGIN
           WRITE(DEPT,'   ',COURSE,'   ',HOUR,'   ');
           WRITE(GRADE,'   ');
           CONVERT_SEMESTER(EX_REPORT[COUNTER].SEMESTER,
                            SEM_TAKEN);
           WRITELN(SEM_TAKEN);
         END;
        END;
    END
    ELSE
      WRITELN('THERE ARE NO ELECTIVE CLASSES FULFILLED.');
END;


BEGIN
  TRANSFER := FALSE;
  PRINT_HEADER(STUDENT_DATA);
  PRINT_BASIC(REPORT.BASIC_SKILLS);
  PRINT_HUMANITIES(REPORT.HUMANITIES);
  PRINT_SOC_SCI(REPORT.SOCIAL_SCIENCES);
  PRINT_SCI_MATH(REPORT.SCIENCES_MATHEMATICS);
  PRINT_PE(REPORT.PHYSICAL_EDUCATION);
  PRINT_MAJOR(MAT_REP,CSC_REP,MAJOR_TOT,MAJOR_COUNT);
  PRINT_EXTRA();
  IF TRANSFER_STUDENT(STUDENT_DATA.DISPOSITION)
    THEN
      BEGIN
        WRITELN('** TRANSFER STUDENT **':51);
        TRANSFER := TRUE;
      END;
  PRINT_TOTALS(STUDENT_DATA,GRAND_TOTAL,TRANSFER);  (MODIFY)
END;

PROCEDURE INIT_REPORT(VAR REPORT : STUDENT_REC);
BEGIN
  REPORT.BASIC_SKILLS.SKILLS_TOT := 0;
  REPORT.HUMANITIES.ARTS_TOTAL := 0;
  REPORT.HUMANITIES.LIT_TOTAL := 0;
  REPORT.HUMANITIES.HISTORY_TOTAL := 0;
  REPORT.HUMANITIES.PHIL_REL_TOTAL := 0;
  REPORT.HUMANITIES.HUM_ELECTIVES.AREA1_TOT := 0;
  REPORT.HUMANITIES.HUM_ELECTIVES.AREA2_TOT := 0;
  REPORT.HUMANITIES.HUM_ELECTIVES.AREA3_TOT := 0;
  REPORT.HUMANITIES.HUM_ELECTIVES.AREA4_TOT := 0;
  REPORT.HUMANITIES.HUM_ELECTIVES.AREA5_TOT := 0;
  REPORT.HUMANITIES.ELEC_TOTAL := 0;
  REPORT.HUMANITIES.HUMAN_TOTAL := 0;
  REPORT.SOCIAL_SCIENCES.ECON_TOTAL := 0;
  REPORT.SOCIAL_SCIENCES.GEOGR_TOTAL := 0;
  REPORT.SOCIAL_SCIENCES.POL_SCI_TOTAL := 0;
  REPORT.SOCIAL_SCIENCES.PSY_TOTAL := 0;
  REPORT.SOCIAL_SCIENCES.SOC_TOTAL := 0;
  REPORT.SOCIAL_SCIENCES.SOC_SCI_TOTAL := 0;
  REPORT.SCIENCE_MATHEMATICS.BIO_TOTAL := 0;
  REPORT.SCIENCE_MATHEMATICS.PHY_SCI_TOTAL := 0;
  REPORT.SCIENCE_MATHEMATICS.MATH_TOTAL := 0;
  REPORT.SCIENCE_MATHEMATICS.ELEC_TOTAL := 0;
  REPORT.SCIENCE_MATHEMATICS.SCI_MATH_TOTAL := 0;
```

```
    REPORT.HUMANITIES.HUM_ELECTIVES.AREA3_TOT := 0;
    REPORT.HUMANITIES.HUM_ELECTIVES.AREA4_TOT := 0;
    REPORT.HUMANITIES.HUM_ELECTIVES.AREA5_TOT := 0;
    REPORT.HUMANITIES.ELEC_TOTAL := 0;
    REPORT.HUMANITIES.HUMAN_TOTAL := 0;
    REPORT.SOCIAL_SCIENCES.ECON_TOTAL := 0;
    REPORT.SOCIAL_SCIENCES.GEOGR_TOTAL := 0;
    REPORT.SOCIAL_SCIENCES.POL_SCI_TOTAL := 0;
    REPORT.SOCIAL_SCIENCES.PSY_TOTAL := 0;
    REPORT.SOCIAL_SCIENCES.SOC_TOTAL := 0;
    REPORT.SOCIAL_SCIENCES.SOC_SCI_TOTAL := 0;
    REPORT.SCIENCE_MATHEMATICS.BIO_TOTAL := 0;
    REPORT.SCIENCE_MATHEMATICS.PHY_SCI_TOTAL := 0;
    REPORT.SCIENCE_MATHEMATICS.MATH_TOTAL := 0;
    REPORT.SCIENCE_MATHEMATICS.ELEC_TOTAL := 0;
    REPORT.SCIENCE_MATHEMATICS.SCI_MATH_TOTAL := 0;
    REPORT.PHYSICAL_EDUCATION.P_E_TOTAL := 0;
END;

PROCEDURE NEW_STUDENT(VAR NEW_FILE     : FILE OF CHAR;
                      VAR STUDENT_DATA : STUDENT_INFO);
VAR
    SS : PACKED ARRAY[1..9] OF CHAR;
    NAM : PACKED ARRAY[1..34] OF CHAR;
    ENT : PACKED ARRAY[1..6] OF CHAR;
    CL : PACKED ARRAY[1..1] OF CHAR;
    TC,VET,DIS,MAR : CHAR;
    ADV : PACKED ARAY[1..8] OF CHAR;
    SV,SM,MAJ : PACKED ARRAY[1..3] OF CHAR;
    HSG,DEG : PACKED ARRAY[1..2] OF CHAR;
    CP,CA,TP : PACKED ARRAY[1..5] OF CHAR;
    CG : PACKED ARRAY[1..4] OF CHAR;


BEGIN
    READ(NEWFILE,SS,NAM,ENT,MAR,DIS,VET,ADV,MAJ,DEG);
    READLN(NEWFILE,SV,SM,TC,HSG,CL,TP,CA,CP,CG);
    STUDENT_DATA.ADVISOR:= ADV;
    STUDENT_DATA.CLASS:= CL;
    STUDENT_DATA.CUM_ATMP:= CA;
    STUDENT_DATA.CUM_GPA:= CG;
    STUDENT_DATA.CUM_PASS:= CP;
    STUDENT_DATA.DEGREE:= DEG;
    STUDENT_DATA.DISPOSITION:= DIS;
    STUDENT_DATA.ENTERED:= ENT;
    STUDENT_DATA.HS_GRAD:= HSG;
    STUDENT_DATA.MAJOR:= MAJ;
    STUDENT_DATA.MARRIED:= MAR;
    STUDENT_DATA.NAME:= NAM;
    STUDENT_DATA.SAT_MATH;= SM;
    STUDENT_DATA.SAT_VERB:= SV;
    STUDENT_DATA.SSAN:= SS;
    STUDENT_DATA.TEACHER_CERT:= TC;
    STUDENT_DATA.TRAN_PASS;- TP;
    STUDENT_DATA.VETERAN:= VET;
END;

PROCEDURE NEW_GRADE(VAR GRADES     : FILE OF CHAR;
                    VAR GRADE_DATA : CLASS_REC);
VAR
    RPT : PACKED ARAY [1..1] OF CHAR;
    GRD,HRS : PACKED ARRAY[1..2] OF CHAR;
    MAJ,DEP,CRS,SEM : PACKED ARRAY[1..3] OF CHAR;
    LC1,LC2,LC3,LC4,LC5 : PACKED ARRAY[1..3] OF CHAR;
    ADV : PACKED ARRAY[1..8] OF CHAR;
    SS : PACKED ARRAY[1..9] OF CHAR;
    NAM : PACKED ARRAY[1..34] OF CHAR;

BEGIN
    READ(GRADES,SS,NAM,ADV,MAJ,DEP,CRS,HRS,GRD,SEM,RPT);
    READLN(GRADES,LC1,LC2,LC3,LC4,LC5);
    GRADE_DATA.ADVISOR:= ADV;
    GRADE_DATA.COURSE:= CRS;
    GRADE_DATA.DEPT:= DEP;
    GRADE_DATA.GRADE:= GRD;
    GRADE_DATA.HOURS:= HRS;
    GRADE_DATA.LOC1:= LC1;
    GRADE_DATA.LOC2:= LC2;
    GRADE_DATA.LOC3:= LC3;
    GRADE_DATA.LOC4:= LC4;
    GRADE_DATA.LOC5:= LC5;
    GRADE_DATA.MAJOR:= MAJ;
    GRADE_DATA.NAME:= NAM;
```

```
  ADV : PACKED ARRAY[1..8] OF CHAR;
  SS : PACKED ARRAY[1..9] OF CHAR;
  NAM : PACKED ARRAY[1..34] OF CHAR;

BEGIN
  READ(GRADES,SS,NAM,ADV,MAJ,DEP,CRS,HRS,GRD,SEM,RPT);
  READLN(GRADES,LC1,LC2,LC3,LC4,LC5);
  GRADE_DATA.ADVISOR:= ADV;
  GRADE_DATA.COURSE:= CRS;
  GRADE_DATA.DEPT:= DEP;
  GRADE_DATA.GRADE:= GRD;
  GRADE_DATA.HOURS:= HRS;
  GRADE_DATA.LOC1:= LC1;
  GRADE_DATA.LOC2:= LC2;
  GRADE_DATA.LOC3:= LC3;
  GRADE_DATA.LOC4:= LC4;
  GRADE_DATA.LOC5:= LC5;
  GRADE_DATA.MAJOR:= MAJ;
  GRADE_DATA.NAME:= NAM;
  GRADE_DATA.RPT:= RPT;
  GRADE_DATA.SEMESTER:= SEM;
  GRADE_DATA.SSAN:= SS;
END;

PROCEDURE EVALUATE_MATH(MAJOR_DATA : CL_RECORD;
                        VAR MAJOR_TOT : INTEGER;
                        COUNT : INTEGER;
                        VAR MAT_REP : MATH_RECORD;
                        VAR EXTRA_BASIC : BASIC_ARRAY);

CONST
  ELEC_MAX = 12;
VAR
  TEMP_HRS : INTEGER;

BEGIN
  IF (MAJOR_DATA.COURSE = '107') OR (MAJOR_DATA.COURSE = '108') THEN
   BEGIN
    IF MAT_REP.REQ_107[1].SSAN = ' ' THEN
    BEGIN
       TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
       MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
       MAT_REP.REQ_107[1]:= MAJOR_DATA;
    END
    ELSE
    BEGIN
       TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
       MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
       MAT_REP.REQ_107[2]:= MAJOR_DATA;
    END;
   END
  ELSE
    IF (MAJOR_DATA.COURSE = '109') THEN
      BEGIN
       IF (MAJ_REP.REQ_107[1].SSAN = ' ' THEN
       BEGIN
          TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
          MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
          MAT_REP.REQ_107[1]:= MAJOR_DATA;
       END
       ELSE
       BEGIN
         TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
         MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
         MAT_REP.REQ_107[2]:= MAJOR_DATA;
       END;
      END
    ELSE
      IF MAJOR_DATA.COURSE = '220'  THEN
      BEGIN
        TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
        MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
        MAT_REP.REQ_220:= MAJOR_DATA;
      END
      ELSE
        IF MAJOR_DATA.COURSE = '221' THEN
        BEGIN
          TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
          MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
          MAT_REP.REQ_221:= MAJOR_DATA;
        END
        ELSE
          IF MAJOR_DATA.COURSE = '222' THEN
          BEGIN
```

```
    HUMAN          = RECORD
      IF MAJOR_DATA.COURSE = '220' THEN
      BEGIN
        TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
        MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
        MAT_REP.REQ_220:= MAJOR_DATA;
      END
      ELSE
        IF MAJOR_DATA.COURSE = '221' THEN
        BEGIN
          TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
          MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
          MAT_REP.REQ_221:= MAJOR_DATA;
        END
        ELSE
          IF MAJOR_DATA.COURSE = '222' THEN
          BEGIN
            TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
            MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
            MAT_REP.REQ_222:= MAJOR_DATA;
          END
          ELSE
            IF MAJOR_DATA.COURSE = '315' THEN
            BEGIN
              TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
              MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
              MAT_REP.REQ_315:= MAJOR_DATA;
            END
            ELSE
              IF MAJOR_DATA.COURSE = '316' THEN
              BEGIN
                TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
                MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
                MAT_REP.REQ_316:= MAJOR_DATA;
              END
              ELSE
                IF MAJOR_DATA.COURSE = '325' THEN
                BEGIN
                  TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
                  MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
                  MAT_REP.REQ_325:= MAJOR_DATA;
                END
                ELSE
                  IF MAJOR_DATA.COURSE = '431' THEN
                  BEGIN
                    TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
                    MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
                    MAT_REP.REQ_431:= MAJOR_DATA;
                  END
                  ELSE
                    IF MAT_REP.ELEC_TOTAL < ELEC_MAX THEN
                      IF MAJOR_DATA.COURSE >= '300' THEN
                       BEGIN
                        MAT_REP.REQ_EX[MAT_REP.REQ_COUNT]:=
                           MAJOR_DATA;
                        MAT_REP.REQ_COUNT:= MAT_REP.REQ_COUNT + 1;
                        TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
                        MAT_REP.ELEC_TOTAL:= MAT_REP.ELEC_TOTAL
                          + TEMP_HRS;
                        MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
                       END
                      ELSE
                       EXTRA_BASIC[COUNT]:=MAJOR_DATA;
                    ELSE
                      EXTRA_BASIC[COUNT]:= MAJOR_DATA;
END;

PROCEDURE EVALUATE_CSC(MAJOR_DATA : CL_RECORD;
                       VAR EXTRA_BASIC : BASIC_ARRAY
                       COUNT : INTEGER;
                       VAR CSC_REP : CSC_RECORD;
                       VAR MAJOR_TOT : INTEGER);
VAR
  TEMP_HRS : INTEGER;
BEGIN
 IF MAJOR_DATA.DEPT = 'MAT' THEN
 BEGIN
  IF (MAJOR_DATA.COURSE = '107) OR (MAJOR_DATA.COURSE = '108') THEN
    IF CSC_REP.REQ_107[1].SSAN = ' ' THEN
    BEGIN
      TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
      MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
      CSC_REP.REQ_107[1]:= MAJOR_DATA;
```

```
END;

PROCEDURE EVALUATE_CSC(MAJOR_DATA : CL_RECORD;
                       VAR EXTRA_BASIC : BASIC_ARRAY
                       COUNT : INTEGER;
                       VAR CSC_REP : CSC_RECORD;
                       VAR MAJOR_TOT : INTEGER);
VAR
  TEMP_HRS : INTEGER;
BEGIN
 IF MAJOR_DATA.DEPT = 'MAT' THEN
 BEGIN
  IF (MAJOR_DATA.COURSE = '107) OR (MAJOR_DATA.COURSE = '108') THEN
    IF CSC_REP.REQ_107[1].SSAN = ' ' THEN
    BEGIN
      TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
      MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
      CSC_REP.REQ_107[1]:= MAJOR_DATA;
    END
    ELSE
     IF CSC_REP.REQ_107[2].SSAN = ' ' THEN
     BEGIN
       TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
       MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
       CSC_REP.REQ_107[2]:= MAJOR_DATA;
     END
  ELSE
   IF MAJOR_DATA.COURSE = '109' THEN
      IF CSC_REP.REQ_107[1].SSAN = ' ' THEN
      BEGIN
         TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
         MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
         CSC_REP.REQ_107[1]:= MAJOR_DATA;
       END
       ELSE
        IF CSC_REP.REQ_107[2].SSAN = ' ' THEN
        BEGIN
          TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
          MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
          CSC_REP.REQ_107[2]:= MAJOR_DATA;
        END
        ELSE
          EXTRA_BASIC[COUNT]:= MAJOR_DATA;
   ELSE
     IF MAJOR_DATA.COURSE = '221' THEN
     BEGIN
       TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
       MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
       CSC_REP.REQ_221:= MAJOR_DATA;
     END
     ELSE
       IF MAJOR_DATA.COURSE = '222' THEN
       BEGIN
         TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
         MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
         CSC_REP.REQ_222:= MAJOR_DATA;
       END
       ELSE
         IF MAJOR_DATA.COURSE = '315' THEN
         BEGIN
           TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
           MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
           CSC_REP.REQ_315:= MAJOR_DATA;
         END
         ELSE
           IF MAJOR_DATA.COURSE = '316' THEN
           BEGIN
             TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
             MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
             CSC_REP.REQ_316:= MAJOR_DATA;
           END
           ELSE
             IF MAJOR_DATA.COURSE = '317' THEN
             BEGIN
               TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
               MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
               CSC_REP.REQ_317:= MAJOR_DATA;
             END
             ELSE
               EXTRA_BASIC[COUNT]:= MAJOR_DATA;
 END
 ELSE
  BEGIN
  IF MAJOR_DATA.DEPT = 'CSC' THEN
```

```
ROMAN          = RECORD
            MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
            CSC_REP.REQ_316:= MAJOR_DATA;
          END
          ELSE
            IF MAJOR_DATA.COURSE = '317' THEN
            BEGIN
              TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
              MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
              CSC_REP.REQ_317:= MAJOR_DATA;
            END
            ELSE
              EXTRA_BASIC[COUNT]:= MAJOR_DATA;
END
ELSE
 BEGIN
 IF MAJOR_DATA.DEPT = 'CSC' THEN
   IF MAJOR_DATA.COURSE = '155' THEN
   BEGIN
      TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
      MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
      CSC_REP.REQ_155:= MAJOR_DATA;
   END
   ELSE
     IF MAJOR_DATA.COURSE = '200' THEN
     BEGIN
       TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
       MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
       CSC_REP.REQ_200:= MAJOR_DATA;
     END
     ELSE
      IF MAJOR_DATA.COURSE = '250' THEN
      BEGIN
         TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
         MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
         CSC_REP.REQ_250:= MAJOR_DATA;
      END
      ELSE
       IF MAJOR_DATA.COURSE = '270' THEN
       BEGIN
         TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
         MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
         CSC_REP.REQ_270:= MAJOR_DATA;
       END
       ELSE
        IF MAJOR_DATA.COURSE = '350' THEN
        BEGIN
          TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
          MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
          CSC_REP.REQ_350:= MAJOR_DATA;
        END
        ELSE
         IF MAJOR_DATA.COURSE = '420' THEN
         BEGIN
           TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
           MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
           CSC_REP.REQ_420:= MAJOR_DATA;
         END
         ELSE
          IF MAJOR_DATA.COURSE = '450' THEN
          BEGIN
            TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
            MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
            CSC_REP.REQ_450:= MAJOR_DATA;
          END
          ELSE
           IF MAJOR_DATA.COURSE = '210' THEN
           BEGIN
            TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
            MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
            CSC_REP.REQ_210:= MAJOR_DATA;
           END
           ELSE
            IF MAJOR_DATA.COURSE = '370' THEN
            BEGIN
             TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
             MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
             CSC_REP.REQ_370:= MAJOR_DATA;
            END
            ELSE
              EXTRA_BASIC[COUNT]:= MAJOR_DATA;
  END;
END;
```

```
            IF MAJOR_DATA.COURSE = '210' THEN
            BEGIN
             TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
             MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
             CSC_REP.REQ_210:= MAJOR_DATA;
            END
            ELSE
             IF MAJOR_DATA.COURSE = '370' THEN
             BEGIN
              TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOUR));
              MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
              CSC_REP.REQ_370:= MAJOR_DATA;
             END
             ELSE
               EXTRA_BASIC[COUNT]:= MAJOR_DATA;
  END;
END;

PROCEDURE EVALUATE_MAJOR(MAJOR_DATA : CL_RECORD;
                         VAR MAJOR_TOT : INTEGER;
                         COUNT : INTEGER;
                         VAR MAJ_REPORT : MAJ_ARRAY);

VAR
  TEMP_HRS : INTEGER;
BEGIN
  MAJ_REPORT[COUNT]:= MAJOR_DATA;
  TEMP_HRS:= TRUNC(STOR(MAJOR_DATA.HOURS));
  MAJOR_TOT:= MAJOR_TOT + TEMP_HRS;
END;

PROCEDURE NEW_MAJOR (VAR MAJ_FILE : FILE OF CHAR;
                     VAR MAJOR_DATA : CL_RECORD);
VAR
  SEM : PACKED ARRAY[1..3] OF CHAR;
  NAM : PACKED ARRAY[1..34] OF CHAR;
  SSN : PACKED ARRAY[1..9] OF CHAR;
  DEP : PACKED ARRAY[1..3] OF CHAR;
  CRS : PACKED ARRAY[1..3] OF CHAR;
  HR  : PACKED ARRAY[1..2] OF CHAR;
  GR  : PACKED ARRAY[1..2] OF CHAR;


BEGIN
   READLN(MAJ_FILE,SSN,NAM,DEP,CRS,HR,GR,SEM);
   MAJOR_DATA.SSAN:= SSN;
   MAJOR_DATA.NAME:= NAM;
   MAJOR_DATA.DEPT:= DEP;
   MAJOR_DATA.COURSE:= CRS;
   MAJOR_DATA.HOUR:= HR;
   MAJOR_DATA.GRADE:= GR;
   MAJOR_DATA.SEMESTER:= SEM;
END;

PROCEDURE NEW_EXTRA(VAR EXTRA_FILE : FILE OF CHAR;
                    VAR EX_DATA : CL_RECORD);

VAR
  SEM : PACKED ARRAY[1..9] OF CHAR;
  NAM : PACKED ARRAY[1..34] OF CHAR;
  SSN : PACKED ARRAY[1..9] OF CHAR;
  DEP : PACKED ARRAY[1..3] OF CHAR;
  CRS : PACKED ARRAY[1..3] OF CHAR;
  HR  : PACKED ARRAY[1..2] OF CHAR;
  GR  : PACKED ARRAY[1..2] OF CHAR;

BEGIN
  READLN(EXTRA_FILE,SSN,NAM,DEP,CRS,HR,GR,SEM);
  EX_DATA.SSAN:= SSN;
  EX_DATA.NAME:= NAM;
  EX_DATA.DEPT:= DEP;
  EX_DATA.COURSE:= CRS;
  EX_DATA.HOUR:= HR;
  EX_DATA.GRADE:= GR;
  EX_DATA.SEMESTER:= SEM;
END;


PROCEDURE EVALUATE_EXTRA (VAR EX_REPORT : EX_ARRAY;
                          EX_DATA : CL_RECORD;
                          VAR EXTRA_TOT : INTEGER;
                          COUNT : INTEGER);

VAR
```

```
    HUMAN          = RECORD
  READLN(EXTRA_FILE,SSN,NAM,DEP,CRS,HR,GR,SEM);
  EX_DATA.SSAN:= SSN;
  EX_DATA.NAME:= NAM;
  EX_DATA.DEPT:= DEP;
  EX_DATA.COURSE:= CRS;
  EX_DATA.HOUR:= HR;
  EX_DATA.GRADE:= GR;
  EX_DATA.SEMESTER:= SEM;
END;


PROCEDURE EVALUATE_EXTRA (VAR EX_REPORT : EX_ARRAY;
                          EX_DATA : CL_RECORD;
                          VAR EXTRA_TOT : INTEGER;
                          COUNT : INTEGER);

VAR
  TENP_HRS : INTEGER;

BEGIN
  EX_REPORT[COUNT]:= EX_DATA;
  TEMP_HRS:= TRUNC(STOR(EX_DATA.HOURS)):
  EXTRA_TOT:= EXTRA_TOT + TEMP_HRS;
END;

(* MAIN LINE *)

BEGIN
  RESET(NEWFILE);
  RESET(GRADES);
  RESET(MAJ_FILE);
  RESET(EXTRA_FILE);
  INIT_REPORT(REPORT);
  GRAND_TOTAL := 0;
  COUNT:= 0;
  EXTRA_TOT:= 0;
  MAJOR_TOT:= 0;
  OVERALL_TOT:=0;
  BASIC_COUNT:= 0;
  MAJOR_COUNT:= 0;
  NEW_STUDENT(NEWFILE,STUDENT_DATA);
  NEW_GRADE(GRADES,GRADE_DATA);
  NEW_MAJOR(MAJ_FILE,MAJOR_DATA);
  NEW_EXTRA(EXTRA_FILE,EX_DATA);
  WHILE NOT EOF(NEW_FILE) DO
    BEGIN
      WHILE STUDENT_DATA.SSAN = GRADE_DATA.SSAN DO
        BEGIN
          COUNT:= COUNT + 1;
          LOC:= GRADE_DATA.LOC1[1];
          CASE LOC OF
            'A' : PART_A(GRADE_DATA,REPORT.BASIC_SKILLS,
                         GRAND_TOTAL,COUNT,EXTRA_BASIC);
            'B' : PART_B(GRADE_DATA,REPORT.HUMANITIES,
                         GRAND_TOTAL,COUNT,EXTRA_BASIC);
            'C' : PART_C(GRADE_DATA,REPORT.SOCIAL_SCIENCES,
                         GRAND_TOTAL,COUNT,EXTRA_BASIC);
            'D' : PART_D(GRADE_DATA,REPORT.SCIENCE_MATHEMATICS,
                         GRAND_TOTAL,COUNT,EXTRA_BASIC);
            'E' : PART_E(GRADE_DATA,REPORT.PHYSICAL_EDUCATION,
                         GRAND_TOTAL,COUNT,EXTRA_BASIC);
          END;
          IF NOT EOF(GRADES) THEN
            NEW_GRADE(GRADES,GRADE_DATA);
PAGE;
        END;
      BASIC_COUNT:= COUNT;
      COUNT:= 0;
      OVERALL_TOT:= OVERALL_TOT + GRAND_TOTAL;
      CHECK_EXTRA_BASIC(EXTRA_BASIC,BASIC_COUNT);
      WHILE STUDENT_DATA.SSAN = MAJOR_DATA.SSAN DO
        BEGIN
          COUNT:= COUNT + 1;
          IF STUDENT_DATA.MAJOR = 'MAT' THEN
            EVALUATE_MATH(MAJOR_DATA,MAJOR_TOT,COUNT,MAT_REP,
                          EXTRA_BASIC)
          ELSE
            IF STUDENT_DATA.MAJOR = 'CSC' THEN
              EVALUATE_CSC(MAJOR_DATA,EXTRA_BASIC,COUNT,
                           CSC_REP,MAJOR_TOT)
            ELSE
              EVALUATE_MAJOR(MAJOR_DATA,MAJOR_TOT,COUNT,
                             MAJ_REP);
```

```
PAGE;
        END;
      BASIC_COUNT:= COUNT;
      COUNT:= 0;
      OVERALL_TOT:= OVERALL_TOT + GRAND_TOTAL;
      CHECK_EXTRA_BASIC(EXTRA_BASIC,BASIC_COUNT);
      WHILE STUDENT_DATA.SSAN = MAJOR_DATA.SSAN DO
        BEGIN
          COUNT:= COUNT + 1;
          IF STUDENT_DATA.MAJOR = 'MAT' THEN
             EVALUATE_MATH(MAJOR_DATA,MAJOR_TOT,COUNT,MAT_REP,
                           EXTRA_BASIC)
          ELSE
            IF STUDENT_DATA.MAJOR = 'CSC' THEN
               EVALUATE_CSC(MAJOR_DATA,EXTRA_BASIC,COUNT,
                            CSC_REP,MAJOR_TOT)
            ELSE
              EVALUATE_MAJOR(MAJOR_DATA,MAJOR_TOT,COUNT,
                             MAJ_REP);
          IF NOT EOF(MAJ_FILE) THEN
            NEW_MAJOR(MAJ_FILE,MAJOR_DATA);
        MAJOR_COUNT:= COUNT;
        END;
      COUNT:= 0;
      OVERALL_TOT:= OVERALL_TOT + MAJOR_TOT;
      WHILE STUDENT_DATA.SSAN = EX_DATA.SSAN DO
        BEGIN
          COUNT:= COUNT + 1;
          EVALUATE_EXTRA(EX_REPORT,EX_DATA,EXTRA_TOT,COUNT);
          IF NOT EOF(EXTRA_FILE) THEN
            NEW_EXTRA(EXTRA_FILE,EX_DATA);
        END;
      B_COUNT:= COUNT;
      ADD_BASIC(B_COUNT,EX_REPORT,EXTRA_BASIC);
      EXTRA_COUNT:= B_COUNT;
      OVERALL_TOT:= OVERALL_TOT + EXTRA_TOT;
      PRINT(STUDENT_DATA,REPORT,GRAND_TOTAL,
            MAJOR_COUNT,EXTRA_COUNT);
      INIT_REPORT(REPORT);
      GRAND_TOTAL := 0;
      OVERALL_TOT:= 0;
      MAJOR_TOT:= 0;
      EXTRA_TOT:= 0;
      COUNT:= 0;
      NEW_STUDENT(NEWFILE,STUDENT_DATA);
    END;
END.
```

```
PROCEDURE PART_B (GRADE_DATA              : CLASS_REC;
                  VAR REPORT.HUMANITIES : HUMAN;
                  VAR GRAND_TOTAL         : INTEGER);
TYPE
LOCATION_AREA = PACKED ARRAY [.1..3.] OF CHAR;
TEMP_RECORD = RECORD
                COURSE_NUM      : PACKED ARRAY [.1..3.] OF CHAR;
                CRSE_HOURS      : PACKED ARRAY [.1..2.] OF CHAR;
                DEPT            : PACKED ARRAY [.1..3.] OF CHAR;
                GRADE           : PACKED ARRAY [.1..2.] OF CHAR;
                LOC1            : PACKED ARRAY [.1..3.] OF CHAR;
                LOC2            : PACKED ARRAY [.1..3.] OF CHAR;
                LOC3            : PACKED ARRAY [.1..3.] OF CHAR;
                LOC4            : PACKED ARRAY [.1..3.] OF CHAR;
                LOC5            : PACKED ARRAY [.1..3.] OF CHAR;
                RPT             : CHAR;
                SEMESTER        : PACKED ARRAY [.1..3.] OF CHAR;
              END;

VAR
FILLED          : BOOLEAN;
PLACED          : BOOLEAN;
MORE            : BOOLEAN;
STOR_FILLED     : BOOLEAN;
TEMP_LOC        : LOCATION_AREA;
HUM_LOC1        : CHAR;
HUM_LOC2        : CHAR;
HUM_LOC3        : CHAR;
STORAGE_REC1    : TEMP_RECORD;
STORAGE_REC2    : TEMP_RECORD;
NEW_LOC1A       : CHAR;
NEW_LOC1B       : CHAR;
NEW_LOC2A       : CHAR;
NEW_LOC2B       : CHAR;
STORAGE_REC     : TEMP_RECORD;

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

PROCEDURE CONVERT_LOCATION (TEMP_AREA      : LOCATION_AREA;
                            VAR HUM_LOC1   : CHAR;
                            VAR HUM_LOC2   : CHAR;
                            VAR HUM_LOC3   : CHAR);

(*THIS PROCEDURE CONVERTS THE LOCATION OF THE BASIC STUDIES
COURSE INTO THREE SEPARATE VARIABLES, SO THAT WHEN CHECKING
THE COURSE FOR PLACEMENT INTO THE RECORD, THERE IS ONLY ONE
VARIABLE CHECKED.*)

VAR
  AREA1 : CHAR;
  AREA2 : CHAR;
  AREA3 : CHAR;

BEGIN
   SUBSTR (TEMP_AREA,1,1,AREA1);
   HUM_LOC1:= AREA1;
   SUBSTR (TEMP_AREA,2,1,AREA2);
   HUM_LOC2:= AREA2;
   SUBSTR (TEMP_AREA,3,1,AREA3);
   HUM_LOC3:= AREA3;
END;


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


PROCEDURE CHECK_NUMBER (NUM_OF_AREAS  : CHAR;
                        VAR MORE      : BOOLEAN);

(*THIS PROCEDURE CHECKS THE CLASS RECORD TO SEE IF THE COURSE
CAN BE PLACED INTO MORE THAN ONE LOCATION.  IF IT CAN BE
PLACED INTO MORE THAN ONE LOCATION, THEN A VARIABLE,
MORE, IS SET TO TRUE. *)

VAR
  NUM_LOCATIONS:REAL;

BEGIN
  MORE := FALSE;
  NUM_LOCATIONS:= STOR (NUM_OF_AREAS);
  TRUNC(NUM_LOCATIONS);
```

```
(*******************************************************)

PROCEDURE CHECK_NUMBER (NUM_OF_AREAS  : CHAR;
                        VAR MORE      : BOOLEAN);

(*THIS PROCEDURE CHECKS THE CLASS RECORD TO SEE IF THE COURSE
CAN BE PLACED INTO MORE THAN ONE LOCATION.  IF IT CAN BE
PLACED INTO MORE THAN ONE LOCATION, THEN A VARIABLE,
MORE, IS SET TO TRUE. *)

VAR
  NUM_LOCATIONS:REAL;

BEGIN
  MORE := FALSE;
  NUM_LOCATIONS:= STOR (NUM_OF_AREAS);
  TRUNC(NUM_LOCATIONS);
  IF NUM_LOCATIONS > 1 THEN
     MORE := TRUE;
END;

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


PROCEDURE CHECK_LOCATION (VAR FILLED       : BOOLEAN;
                          VAR STORAGE_REC : TEMP_RECORD;
                          LOCATION         : STR1;
                          LOCATION2        : STR1;
                          AREA_B           : HUMAN;
                          TEMP_FILLED      : BOOLEAN);


(*THIS PROCEDURE CHECKS THE FIRST ELEMENT OF THE BASIC STUDY
COURSE CODE TO FIND THE PROPER AREA THAT THE COURSE NEEDS
TO BE PLACED INTO THE RECORD.  IF THE AREA THAT THE COURSE
CAN BE INSERTED INTO IS FILLED, THE VALUE FOR FILLED IS
SENT BACK AS TRUE*)

CONST
  MIN_HOURS = 3;
  MAX_HOURS = 6;


PROCEDURE INSERT_TEMP (AREA_B             : HUMAN;
                       VAR FILLED         : BOOLEAN;
                       VAR STORAGE_REC    : TEMP_RECORD;
                       LOCATION           : STR1;
                       LOCATION2          : STR1);
BEGIN
    IF LOCATION = "1" THEN
    BEGIN
      STORAGE_REC.COURSE_NUM:= AREA_B.FINE_ARTS.COURSE_NUM;
      STORAGE_REC.HOURS:= AREA_B.FINE_ARTS.CRSE_HOURS;
      STORAGE_REC.DEPT:= AREA_B.FINE_ARTS.DEPT_CODE;
      STORAGE_REC.GRADE:= AREA_B.FINE_ARTS.CRSE_GRADE;
      STORAGE_REC.LOC1:= AREA_B.FINE_ARTS.LOC1;
      STORAGE_REC.LOC2:= AREA_B.FINE_ARTS.LOC2;
      STORAGE_REC.LOC3:= AREA_B.FINE_ARTS.LOC3;
      STORAGE_REC.LOC4:= AREA_B.FINE_ARTS.LOC4;
      STORAGE_REC.LOC5:= AREA_B.FINE_ARTS.LOC5;
      STORAGE_REC.RPT:= AREA_B.FINE_ARTS.RPT;
      STORAGE_REC.SEMESTER:= AREA_B.FINE_ARTS.SEMESTER;
   END;
   IF LOCATION = "2" THEN
      BEGIN
      STORAGE_REC.COURSE_NUM:= AREA_B.LITERATURE.COURSE_NUM;
      STORAGE_REC.HOURS:= AREA_B.LITERATURE.CRSE_HOURS;
      STORAGE_REC.DEPT:= AREA_B.LITERATURE.DEPT_CODE;
      STORAGE_REC.GRADE:= AREA_B.LITERATURE.CRSE_GRADE;
      STORAGE_REC.LOC1:= AREA_B.LITERATURE.LOC1;
      STORAGE_REC.LOC2:= AREA_B.LITERATURE.LOC2;
      STORAGE_REC.LOC3:= AREA_B.LITERATURE.LOC3;
      STORAGE_REC.LOC4:= AREA_B.LITERATURE.LOC4;
      STORAGE_REC.LOC5:= AREA_B.LITERATURE.LOC5;
      STORAGE_REC.RPT:= AREA_B.LITERATURE.RPT;
      STORAGE_REC.SEMESTER:= AREA_B.LITERATURE.SEMESTER;
   END;
   IF LOCATION = "3" THEN
   BEGIN
      STORAGE_REC.COURSE_NUM:= AREA_B.HISTORY.COURSE_NUM;
      STORAGE_REC.HOURS:= AREA_B.HISTORY.CRSE_HOURS;
      STORAGE_REC.DEPT:= AREA_B.HISTORY.DEPT_CODE;
      STORAGE_REC.GRADE:= AREA_B.HISTORY.CRSE_GRADE;
```

```
    STORAGE_REC.COURSE_NUM:= AREA_B.LITERATURE.COURSE_NUM;
    STORAGE_REC.HOURS:= AREA_B.LITERATURE.CRSE_HOURS;
    STORAGE_REC.DEPT:= AREA_B.LITERATURE.DEPT_CODE;
    STORAGE_REC.GRADE:= AREA_B.LITERATURE.CRSE_GRADE;
    STORAGE_REC.LOC1:= AREA_B.LITERATURE.LOC1;
    STORAGE_REC.LOC2:= AREA_B.LITERATURE.LOC2;
    STORAGE_REC.LOC3:= AREA_B.LITERATURE.LOC3;
    STORAGE_REC.LOC4:= AREA_B.LITERATURE.LOC4;
    STORAGE_REC.LOC5:= AREA_B.LITERATURE.LOC5;
    STORAGE_REC.RPT:= AREA_B.LITERATURE.RPT;
    STORAGE_REC.SEMESTER:= AREA_B.LITERATURE.SEMESTER;
END;
IF LOCATION = "3" THEN
BEGIN
    STORAGE_REC.COURSE_NUM:= AREA_B.HISTORY.COURSE_NUM;
    STORAGE_REC.HOURS:= AREA_B.HISTORY.CRSE_HOURS;
    STORAGE_REC.DEPT:= AREA_B.HISTORY.DEPT_CODE;
    STORAGE_REC.GRADE:= AREA_B.HISTORY.CRSE_GRADE;
    STORAGE_REC.LOC1:= AREA_B.HISTORY.LOC1;
    STORAGE_REC.LOC2:= AREA_B.HISTORY.LOC2;
    STORAGE_REC.LOC3:= AREA_B.HISTORY.LOC3;
    STORAGE_REC.LOC4:= AREA_B.HISTORY.LOC4;
    STORAGE_REC.LOC5:= AREA_B.HISTORY.LOC5;
    STORAGE_REC.RPT:= AREA_B.HISTORY.RPT;
    STORAGE_REC.SEMESTER:= AREA_B.HISTORY.SEMESTER;
END;
IF LOCATION = "4" THEN
BEGIN
    STORAGE_REC.COURSE_NUM:= AREA_B.PHIL_REL.COURSE_NUM;
    STORAGE_REC.HOURS:= AREA_B.PHIL_REL.CRSE_HOURS;
    STORAGE_REC.DEPT:= AREA_B.PHIL_REL.DEPT_CODE;
    STORAGE_REC.GRADE:= AREA_B.PHIL_REL.CRSE_GRADE;
    STORAGE_REC.LOC1:= AREA_B.PHIL_REL.LOC1;
    STORAGE_REC.LOC2:= AREA_B.PHIL_REL.LOC2;
    STORAGE_REC.LOC3:= AREA_B.PHIL_REL.LOC3;
    STORAGE_REC.LOC4:= AREA_B.PHIL_REL.LOC4;
    STORAGE_REC.LOC5:= AREA_B.PHIL_REL.LOC5;
    STORAGE_REC.RPT:= AREA_B.PHIL_REL.RPT;
    STORAGE_REC.SEMESTER:= AREA_B.PHIL_REL.SEMESTER;
END;
IF LOCATION = "5" THEN
BEGIN
  IF LOCATION2 = "1" THEN
  BEGIN
     STORAGE_REC.COURSE_NUM:=
             AREA_B.HUM_ELECTIVES.AREA1.COURSE_NUM;
     STORAGE_REC.HOURS:=
             AREA_B.HUM_ELECTIVES.AREA1.CRSE_HOURS;
     STORAGE_REC.DEPT:=
             AREA_B.HUM_ELECTIVES.AREA1.DEPT_CODE;
     STORAGE_REC.GRADE:=
             AREA_B.HUM_ELECTIVES.AREA1.CRSE_GRADE;
     STORAGE_REC.LOC1:=
             AREA_B.HUM_ELECTIVES.AREA1.LOC1;
     STORAGE_REC.LOC2:=
             AREA_B.HUM_ELECTIVES.AREA1.LOC2;
     STORAGE_REC.LOC3:=
             AREA_B.HUM_ELECTIVES.AREA1.LOC3;
     STORAGE_REC.LOC4:=
             AREA_B.HUM_ELECTIVES.AREA1.LOC4;
     STORAGE_REC.LOC5:=
             AREA_B.HUM_ELECTIVES.AREA1.LOC5;
     STORAGE_REC.RPT:=
             AREA_B.HUM_ELECTIVES.AREA1.RPT;
     STORAGE_REC.SEMESTER:=
             AREA_B.HUM_ELECTIVES.AREA1.SEMESTER;
  END;
  IF LOCATION2 = "2" THEN
  BEGIN
     STORAGE_REC.COURSE_NUM:=
             AREA_B.HUM_ELECTIVES.AREA2.COURSE_NUM;
     STORAGE_REC.HOURS:=
             AREA_B.HUM_ELECTIVES.AREA2.CRSE_HOURS;
     STORAGE_REC.DEPT:=
             AREA_B.HUM_ELECTIVES.AREA2.DEPT_CODE;
     STORAGE_REC.GRADE:=
             AREA_B.HUM_ELECTIVES.AREA2.CRSE_GRADE;
     STORAGE_REC.LOC1:=
             AREA_B.HUM_ELECTIVES.AREA2.LOC1;
     STORAGE_REC.LOC2:=
             AREA_B.HUM_ELECTIVES.AREA2.LOC2;
     STORAGE_REC.LOC3:=
             AREA_B.HUM_ELECTIVES.AREA2.LOC3;
     STORAGE_REC.LOC4:=
```

```
END;
IF LOCATION2 = "2" THEN
BEGIN
   STORAGE_REC.COURSE_NUM:=
           AREA_B.HUM_ELECTIVES.AREA2.COURSE_NUM;
   STORAGE_REC.HOURS:=
           AREA_B.HUM_ELECTIVES.AREA2.CRSE_HOURS;
   STORAGE_REC.DEPT:=
           AREA_B.HUM_ELECTIVES.AREA2.DEPT_CODE;
   STORAGE_REC.GRADE:=
           AREA_B.HUM_ELECTIVES.AREA2.CRSE_GRADE;
   STORAGE_REC.LOC1:=
           AREA_B.HUM_ELECTIVES.AREA2.LOC1;
   STORAGE_REC.LOC2:=
           AREA_B.HUM_ELECTIVES.AREA2.LOC2;
   STORAGE_REC.LOC3:=
           AREA_B.HUM_ELECTIVES.AREA2.LOC3;
   STORAGE_REC.LOC4:=
           AREA_B.HUM_ELECTIVES.AREA2.LOC4;
   STORAGE_REC.LOC5:=
           AREA_B.HUM_ELECTIVES.AREA2.LOC5;
   STORAGE_REC.RPT:=
           AREA_B.HUM_ELECTIVES.AREA2.RPT;
   STORAGE_REC.SEMESTER:=
           AREA_B.HUM_ELECTIVES.AREA2.SEMESTER;
END;
IF LOCATION2 = "3" THEN
BEGIN
   STORAGE_REC.COURSE_NUM:=
        AREA_B.HUM_ELECTIVES.AREA3.COURSE_NUM;
   STORAGE_REC.HOURS:=
        AREA_B.HUM_ELECTIVES.AREA3.CRSE_HOURS;
   STORAGE_REC.DEPT:=
        AREA_B.HUM_ELECTIVES.AREA3.DEPT_CODE;
   STORAGE_REC.GRADE:=
        AREA_B.HUM_ELECTIVES.AREA3.CRSE_GRADE;
   STORAGE_REC.LOC1:=
        AREA_B.HUM_ELECTIVES.AREA3.LOC1;
   STORAGE_REC.LOC2:=
        AREA_B.HUM_ELECTIVES.AREA3.LOC2;
   STORAGE_REC.LOC3:=
        AREA_B.HUM_ELECTIVES.AREA3.LOC3;
   STORAGE_REC.LOC4:=
        AREA_B.HUM_ELECTIVES.AREA3.LOC4;
   STORAGE_REC.LOC5:=
        AREA_B.HUM_ELECTIVES.AREA3.LOC5;
   STORAGE_REC.RPT:=
        AREA_B.HUM_ELECTIVES.AREA3.RPT;
   STORAGE_REC.SEMESTER:=
        AREA_B.HUM_ELECTIVES.AREA3.SEMESTER;
END;
IF LOCATION2 = "4" THEN
   BEGIN
   STORAGE_REC.COURSE_NUM:=
       AREA_B.HUM_ELECTIVES.AREA4.COURSE_NUM;
   STORAGE_REC.HOURS:=
       AREA_B.HUM_ELECTIVES.AREA4.CRSE_HOURS;
   STORAGE_REC.DEPT:=
       AREA_B.HUM_ELECTIVES.AREA4.DEPT_CODE;
   STORAGE_REC.GRADE:=
       AREA_B.HUM_ELECTIVES.AREA4.CRSE_GRADE;
   STORAGE_REC.LOC1:=
       AREA_B.HUM_ELECTIVES.AREA4.LOC1;
   STORAGE_REC.LOC2:=
       AREA_B.HUM_ELECTIVES.AREA4.LOC2;
   STORAGE_REC.LOC3:=
       AREA_B.HUM_ELECTIVES.AREA4.LOC3;
   STORAGE_REC.LOC4:=
       AREA_B.HUM_ELECTIVES.AREA4.LOC4;
   STORAGE_REC.LOC5:=
       AREA_B.HUM_ELECTIVES.AREA4.LOC5;
   STORAGE_REC.RPT:=
       AREA_B.HUM_ELECTIVES.AREA4.RPT;
   STORAGE_REC.SEMESTER:=
       AREA_B.HUM_ELECTIVES.AREA4.SEMESTER;
END;
IF LOCATION2 = "5" THEN
   BEGIN
   STORAGE_REC.COURSE_NUM:=
       AREA_B.HUM_ELECTIVES.AREA5.COURSE_NUM;
   STORAGE_REC.HOURS:=
       AREA_B.HUM_ELECTIVES.AREA5.CRSE_HOURS;
   STORAGE_REC.DEPT:=
       AREA_B.HUM_ELECTIVES.AREA5.DEPT_CODE;
```

```
            AREA_B.HUM_ELECTIVES.AREA4.LOC3;
        STORAGE_REC.LOC4:=
            AREA_B.HUM_ELECTIVES.AREA4.LOC4;
        STORAGE_REC.LOC5:=
            AREA_B.HUM_ELECTIVES.AREA4.LOC5;
        STORAGE_REC.RPT:=
            AREA_B.HUM_ELECTIVES.AREA4.RPT;
        STORAGE_REC.SEMESTER:=
            AREA_B.HUM_ELECTIVES.AREA4.SEMESTER;
     END;
     IF LOCATION2 = "5" THEN
        BEGIN
        STORAGE_REC.COURSE_NUM:=
            AREA_B.HUM_ELECTIVES.AREA5.COURSE_NUM;
        STORAGE_REC.HOURS:=
            AREA_B.HUM_ELECTIVES.AREA5.CRSE_HOURS;
        STORAGE_REC.DEPT:=
            AREA_B.HUM_ELECTIVES.AREA5.DEPT_CODE;
        STORAGE_REC.GRADE:=
             AREA_B.HUM_ELECTIVES.AREA5.CRSE_GRADE;
        STORAGE_REC.LOC1:=
            AREA_B.HUM_ELECTIVES.AREA5.LOC1;
        STORAGE_REC.LOC2:=
             AREA_B.HUM_ELECTIVES.AREA5.LOC2;
        STORAGE_REC.LOC3:=
            AREA_B.HUM_ELECTIVES.AREA5.LOC3;
        STORAGE_REC.LOC4:=
             AREA_B.HUM_ELECTIVES.AREA5.LOC4;
        STORAGE_REC.LOC5:=
             AREA_B.HUM_ELECTIVES.AREA5.LOC5;
        STORAGE_REC.RPT:=
             AREA_B.HUM_ELECTIVES.AREA5.RPT;
        STORAGE_REC.SEMESTER:=
             AREA_B.HUM_ELECTIVES.AREA5.SEMESTER;
     END;
   END;
END;

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

BEGIN
  FILLED:= FALSE;
  IF LOCATION = "1" THEN
     IF AREA_B.ARTS_TOTAL = MIN_HOURS THEN
        IF STOR_FILLED THEN
          FILLED:= TRUE;
        ELSE
          INSERT_TEMP(AREA_B,FILLED,STORAGE_REC,
                      LOCATION,LOCATION2);
  IF LOCATION = "2" THEN
     IF AREA_B.LIT_ TOTAL = MIN_HOURS THEN
        IF STOR_FILLED THEN
          FILLED:= TRUE;
        ELSE
          INSERT_TEMP(AREA_B,FILLED,STORAGE_REC,
                      LOCATION,LOCATION2);
  IF LOCATION = "3" THEN
     IF AREA_B.HISTORY_TOTAL = MIN_HOURS THEN
        IF STOR_FILLED THEN
           FILLED:= TRUE;
        ELSE
           INSERT_TEMP(AREA_B,FILLED,
                       STORAGE_REC,LOCATION,LOCATION2);
  IF LOCATION = "4" THEN
     IF AREA_B.PHIL_REL_TOTAL = MIN_HOURS THEN
        IF STOR_FILLED THEN
           FILLED:= TRUE;
        ELSE
          INSERT_TEMP (AREA_B,FILLED,
                       STORAGE_REC,LOCATION,LOCATION2);
  IF LOCATION = "5" THEN
     IF AREA_B.ELEC_TOTAL = MAX_HOURS THEN
     BEGIN
        FILLED:= TRUE;
        IF LOCATION2 = "1" THEN
           IF AREA_B.HUM_ELECTIVES.AREA1_TOT = MIN_HOURS THEN
              IF STOR_FILLED THEN
                 FILLED:= TRUE;
              ELSE
                INSERT_TEMP (AREA_B,FILLED,
                             STORAGE_REC,LOCATION,LOCATION2);
        IF LOCATION2 = "2" THEN
           IF AREA_B.HUM_ELECTIVES.AREA2_TOT = MIN_HOURS THEN
```

```
          FILLED:= TRUE;
       ELSE
         INSERT_TEMP (AREA_B,FILLED,
                      STORAGE_REC,LOCATION,LOCATION2);
IF LOCATION = "5" THEN
   IF AREA_B.ELEC_TOTAL = MAX_HOURS THEN
   BEGIN
      FILLED:= TRUE;
      IF LOCATION2 = "1" THEN
         IF AREA_B.HUM_ELECTIVES.AREA1_TOT = MIN_HOURS THEN
            IF STOR_FILLED THEN
               FILLED:= TRUE;
            ELSE
              INSERT_TEMP (AREA_B,FILLED,
                           STORAGE_REC,LOCATION,LOCATION2);
      IF LOCATION2 = "2" THEN
         IF AREA_B.HUM_ELECTIVES.AREA2_TOT = MIN_HOURS THEN
            IF STOR_FILLED THEN
               FILLED:= TRUE;
            ELSE
              INSERT_TEMP (AREA_B,FILLED,
                           STORAGE_REC,LOCATION.LOCATION2);
      IF LOCATION2 = "3" THEN
         IF AREA_B.HUM_ELECTIVES.AREA3_TOT = MIN_HOURS THEN
            IF STOR_FILLED THEN
               FILLED:= TRUE;
            ELSE
              INSERT_TEMP (AREA_B, FILLED,
                           STORAGE_REC,LOCATION,LOCATION2);
      IF LOCATION2 = "4" THEN
         IF AREA_B.HUM_ELECTIVES.AREA4_TOT = MIN_HOURS THEN
            IF STOR_FILLED THEN
               FILLED:= TRUE;
            ELSE
              INSERT_TEMP (AREA_B,FILLED,
                           STORAGE_REC,LOCATION,LOCATION2);
      IF LOCATION2 = "5" THEN
         IF AREA_B.HUM_ELECTIVES.AREA5_TOT = MIN_HOURS THEN
            IF STOR_FILLED THEN
               FILLED:= TRUE;
            ELSE
               INSERT_TEMP (AREA_B,FILLED,
                            STORAGE_REC,LOCATION,LOCATION2);
   END
      ELSE
         IF AREA_B.ELEC_TOTAL = MIN_HOURS THEN
         BEGIN
            IF LOCATION2 = "1" THEN
               IF AREA_B.HUM_ELECTIVES.AREA1_TOT = MIN_HOURS
               THEN
                  IF STOR_FILLED THEN
                     FILLED:= TRUE;
                  ELSE
                    INSERT_TEMP (,AREA_B,FILLED,STORAGE_REC,
                                 LOCATION,LOCATION2);
            IF LOCATION2 = "2" THEN
                IF AREA_B.HUM_ELECTIVES.AREA2_TOT =
               MIN_HOURS THEN
                   IF STOR_FILLED THEN
                      FILLED:= TRUE;
                   ELSE
                      INSERT_TEMP (AREA_B,FILLED,
                       STORAGE_REC,LOCATION,LOCATION2);
            IF LOCATION2 = "3" THEN
               IF AREA_B.HUM_ELECTIVES.AREA3_TOT =
                           MIN_HOURS THEN
                  IF STOR_FILLED THEN
                     FILLED:= TRUE;
                  ELSE
                     INSERT_TEMP (AREA_B,FILLED,
                      STORAGE_REC,LOCATION,LOCATION2);
            IF LOCATION2 = "4"THEN
               IF AREA_B.HUM_ELECTIVES.AREA4_TOT =
                        MIN_HOURS THEN
                  IF STOR_FILLED THEN
                     FILLED:= TRUE;
                  ELSE
                     INSERT_TEMP (AREA_B,FILLED,
                     STORAGE_REC,LOCATION,LOCATION2);
            IF LOCATION2 = "5" THEN
               IF AREA_B.HUM_ELECTIVES.AREA5_TOT =
                         MIN_HOURS THEN
                  IF STOR_FILLED THEN
                     FILLED:= TRUE;
```

```
                              MIN_HOURS THEN
                        IF STOR_FILLED THEN
                           FILLED:= TRUE;
                        ELSE
                           INSERT_TEMP (AREA_B,FILLED,
                            STORAGE_REC,LOCATION,LOCATION2);
                  IF LOCATION2 = "4"THEN
                     IF AREA_B.HUM_ELECTIVES.AREA4_TOT =
                              MIN_HOURS THEN
                        IF STOR_FILLED THEN
                           FILLED:= TRUE;
                        ELSE
                           INSERT_TEMP (AREA_B,FILLED,
                           STORAGE_REC,LOCATION,LOCATION2);
                  IF LOCATION2 = "5" THEN
                     IF AREA_B.HUM_ELECTIVES.AREA5_TOT =
                           MIN_HOURS THEN
                        IF STOR_FILLED THEN
                           FILLED:= TRUE;
                        ELSE
                           INSERT_TEMP (AREA_B,FILLED,
                           STORAGE_REC,LOCATION,LOCATION2);

               END;
END;
(*************************************************************)


PROCEDURE INSERT_CLASS (VAR AREA_B         : HUMAN;
                            LOCATION1      : CHAR;
                            LOCATION2      : CHAR;
                            GRADE_DATA     : CLASS_REC:
                            VAR GRAND_TOTAL : INTEGER);

(*THIS PROCEDURE INSERTS THE CLASS INTO THE RECORD*)


VAR
  NUMBER : REAL;

BEGIN
     WITH AREA_B DO
     IF LOCATION1 = '1' THEN
        BEGIN
        FINE_ARTS.DEPT_CODE:= GRADE_DATA.DEPT;
        FINE_ARTS.COURSE_NUM:= GRADE_DATA.COURSE;
        FINE_ARTS.CRSE_GRADE:= GRADE_DATA.GRADE;
        FINE_ARTS.CRSE_HOURS:= GRADE_DATA.HOURS;
        FINE_ARTS.LOC1:= GRADE_DATA.LOC1;
        FINE_ARTS.LOC2:= GRADE_DATA.LOC2;
        FINE_ARTS.LOC3:= GRADE_DATA.LOC3;
        FINE_ARTS.LOC4:= GRADE_DATA.LOC4;
        FINE_ARTS.LOC5:= GRADE_DATA.LOC5;
        FINE_ARTS.RPT := GRADE_DATA.RPT;
        FINE_ARTS.SEMESTER:= GRADE_DATA.SEMESTER;
        ARTS_TOTAL:= TRUNC(STOR(GRADE_DATA.HOURS));
        END;
     IF LOCATION1 = '2' THEN
        BEGIN
        LITERATURE.DEPT_CODE:= GRADE_DATA.DEPT;
        LITERATURE.COURSE_NUM:= GRADE_DATA.COURSE;
        LITERATURE.CRSE_GRADE:= GRADE_DATA.GRADE;
        LITERATURE.CRSE_HOURS:= GRADE_DATA.SEMESTER;
        LITERATURE.LOC1:= GRADE_DATA.LOC1;
        LITERATURE.LOC2:= GRADE_DATA.LOC2;
        LITERATURE.LOC3:= GRADE_DATA.LOC3;
        LITERATURE.LOC4:= GRADE_DATA.LOC4;
        LITERATURE.LOC5:= GRADE_DATA.LOC5;
        LITERATURE.RPT:= GRADE_DATA.RPT;
        LITERATURE.SEMESTER:= GRADE_DATA.SEMESTER;
        LIT_TOTAL:= TRUNC(STOR(GRADE_DATA.HOURS));
        END;
     IF LOCATION1 = '3' THEN
        BEGIN
        HISTORY.DEPT_CODE:= GRADE_DATA.DEPT;
        HISTORY.COURSE_NUM:= GRADE_DATA.COURSE;
        HISTORY.CRSE_GRADE:= GRADE_DATA.GRADE;
        HISTORY.CRSE_HOURS:= GRADE_DATA.SEMESTER;
        HISTORY.LOC1:= GRADE_DATA.LOC1;
        HISTORY.LOC2:= GRADE_DATA.LOC2;
        HISTORY.LOC3:= GRADE_DATA.LOC3;
        HISTORY.LOC4:= GRADE_DATA.LOC4;
        HISTORY.LOC5:= GRADE_DATA.LOC5;
        HISTORY.RPT:= GRADE_DATA.RPT;
```

```
    LITERATURE.LOC5:= GRADE_DATA.LOC5;
    LITERATURE.RPT:= GRADE_DATA.RPT;
    LITERATURE.SEMESTER:= GRADE_DATA.SEMESTER;
    LIT_TOTAL:= TRUNC(STOR(GRADE_DATA.HOURS));
    END;
  IF LOCATION1 = '3' THEN
    BEGIN
    HISTORY.DEPT_CODE:= GRADE_DATA.DEPT;
    HISTORY.COURSE_NUM:= GRADE_DATA.COURSE;
    HISTORY.CRSE_GRADE:= GRADE_DATA.GRADE;
    HISTORY.CRSE_HOURS:= GRADE_DATA.SEMESTER;
    HISTORY.LOC1:= GRADE_DATA.LOC1;
    HISTORY.LOC2:= GRADE_DATA.LOC2;
    HISTORY.LOC3:= GRADE_DATA.LOC3;
    HISTORY.LOC4:= GRADE_DATA.LOC4;
    HISTORY.LOC5:= GRADE_DATA.LOC5;
    HISTORY.RPT:= GRADE_DATA.RPT;
    HISTORY.SEMESTER:= GRADE_DATA.SEMESTER;
    HISTORY_TOTAL:= TRUNC(STOR(GRADE_DATA.HOURS));
    END;
  IF LOCATION1 = '4' THEN
    BEGIN
    PHIL_REL.DEPT_CODE:= GRADE_DATA.DEPT;
    PHIL_REL.COURSE_NUM:= GRADE_DATA.COURSE;
    PHIL_REL.CRSE_GRADE:= GRADE_DATA.GRADE;
    PHIL_REL.CRSE_HOURS:= GRADE_DATA.SEMESTER;
    PHIL_REL.LOC1:= GRADE_DATA.LOC1;
    PHIL_REL.LOC2:= GRADE_DATA.LOC2;
    PHIL_REL.LOC3:= GRADE_DATA.LOC3;
    PHIL_REL.LOC4:= GRADE_DATA.LOC4;
    PHIL_REL.LOC5:= GRADE_DATA.LOC5;
    PHIL_REL.RPT:= GRADE_DATA.RPT;
    PHIL_REL.SEMESTER:= GRADE_DATA.SEMESTER;
    PHIL_REL_TOTAL:= TRUNC(STOR(GRADE_DATA.HOURS));
    END;
  IF LOCATION1 = '5' THEN
    BEGIN
    IF LOCATION2 = '1' THEN
      BEGIN
      HUM_ELECTIVES.AREA1.DEPT_CODE:= GRADE_DATA.DEPT;
      HUM_ELECTIVES.AREA1.COURSE_NUM:=
          GRADE_DATA.COURSE;
      HUM_ELECTIVES.AREA1.CRSE_GRADE:=
          GRADE_DATA.GRADE;
      HUM_ELECTIVES.AREA1.CRSE_HOURS:=
          GRADE_DATA.HOURS;
      HUM_ELECTIVES.AREA1.LOC1:= GRADE_DATA.LOC1;
      HUM_ELECTIVES.AREA1.LOC2:= GRADE_DATA.LOC2;
      HUM_ELECTIVES.AREA1.LOC3:= GRADE_DATA.LOC3;
      HUM_ELECTIVES.AREA1.LOC4:= GRADE_DATA.LOC4;
      HUM_ELECTIVES.AREA1.LOC5:= GRADE_DATA.LOC5;
      HUM_ELECTIVES.AREA1.RPT:= GRADE_DATA.RPT;
      HUM_ELECTIVES.AREA1.SEMESTER:=
          GRADE_DATA.SEMESTER;
      HUM_ELECTIVES.AREA1_TOT:=
          TRUNC(STOR(GRADE_DATA.HOURS));
      END;
    IF LOCATION2 = '2' THEN
      BEGIN
      HUM_ELECTIVES.AREA2.DEPT_CODE:= GRADE_DATA.DEPT;
      HUM_ELECTIVES.AREA2.COURSE_NUM:=
          GRADE_DATA.COURSE;
      HUM_ELECTIVES.AREA2.CRSE_GRADE:=
          GRADE_DATA.GRADE;
      HUM_ELECTIVES.AREA2.CRSE_HOURS:=
          GRADE_DATA.HOURS;
      HUM_ELECTIVES.AREA2.LOC1:= GRADE_DATA.LOC1;
      HUM_ELECTIVES.AREA2.LOC2:= GRADE_DATA.LOC2;
      HUM_ELECTIVES.AREA2.LOC3:= GRADE_DATA.LOC3;
      HUM_ELECTIVES.AREA2.LOC4:= GRADE_DATA.LOC4;
      HUM_ELECTIVES.AREA2.LOC5:= GRADE_DATA.LOC5;
      HUM_ELECTIVES.AREA2.RPT:= GRADE_DATA.RPT;
      HUM_ELECTIVES.AREA2.SEMESTER:=
          GRADE_DATA.SEMESTER;
      HUM_ELECTIVES.AREA2_TOT:=
          HUM_ELECTIVES.AREA2_TOT +
          TRUNC(STOR(GRADE_DATA.HOURS));
      END;
    IF LOCATION2 = '3' THEN
      BEGIN
      HUM_ELECTIVES.AREA3.DEPT_CODE:= GRADE_DATA.DEPT;
      HUM_ELECTIVES.AREA3.COURSE_NUM:=
          GRADE_DATA.COURSE;
```

```
            GRADE_DATA.HOURS);
          HUM_ELECTIVES.AREA2.LOC1:= GRADE_DATA.LOC1;
          HUM_ELECTIVES.AREA2.LOC2:= GRADE_DATA.LOC2;
          HUM_ELECTIVES.AREA2.LOC3:= GRADE_DATA.LOC3;
          HUM_ELECTIVES.AREA2.LOC4:= GRADE_DATA.LOC4;
          HUM_ELECTIVES.AREA2.LOC5:= GRADE_DATA.LOC5;
          HUM_ELECTIVES.AREA2.RPT:= GRADE_DATA.RPT;
          HUM_ELECTIVES.AREA2.SEMESTER:=
              GRADE_DATA.SEMESTER;
          HUM_ELECTIVES.AREA2_TOT:=
              HUM_ELECTIVES.AREA2_TOT +
              TRUNC(STOR(GRADE_DATA.HOURS));
          END;
       IF LOCATION2 = '3' THEN
          BEGIN
          HUM_ELECTIVES.AREA3.DEPT_CODE:= GRADE_DATA.DEPT;
          HUM_ELECTIVES.AREA3.COURSE_NUM:=
              GRADE_DATA.COURSE;
          HUM_ELECTIVES.AREA3.CRSE_GRADE:=
              GRADE_DATA.GRADE;
          HUM_ELECTIVES.AREA3.CRSE_HOURS:=
              GRADE_DATA.HOURS;
          HUM_ELECTIVES.AREA3.LOC1:= GRADE_DATA.LOC1;
          HUM_ELECTIVES.AREA3.LOC2:= GRADE_DATA.LOC2;
          HUM_ELECTIVES.AREA3.LOC3:= GRADE_DATA.LOC3;
          HUM_ELECTIVES.AREA3.LOC4:= GRADE_DATA.LOC4;
          HUM_ELECTIVES.AREA3.LOC5:= GRADE_DATA.LOC5;
          HUM_ELECTIVES.AREA3.RPT:= GRADE_DATAg.RPT;
          HUM_ELECTIVES.AREA3.SEMESTER:=
              GRADE_DATA.SEMESTER;
          HUM_ELECTIVES.AREA3_TOT:=
              TRUNC(STOR(GRADE_DATA.HOURS));
          END;
       IF LOCATION2 = '4' THEN
          BEGIN
          HUM_ELECTIVES.AREA4.DEPT_CODE:= GRADE_DATA.DEPT;
          HUM_ELECTIVES.AREA4.COURSE_NUM:=
                    GRADE_DATA.COURSE;
          HUM_ELECTIVES.AREA4.CRSE_GRADE:=
                    GRADE_DATA.GRADE;
          HUM_ELECTIVES.AREA4.CRSE_HOURS:=
                    GRADE_DATA.HOURS;
          HUM_ELECTIVES.AREA4.LOC1:= GRADE_DATA.LOC1;
          HUM_ELECTIVES.AREA4.LOC2:= GRADE_DATA.LOC2;
          HUM_ELECTIVES.AREA4.LOC3:= GRADE_DATA.LOC3;
          HUM_ELECTIVES.AREA4.LOC4:= GRADE_DATA.LOC4;
          HUM_ELECTIVES.AREA4.LOC5:= GRADE_DATA.LOC5;
          HUM_ELECTIVES.AREA4.RPT:= GRADE_DATA.RPT;
          HUM_ELECTIVES.AREA4.SEMESTER:=
                    GRADE_DATA.SEMESTER;
          HUM_ELECTIVES.AREA4_TOT:=
                    TRUNC(STOR(GRADE_DATA.HOURS);
          END;
       IF LOCATION2 = '5' THEN
          BEGIN
          HUM_ELECTIVES.AREA5.DEPT_CODE:=
                        GRADE_DATA.DEPT;
          HUM_ELECTIVES.AREA5.COURSE_NUM:=
                        GRADE_DATA.COURSE;
          HUM_ELECTIVES.AREA5.CRSE_GRADE:=
                        GRADE_DATA.GRADE;
          HUM_ELECTIVES.AREA5.CRSE_HOURS:=
                        GRADE_DATA.HOURS;
          HUM_ELECTIVES.AREA5.LOC1:= GRADE_DATA.LOC1;
          HUM_ELECTIVES.AREA5.LOC2:= GRADE_DATA.LOC2;
          HUM_ELECTIVES.AREA5.LOC3:= GRADE_DATA.LOC3;
          HUM_ELECTIVES.AREA5.LOC4:= GRADE_DATA.LOC4;
          HUM_ELECTIVES.AREA5.LOC5:= GRADE_DATA.LOC5;
          HUM_ELECTIVES.AREA5.RPT:= GRADE_DATA.RPT;
          HUM_ELECTIVES.AREA5.SEMESTER:=
                        GRADE_DATA.SEMESTER;
          HUM_ELECTIVES.AREA5_TOT:=
              HUM_ELECTIVES.AREA5_TOT +
              TRUNC(STOR(GRADE_DATA.HOURS));
          END;
       END;
NUMBER:= STOR(GRADE_DATA.HOURS);
GRAND_TOTAL:= GRAND_TOTAL + TRUNC(NUMBER);
END;

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

PROCEDURE CLEAR_STORAGE ( VAR STORAGE_REC1 : TEMP_RECORD;
                          VAR STORAGE_REC2 : TEMP_RECORD);
```

```
            HUM_ELECTIVES.AREA5.LOC5:= GRADE_DATA.LOC5;
            HUM_ELECTIVES.AREA5.RPT:= GRADE_DATA.RPT;
            HUM_ELECTIVES.AREA5.SEMESTER:=
                        GRADE_DATA.SEMESTER;
            HUM_ELECTIVES.AREA5_TOT:=
                HUM_ELECTIVES.AREA5_TOT +
                TRUNC(STOR(GRADE_DATA.HOURS));
            END;
         END;
NUMBER:= STOR(GRADE_DATA.HOURS);
GRAND_TOTAL:= GRAND_TOTAL + TRUNC(NUMBER);
END;

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

PROCEDURE CLEAR_STORAGE ( VAR STORAGE_REC1 : TEMP_RECORD;
                          VAR STORAGE_REC2 : TEMP_RECORD);

(*THIS PROCEDURE CLEARS OUT THE TEMPORARY RECORDS*)


BEGIN
  STORAGE_REC1.COURSE_NUM:= '    ';
  STORAGE_REC1.CRSE_HOURS:= ' ';
  STORAGE_REC1.DEPT:= '    ';
  STORAGE_REC1.GRADE:= ' ';
  STORAGE_REC1.LOC1:= '    ';
  STORAGE_REC1.LOC2:= '    ';
  STORAGE_REC1.LOC3:= '    ';
  STORAGE_REC1.LOC4:= '    ';
  STORAGE_REC1.LOC5:= '    ';
  STORAGE_REC1.RPT:= ' ';
  STORAGE_REC1.SEMESTER:= '    ';
  STORAGE_REC2.COURSE_NUM:= '    ';
  STORAGE_REC2.CRSE_HOURS:= ' ';
  STORAGE_REC2.DEPT:= '    ';
  STORAGE_REC2.GRADE:= ' ';
  STORAGE_REC2.LOC1:= '    ';
  STORAGE_REC2.LOC2:= '    ';
  STORAGE_REC2.LOC3:= '    ';
  STORAGE_REC2.LOC4:= '    ';
  STORAGE_REC2.LOC5:= '    ';
  STORAGE_REC2.RPT:= ' ';
  STORAGE_REC2.SEMESTER:= '    ';
END;

(*********************************************************)
PROCEDURE MOVE_CLASS (VAR AREA_B            : HUMAN;
                      STOR_RECORD           : TEMP_RECORD;
                      LOCATION1             : STR1;
                      LOCATION2             : STR1);

(*THIS PROCEDURE MOVES THE CLASSES INTO THE PROPER*)
(*AREA OF THE RECORD*)


BEGIN
  IF LOCATION1 = "1" THEN
       BEGIN
       AREA_B.FINE_ARTS.COURSE_NUM:= STOR_RECORD.COURSE_NUM;
       AREA_B.FINE_ARTS.CRSE_HOURS:= STOR_RECORD.CRSE_HOURS;
       AREA_B.FINE_ARTS.DEPT_CODE:= STOR_RECORD.DEPT;
       AREA_B.FINE_ARTS.CRSE_GRADE:= STOR_RECORD.GRADE;
       AREA_B.FINE_ARTS.LOC1:= STOR_RECORD.LOC1;
       AREA_B.FINE_ARTS.LOC2:= STOR_RECORD.LOC2;
       AREA_B.FINE_ARTS.LOC3:= STOR_RECORD.LOC3;
       AREA_B.FINE_ARTS.LOC4:= STOR_RECORD.LOC4;
       AREA_B.FINE_ARTS.LOC5:= STOR_RECORD.LOC5;
       AREA_B.FINE_ARTS.RPT:= STOR_RECORD.RPT;
       AREA_B.FINE_ARTS.SEMESTER:= STOR_RECORD.SEMESTER;
       END;
  IF LOCATION1 = "2" THEN
       BEGIN
       AREA_B.LITERATURE.COURSE_NUM:= STOR_RECORD.COURSE_NUM;
       AREA_B.LITERATURE.CRSE_HOURS:= STOR_RECORD.CRSE_HOURS;
       AREA_B.LITERATURE.DEPT_CODE:= STOR_RECORD.DEPT;
       AREA_B.LITERATURE.CRSE_GRADE:= STOR_RECORD.GRADE;
       AREA_B.LITERATURE.LOC1:= STOR_RECORD.LOC1;
       AREA_B.LITERATURE.LOC2:= STOR_RECORD.LOC2;
       AREA_B.LITERATURE.LOC3:= STOR_RECORD.LOC3;
       AREA_B.LITERATURE.LOC4:= STOR_RECORD.LOC4;
       AREA_B.LITERATURE.LOC5:= STOR_RECORD.LOC5;
       AREA_B.LITERATURE.RPT:= STOR_RECORD.RPT;
```

```
    AREA_B.FINE_ARTS.LOC4:= STOR_RECORD.LOC4;
    AREA_B.FINE_ARTS.LOC5:= STOR_RECORD.LOC5;
    AREA_B.FINE_ARTS.RPT:= STOR_RECORD.RPT;
    AREA_B.FINE_ARTS.SEMESTER:= STOR_RECORD.SEMESTER;
    END;
IF LOCATION1 = "2" THEN
    BEGIN
    AREA_B.LITERATURE.COURSE_NUM:= STOR_RECORD.COURSE_NUM;
    AREA_B.LITERATURE.CRSE_HOURS:= STOR_RECORD.CRSE_HOURS;
    AREA_B.LITERATURE.DEPT_CODE:= STOR_RECORD.DEPT;
    AREA_B.LITERATURE.CRSE_GRADE:= STOR_RECORD.GRADE;
    AREA_B.LITERATURE.LOC1:= STOR_RECORD.LOC1;
    AREA_B.LITERATURE.LOC2:= STOR_RECORD.LOC2;
    AREA_B.LITERATURE.LOC3:= STOR_RECORD.LOC3;
    AREA_B.LITERATURE.LOC4:= STOR_RECORD.LOC4;
    AREA_B.LITERATURE.LOC5:= STOR_RECORD.LOC5;
    AREA_B.LITERATURE.RPT:= STOR_RECORD.RPT;
    AREA_B.LITERATURE.SEMESTER:= STOR_RECORD.SEMESTER;
    END;
IF LOCATION1 = "3" THEN
    BEGIN
    AREA_B.HISTORY.COURSE_NUM:= STOR_RECORD.COURSE_NUM;
    AREA_B.HISTORY.CRSE_HOURS:= STOR_RECORD.CRSE_HOURS;
    AREA_B.HISTORY.DEPT_CODE:= STOR_RECORD.DEPT;
    AREA_B.HISTORY.CRSE_GRADE:= STOR_RECORD.GRADE;
    AREA_B.HISTORY.LOC1:= STOR_RECORD.LOC1;
    AREA_B.HISTORY.LOC2:= STOR_RECORD.LOC2;
    AREA_B.HISTORY.LOC3:= STOR_RECORD.LOC3;
    AREA_B.HISTORY.LOC4:= STOR_RECORD.LOC4;
    AREA_B.HISTORY.LOC5:= STOR_RECORD.LOC5;
    AREA_B.HISTORY.RPT:= STOR_RECORD.RPT;
    AREA_B.HISTORY.SEMESTER:= STOR_RECORD.SEMESTER;
    END;
IF LOCATION1 = "4" THEN
    BEGIN
    AREA_B.PHIL_REL.COURSE_NUM:= STOR_RECORD.COURSE_NUM;
    AREA_B.PHIL_REL.CRSE_HOURS:= STOR_RECORD.CRSE_HOURS;
    AREA_B.PHIL_REL.DEPT_CODE:= STOR_RECORD.DEPT;
    AREA_B.PHIL_REL.CRSE_GRADE:= STOR_RECORD.GRADE;
    AREA_B.PHIL_REL.LOC1:= STOR_RECORD.LOC1;
    AREA_B.PHIL_REL.LOC2:= STOR_RECORD.LOC2;
    AREA_B.PHIL_REL.LOC3:= STOR_RECORD.LOC3;
    AREA_B.PHIL_REL.LOC4:= STOR_RECORD.LOC4;
    AREA_B.PHIL_REL.LOC5:= STOR_RECORD.LOC5;
    AREA_B.PHIL_REL.RPT:= STOR_RECORD.RPT;
    AREA_B.PHIL_REL.SEMESTER:= STOR_RECORD.SEMESTER;
    END;
IF LOCATION1 = "5" THEN
  BEGIN
  IF LOCATION2 = "1" THEN
    BEGIN
    AREA_B.HUM_ELECTIVES.AREA1.COURSE_NUM:=
                    STOR_RECORD.COURSE_NUM;
    AREA_B.HUM_ELECTIVES.AREA1.CRSE_HOURS:=
                    STOR_RECORD.CRSE_HOURS;
    AREA_B.HUM_ELECTIVES.AREA1.DEPT_CODE:=
                    STOR_RECORD.DEPT;
    AREA_B.HUM_ELECTIVES.AREA1.CRSE_GRADE:=
                    STOR_RECORD.GRADE;
    AREA_B.HUM_ELECTIVES.AREA1.LOC1:=
                    STOR_RECORD.LOC1;
    AREA_B.HUM_ELECTIVES.AREA1.LOC2:=
                    STOR_RECORD.LOC2;
    AREA_B.HUM_ELECTIVES.AREA1.LOC3:=
                    STOR_RECORD.LOC3;
    AREA_B.HUM_ELECTIVES.AREA1.LOC4:=
                    STOR_RECORD.LOC4;
    AREA_B.HUM_ELECTIVES.AREA1.LOC5:=
                    STOR_RECORD.LOC5;
    AREA_B.HUM_ELECTIVES.AREA1.RPT:=
                    STOR_RECORD.RPT;
    AREA_B.HUM_ELECTIVES.AREA1.SEMESTER:=
                    STOR_RECORD.SEMESTER;
    END;
  IF LOCATION2 = "2" THEN
    BEGIN
    AREA_B.HUM_ELECTIVES.AREA2.COURSE_NUM:=
                    STOR_RECORD.COURSE_NUM;
    AREA_B.HUM_ELECTIVES.AREA2.CRSE_HOURS:=
                    STOR_RECORD.CRSE_HOURS;
    AREA_B.HUM_ELECTIVES.AREA2.DEPT_CODE:=
                    STOR_RECORD.DEPT;
    AREA_B.HUM_ELECTIVES.AREA2.CRSE_GRADE:=
                    STOR_RECORD.GRADE;
```

```
                    STOR_RECORD.LOC4;
    AREA_B.HUM_ELECTIVES.AREA1.LOC5:=
                    STOR_RECORD.LOC5;
    AREA_B.HUM_ELECTIVES.AREA1.RPT:=
                    STOR_RECORD.RPT;
    AREA_B.HUM_ELECTIVES.AREA1.SEMESTER:=
                    STOR_RECORD.SEMESTER;
    END;
  IF LOCATION2 = "2" THEN
    BEGIN
    AREA_B.HUM_ELECTIVES.AREA2.COURSE_NUM:=
                    STOR_RECORD.COURSE_NUM;
    AREA_B.HUM_ELECTIVES.AREA2.CRSE_HOURS:=
                    STOR_RECORD.CRSE_HOURS;
    AREA_B.HUM_ELECTIVES.AREA2.DEPT_CODE:=
                    STOR_RECORD.DEPT;
    AREA_B.HUM_ELECTIVES.AREA2.CRSE_GRADE:=
                    STOR_RECORD.GRADE;
    AREA_B.HUM_ELECTIVES.AREA2.LOC1:=
                    STOR_RECORD.LOC1;
    AREA_B.HUM_ELECTIVES.AREA2.LOC2:=
                    STOR_RECORD.LOC2;
    AREA_B.HUM_ELECTIVES.AREA2.LOC3:=
                    STOR_RECORD.LOC3;
    AREA_B.HUM_ELECTIVES.AREA2.LOC4:=
                    STOR_RECORD.LOC4;
    AREA_B.HUM_ELECTIVES.AREA2.LOC5:=
                    STOR_RECORD.LOC5;
    AREA_B.HUM_ELECTIVES.AREA2.RPT:=
                    STOR_RECORD.RPT;
    AREA_B.HUM_ELECTIVES.AREA2.SEMESTER:=
                    STOR_RECORD.SEMESTER;
    END;
  IF LOCATION2 = "3" THEN
    BEGIN
    AREA_B.HUM_ELECTIVES.AREA3.COURSE_NUM:=
                  STOR_RECORD.COURSE_NUM;
    AREA_B.HUM_ELECTIVES.AREA3.CRSE_HOURS:=
                  STOR_RECORD.CRSE_HOURS;
    AREA_B.HUM_ELECTIVES.AREA3.DEPT_CODE:=
                  STOR_RECORD.DEPT;
    AREA_B.HUM_ELECTIVES.AREA3.CRSE_GRADE:=
                  STOR_RECORD.GRADE;
    AREA_B.HUM_ELECTIVES.AREA3.LOC1:=
                  STOR_RECORD.LOC1;
    AREA_B.HUM_ELECTIVES.AREA3.LOC2:=
                  STOR_RECORD.LOC2;
    AREA_B.HUM_ELECTIVES.AREA3.LOC3:=
                  STOR_RECORD.LOC3;
    AREA_B.HUM_ELECTIVES.AREA3.LOC4:=
                  STOR_RECORD.LOC4;
    AREA_B.HUM_ELECTIVES.AREA3.LOC5:=
                  STOR_RECORD.LOC5;
    AREA_B.HUM_ELECTIVES.AREA3.RPT:=
                  STOR_RECORD.RPT;
    AREA_B.HUM_ELECTIVES.AREA3.SEMESTER:=
    STOR_RECORD.SEMESTER;
    END;
  IF LOCATION2 = "4" THEN
    BEGIN
    AREA_B.HUM_ELECTIVES.AREA4.COURSE_NUM:=
                STOR_RECORD.COURSE_NUM;
    AREA_B.HUM_ELECTIVES.AREA4.CRSE_HOURS:=
                STOR_RECORD.CRSE_HOURS;
    AREA_B.HUM_ELECTIVES.AREA4.DEPT_CODE:=
                STOR_RECORD.DEPT;
    AREA_B.HUM_ELECTIVES.AREA4.CRSE_GRADE:=
                STOR_RECORD.GRADE;
    AREA_B.HUM_ELECTIVES.AREA4.LOC1:=
                STOR_RECORD.LOC1;
    AREA_B.HUM_ELECTIVES.AREA4.LOC2:=
                STOR_RECORD.LOC2;
    AREA_B.HUM_ELECTIVES.AREA4.LOC3:=
                STOR_RECORD.LOC3;
    AREA_B.HUM_ELECTIVES.AREA4.LOC4:=
                STOR_RECORD.LOC4;
    AREA_B.HUM_ELECTIVES.AREA4.LOC5:=
                STOR_RECORD.LOC5;
    AREA_B.HUM_ELECTIVES.AREA4.RPT:=
                STOR_RECORD.RPT;
    AREA_B.HUM_ELECTIVES.AREA4.SEMESTER:=
                STOR_RECORD.SEMESTER;
    END;
  IF LOCATION2 = "5" THEN
```

```
      AREA_B.HUM_ELECTIVES.AREA4.CRSE_GRADE:=
                  STOR_RECORD.GRADE;
      AREA_B.HUM_ELECTIVES.AREA4.LOC1:=
                  STOR_RECORD.LOC1;
      AREA_B.HUM_ELECTIVES.AREA4.LOC2:=
                  STOR_RECORD.LOC2;
      AREA_B.HUM_ELECTIVES.AREA4.LOC3:=
                  STOR_RECORD.LOC3;
      AREA_B.HUM_ELECTIVES.AREA4.LOC4:=
                  STOR_RECORD.LOC4;
      AREA_B.HUM_ELECTIVES.AREA4.LOC5:=
                  STOR_RECORD.LOC5;
      AREA_B.HUM_ELECTIVES.AREA4.RPT:=
                  STOR_RECORD.RPT;
      AREA_B.HUM_ELECTIVES.AREA4.SEMESTER:=
                  STOR_RECORD.SEMESTER;
      END;
    IF LOCATION2 = "5" THEN
      BEGIN
      AREA_B.HUM_ELECTIVES.AREA5.COURSE_NUM:=
                  STOR_RECORD.COURSE_NUM;
      AREA_B.HUM_ELECTIVES.AREA5.CRSE_HOURS:=
                  STOR_RECORD.CRSE_HOURS;
      AREA_B.HUM_ELECTIVES.AREA5.DEPT_CODE:=
                   STOR_RECORD.DEPT;
      AREA_B.HUM_ELECTIVES.AREA5.CRSE_GRADE:=
                  STOR_RECORD.GRADE;
      AREA_B.HUM_ELECTIVES.AREA5.LOC1:=
                  STOR_RECORD.LOC1;
      AREA_B.HUM_ELECTIVES.AREA5.LOC2:=
                   STOR_RECORD.LOC2;
      AREA_B.HUM_ELECTIVES.AREA5.LOC3:=
                  STOR_RECORD.LOC3;
      AREA_B.HUM_ELECTIVES.AREA5.LOC4:=
                   STOR_RECORD.LOC4;
      AREA_B.HUM_ELECTIVES.AREA5.LOC5:=
                  STOR_RECORD.LOC5;
      AREA_B.HUM_ELECTIVES.AREA5.RPT:=
                  STOR_RECORD.RPT;
      AREA_B.HUM_ELECTIVES.AREA5.SEMESTER:=
                  STOR_RECORD.SEMESTER;
      END;
    END;
END;

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

PROCEDURE DELETE_CLASS (VAR AREA_B              :HUMAN;
                        OLD_LOC2                :STR1;
                        OLD_LOC3                :STR1);

(*THIS PROCEDURE DELETES THE CLASS FROM THE RECORD*)


CONST
  MIN_HOURS = 3;


BEGIN
  IF OLD_LOC2 = "1" THEN
    BEGIN
    AREA_B.ARTS_TOTAL:= AREA_B.ARTS_TOTAL - MIN_HOURS;
    AREA_B.FINE_ARTS.COURSE_NUM:= '    ';
    AREA_B.FINE_ARTS.CRSE_HOURS:= '   ';
    AREA_B.FINE_ARTS.DEP_CODE:= '    ';
    AREA_B.FINE_ARTS.CRSE_GRADE:='   ';
    AREA_B.FINE_ARTS.LOC1:= '    ';
    AREA_B.FINE_ARTS.LOC2:= '    ';
    AREA_B.FINE_ARTS.LOC3:= '    ';
    AREA_B.FINE_ARTS.LOC4:= '    ';
    AREA_B.FINE_ARTS.LOC5:= '    ';
    AREA_B.FINE_ARTS.SEMESTER:= '    ';
    END;
  IF OLD_LOC2 = "2" THEN
    BEGIN
    AREA_B.LIT_TOTAL:= AREA_B.LIT_TOTAL - MIN_HOURS;
    AREA_B.LITERATURE.COURSE_NUM:= '    ';
    AREA_B.LITERATURE.CRSE_HOURS:= '   ';
    AREA_B.LITERATURE.DEP_CODE:= '    ';
    AREA_B.LITERATURE.CRSE_GRADE:='   ';
    AREA_B.LITERATURE.LOC1:= '    ';
    AREA_B.LITERATURE.LOC2:= '    ';
    AREA_B.LITERATURE.LOC3:= '    ';
```

```
    AREA_B.FINE_ARTS.LOC1:= '    ';
    AREA_B.FINE_ARTS.LOC2:= '    ';
    AREA_B.FINE_ARTS.LOC3:= '    ';
    AREA_B.FINE_ARTS.LOC4:= '    ';
    AREA_B.FINE_ARTS.LOC5:= '    ';
    AREA_B.FINE_ARTS.SEMESTER:= '    ';
    END;
  IF OLD_LOC2 = "2" THEN
    BEGIN
    AREA_B.LIT_TOTAL:= AREA_B.LIT_TOTAL - MIN_HOURS;
    AREA_B.LITERATURE.COURSE_NUM:= '    ';
    AREA_B.LITERATURE.CRSE_HOURS:= '   ';
    AREA_B.LITERATURE.DEP_CODE:= '    ';
    AREA_B.LITERATURE.CRSE_GRADE:='   ';
    AREA_B.LITERATURE.LOC1:= '    ';
    AREA_B.LITERATURE.LOC2:= '    ';
    AREA_B.LITERATURE.LOC3:= '    ';
    AREA_B.LITERATURE.LOC4:= '    ';
    AREA_B.LITERATURE.LOC5:= '    ';
    AREA_B.LITERATURE.SEMESTER:= '    ';
    END;
  IF OLD_LOC2 = "3" THEN
    BEGIN
    AREA_B.HISTORY_TOTAL:= AREA_B.HISTORY_TOTAL - MIN_HOURS;
    AREA_B.HISTORY.COURSE_NUM:= '    ';
    AREA_B.HISTORY.CRSE_HOURS:= '   ';
    AREA_B.HISTORY.DEP_CODE:= '    ';
    AREA_B.HISTORY.CRSE_GRADE:='   ';
    AREA_B.HISTORY.LOC1:= '    ';
    AREA_B.HISTORY.LOC2:= '    ';
    AREA_B.HISTORY.LOC3:= '    ';
    AREA_B.HISTORY.LOC4:= '    ';
    AREA_B.HISTORY.LOC5:= '    ';
    AREA_B.HISTORY.SEMESTER:= '    ';
    END;
  IF OLD_LOC2 = "4" THEN
    BEGIN
    AREA_B.PHIL_REL_TOTAL:= AREA_B.PHIL_REL_TOTAL - MIN_HOURS;
    AREA_B.PHIL_REL.COURSE_NUM:= '    ';
    AREA_B.PHIL_REL.CRSE_HOURS:= '   ';
    AREA_B.PHIL_REL.DEP_CODE:= '    ';
    AREA_B.PHIL_REL.CRSE_GRADE:='   ';
    AREA_B.PHIL_REL.LOC1:= '    ';
    AREA_B.PHIL_REL.LOC2:= '    ';
    AREA_B.PHIL_REL.LOC3:= '    ';
    AREA_B.PHIL_REL.LOC4:= '    ';
    AREA_B.PHIL_REL.LOC5:= '    ';
    AREA_B.PHIL_REL.SEMESTER:= '    ';
    END;
  IF OLD_LOC2 = "5" THEN
    BEGIN
      AREA_B.ELEC_TOTAL:= AREA_B.ELEC_TOTAL - MIN_HOURS;
      IF OLD_LOC3 = "1" THEN
        BEGIN
        AREA_B.HUM_ELECTIVES.AREA1_TOT:=
             AREA_B.HUM_ELECTIVES.AREA1_TOT - MIN_HOURS;
        AREA_B.HUM_ELECTIVES.AREA1.COURSE_NUM:= '    ';
        AREA_B.HUM_ELECTIVES.AREA1.CRSE_HOURS:= '   ';
        AREA_B.HUM_ELECTIVES.AREA1.DEP_CODE:= '    ';
        AREA_B.HUM_ELECTIVES.AREA1.CRSE_GRADE:='   ';
        AREA_B.HUM_ELECTIVES.AREA1.LOC1:= '    ';
        AREA_B.HUM_ELECTIVES.AREA1.LOC2:= '    ';
        AREA_B.HUM_ELECTIVES.AREA1.LOC3:= '    ';
        AREA_B.HUM_ELECTIVES.AREA1.LOC4:= '    ';
        AREA_B.HUM_ELECTIVES.AREA1.LOC5:= '    ';
        AREA_B.HUM_ELECTIVES.AREA1.SEMESTER:= '    ';
        END;
      IF OLD_LOC3 = "2" THEN
        BEGIN
        AREA_B.HUM_ELECTIVES.AREA2_TOT:=
             AREA_B.HUM_ELECTIVES.AREA2_TOT - MIN_HOURS;
        AREA_B.HUM_ELECTIVES.AREA2.COURSE_NUM:= '    ';
        AREA_B.HUM_ELECTIVES.AREA2.CRSE_HOURS:= '   ';
        AREA_B.HUM_ELECTIVES.AREA2.DEP_CODE:= '    ';
        AREA_B.HUM_ELECTIVES.AREA2.CRSE_GRADE:='   ';
        AREA_B.HUM_ELECTIVES.AREA2.LOC1:= '    ';
        AREA_B.HUM_ELECTIVES.AREA2.LOC2:= '    ';
        AREA_B.HUM_ELECTIVES.AREA2.LOC3:= '    ';
        AREA_B.HUM_ELECTIVES.AREA2.LOC4:= '    ';
        AREA_B.HUM_ELECTIVES.AREA2.LOC5:= '    ';
        AREA_B.HUM_ELECTIVES.AREA2.SEMESTER:= '    ';
        END;
      IF OLD_LOC3 = "3" THEN
        BEGIN
```

```
        END;
      IF OLD_LOC3 = "2" THEN
        BEGIN
        AREA_B.HUM_ELECTIVES.AREA2_TOT:=
              AREA_B.HUM_ELECTIVES.AREA2_TOT - MIN_HOURS;
        AREA_B.HUM_ELECTIVES.AREA2.COURSE_NUM:= '    ';
        AREA_B.HUM_ELECTIVES.AREA2.CRSE_HOURS:= '   ';
        AREA_B.HUM_ELECTIVES.AREA2.DEP_CODE:= '    ';
        AREA_B.HUM_ELECTIVES.AREA2.CRSE_GRADE:='   ';
        AREA_B.HUM_ELECTIVES.AREA2.LOC1:= '    ';
        AREA_B.HUM_ELECTIVES.AREA2.LOC2:= '    ';
        AREA_B.HUM_ELECTIVES.AREA2.LOC3:= '    ';
        AREA_B.HUM_ELECTIVES.AREA2.LOC4:= '    ';
        AREA_B.HUM_ELECTIVES.AREA2.LOC5:= '    ';
        AREA_B.HUM_ELECTIVES.AREA2.SEMESTER:= '    ';
        END;
      IF OLD_LOC3 = "3" THEN
        BEGIN
        AREA_B.HUM_ELECTIVES.AREA3_TOT:=
              AREA_B.HUM_ELECTIVES.AREA3_TOT - MIN_HOURS;
        AREA_B.HUM_ELECTIVES.AREA3.COURSE_NUM:= '    ';
        AREA_B.HUM_ELECTIVES.AREA3.CRSE_HOURS:= '   ';
        AREA_B.HUM_ELECTIVES.AREA3.DEP_CODE:= '    ';
        AREA_B.HUM_ELECTIVES.AREA3.CRSE_GRADE:='   ';
        AREA_B.HUM_ELECTIVES.AREA3.LOC1:= '    ';
        AREA_B.HUM_ELECTIVES.AREA3.LOC2:= '    ';
        AREA_B.HUM_ELECTIVES.AREA3.LOC3:= '    ';
        AREA_B.HUM_ELECTIVES.AREA3.LOC4:= '    ';
        AREA_B.HUM_ELECTIVES.AREA3.LOC5:= '    ';
        AREA_B.HUM_ELECTIVES.AREA3.SEMESTER:= '    ';
        END;
      IF OLD_LOC3 = "4" THEN
        BEGIN
        AREA_B.HUM_ELECTIVES.AREA4_TOT:=
                AREA_B.HUM_ELECTIVES.AREA4_TOT - MIN_HOURS;
        AREA_B.HUM_ELECTIVES.AREA4.COURSE_NUM:= '    ';
        AREA_B.HUM_ELECTIVES.AREA4.CRSE_HOURS:= '   ';
        AREA_B.HUM_ELECTIVES.AREA4.DEP_CODE:= '    ';
        AREA_B.HUM_ELECTIVES.AREA4.CRSE_GRADE:='   ';
        AREA_B.HUM_ELECTIVES.AREA4.LOC1:= '    ';
        AREA_B.HUM_ELECTIVES.AREA4.LOC2:= '    ';
        AREA_B.HUM_ELECTIVES.AREA4.LOC3:= '    ';
        AREA_B.HUM_ELECTIVES.AREA4.LOC4:= '    ';
        AREA_B.HUM_ELECTIVES.AREA4.LOC5:= '    ';
        AREA_B.HUM_ELECTIVES.AREA4.SEMESTER:= '    ';
        END;
      IF OLD_LOC3 = "5" THEN
        BEGIN
        AREA_B.HUM_ELECTIVES.AREA5_TOT:=
             AREA_B.HUM_ELECTIVES.AREA5_TOT - MIN_HOURS;
        AREA_B.HUM_ELECTIVES.AREA5.COURSE_NUM:= '    ';
        AREA_B.HUM_ELECTIVES.AREA5.CRSE_HOURS:= '   ';
        AREA_B.HUM_ELECTIVES.AREA5.DEP_CODE:= '    ';
        AREA_B.HUM_ELECTIVES.AREA5.CRSE_GRADE:='   ';
        AREA_B.HUM_ELECTIVES.AREA5.LOC1:= '    ';
        AREA_B.HUM_ELECTIVES.AREA5.LOC2:= '    ';
        AREA_B.HUM_ELECTIVES.AREA5.LOC3:= '    ';
        AREA_B.HUM_ELECTIVES.AREA5.LOC4:= '    ';
        AREA_B.HUM_ELECTIVES.AREA5.LOC5:= '    ';
        AREA_B.HUM_ELECTIVES.AREA5.SEMESTER:= '    ';
        END;
    END;
  HUMAN_TOTAL:= HUMAN_TOTAL -
              TRUNC(STOR(AREA_TITLE.CRSE_HOURS));
END;
(*******************************************************)

PROCEDURE CHECK_SPECIAL (VAR PLACED : BOOLEAN;
                         HUM_LOC2 : CHAR;
                         HUM_LOC3 : CHAR;
                         STOR_RECORD : TEMP_RECORD;
                         GRADE_DATA: CLASS_REC;
                         AREA_B: HUMAN);

(*THIS PROCEDURE CHECKS TO SEE IF THE NEW COURSE IS A*)
(*LANGUAGE COURSE OR A CSP COURSE.  IF IT IS EITHER *)
(*ONE OF THOSE OPTIONS, THE NEW COURSE IS PLACED *)
(*INTO THE RECORD*)


CONST
  MAX_HOURS = 6;

BEGIN
```

```
PROCEDURE CHECK_SPECIAL (VAR PLACED : BOOLEAN;
                         HUM_LOC2 : CHAR;
                         HUM_LOC3 : CHAR;
                         STOR_RECORD : TEMP_RECORD;
                         GRADE_DATA: CLASS_REC;
                         AREA_B: HUMAN);

(*THIS PROCEDURE CHECKS TO SEE IF THE NEW COURSE IS A*)
(*LANGUAGE COURSE OR A CSP COURSE.  IF IT IS EITHER *)
(*ONE OF THOSE OPTIONS, THE NEW COURSE IS PLACED *)
(*INTO THE RECORD*)


CONST
  MAX_HOURS = 6;

BEGIN
  IF STOR_RECORD.DEPT = 'FRH','GER', OR 'SPN' THEN
    IF GRADE_DATA.DEPT = 'FRH', 'GER', OR 'SPN' THEN
      IF AREA_B.ELEC_TOTAL < MAX_HOURS THEN
        BEGIN
        INSERT_CLASS (AREA_B,HUM_LOC2,HUM_LOC3,GRADE_DATA,
                      GRAND_TOTAL);
        PLACED:= TRUE;
        END;
  ELSE
    IF STOR_RECORD.DEPT = 'CSP' THEN
      IF GRADE_DATA.DEPT = 'CSP' THEN
        IF AREA_B.ELEC_TOTAL < MAX_HOURS THEN
          BEGIN
          INSERT_CLASS (REPORT.HUMANITIES,HUM_LOC2,HUM_LOC3,
                        GRADE_DATA,GRAND_TOTAL);
          PLACED:= TRUE;
          END;
END;

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



BEGIN
 FILLED := FALSE;
 PLACED:= FALSE;
 MORE:= FALSE;
 STOR_FILLED:= FALSE;
 GRADE_DATA.LOC1:= TEMP_LOC;
 CONVERT_LOCATION(TEMP_LOC,HUM_LOC1,HUM_LOC2,HUM_LOC3);
 NEW_LOC1A:= HUM_LOC2;
 NEW_LOC1B:= HUM_LOC3;
 CHECK_LOCATION (FILLED,STORAGE_REC1,HUM_LOC2,HUM_LOC3,
                 AREA_B,STORAGE_REC2);
 IF FILLED THEN
    BEGIN
    CHECK_SPECIAL (PLACED, HUM_LOC2,HUM_LOC3,STOR_RECORD,
                   GRADE_DATA,AREA_B);
    IF NOT(PLACED) THEN
      BEGIN
      CHECK_NUMBER (GRADE_DATA.RPT, MORE);
      IF MORE THEN
        BEGIN
        FILLED:= FALSE;
        TEMP_LOC:= GRADE_DATA.LOC2;
        CONVERT_LOCATION (TEMP_LOC,HUM_LOC1, HUM_LOC2, ,
                          HUM_LOC3);
        NEW_LOC2A:= HUM_LOC2;
        NEW_LOC2B:= HUM_LOC3;
        CHECK_LOCATION (FILLED,STORAGE_REC2,HUM_LOC2,
                        HUM_LOC3,AREA_B,STOR_FILLED);
        IF FILLED THEN
          BEGIN
          MORE:= FALSE;
          CHECK_NUMBER (STORAGE_REC1.RPT, MORE);
          IF MORE THEN
            BEGIN
            TEMP_LOC:= STORAGE_REC1.LOC2;
            CONVERT_LOCATION (TEMP_LOC,HUM_LOC1,HUM_LOC2,
                              HUM_LOC3);
            CHECK_LOCATION (FILLED,STORAGE_REC1,HUM_LOC2,
                            HUM_LOC3,AREA_B,STOR_FILLED);
            IF FILLED THEN
              BEGIN
              MORE:= FALSE;
              CHECK_NUMBER(STORAGE_REC2.RPT, MORE);
```

```
NEW_LOC2A:= HUM_LOC2;
NEW_LOC2B:= HUM_LOC3;
CHECK_LOCATION (FILLED,STORAGE_REC2,HUM_LOC2,
                HUM_LOC3,AREA_B,STOR_FILLED);
IF FILLED THEN
  BEGIN
  MORE:= FALSE;
  CHECK_NUMBER (STORAGE_REC1.RPT, MORE);
  IF MORE THEN
    BEGIN
    TEMP_LOC:= STORAGE_REC1.LOC2;
    CONVERT_LOCATION (TEMP_LOC,HUM_LOC1,HUM_LOC2,
                      HUM_LOC3);
    CHECK_LOCATION (FILLED,STORAGE_REC1,HUM_LOC2,
                    HUM_LOC3,AREA_B,STOR_FILLED);
    IF FILLED THEN
      BEGIN
      MORE:= FALSE;
      CHECK_NUMBER(STORAGE_REC2.RPT, MORE);
      IF MORE THEN
        BEGIN
        TEMP_LOC:= STORAGE_REC2.LOC2;
        CONVERT_LOCATION ( TEMP_LOC, HUM_LOC1,
                           HUM_LOC2,HUM_LOC3);
        FILLED:= FALSE;
        CHECK_LOCATION (FILLED, STORAGE_REC2,
             HUM_LOC2,HUM_LOC3,AREA_B,STOR_FILLED);
        IF FILLED THEN
          CLEAR_STORAGE (STORAGE_REC1,
                         STORAGE_REC2);
        ELSE
          BEGIN
            MOVE_CLASS (AREA_B,STORAGE_REC2,
                        HUM_LOC2,HUM_LOC3);
            TEMP_LOC:= STORAGE_REC2.LOC1
            CONVERT_LOCATION (TEMP_LOC, OLD_LOC1,
                        OLD_LOC2,OLD_LOC3);
            DELETE_CLASS (AREA_B,OLD_LOC2,OLD_LOC3);
            INSERT_CLASS (AREA_B,NEW_LOC2A,
                NEW_LOC2B,GRADE_DATA,GRAND_TOTAL);
            CLEAR_STORAGE ( STORAGE_REC1,
                            STORAGE_REC2);
          END;
        END
      ELSE
        CLEAR_STORAGE ( STORAGE_REC1, STORAGE_REC2);
      END
    ELSE
      BEGIN
      MOVE_CLASS (AREA_B,HUM_LOC2,HUM_LOC3,
                  STORAGE_REC1);
      TEMP_LOC:= STORAGE_REC1.LOC1;
      CONVERT_LOCATION (TEMP_LOC,OLD_LOC1, OLD_LOC2,
                        OLD_LOC3);
      DELETE_CLASS (AREA_B,OLD_LOC2,OLD_LOC3);
      INSERT_CLASS (AREA_B,NEW_LOC1A,NEW_LOC1B,
                    GRADE_DATA,GRAND_TOTAL);
      CLEAR_STORAGE(STORAGE_REC1, STORAGE_REC2);
      END;
  END
  ELSE
    BEGIN
    MORE:= FALSE;
    CHECK_NUMBER (STORAGE_REC2.RPT, MORE);
    IF MORE THEN
      BEGIN
      FILLED:= FALSE;
      TEMP_LOC:= STORAGE_REC2.LOC2;
      CONVERT_LOCATION(TEMP_LOC,HUM_LOC1,HUM_LOC2,
                       HUM_LOC3);
      CHECK_LOCATION (FILLED, STORAGE_REC2,HUM_LOC2,
                      HUM_LOC3,AREA_B,STOR_FILLED);
      IF FILLED THEN
         CLEAR_STORAGE ( STORAGE_REC1,
                         STORAGE_REC2);
      ELSE
        BEGIN
        MOVE_CLASS (AREA_B,HUM_LOC2, HUM_LOC3,
                    STORAGE_REC2);
        TEMP_LOC:= STORAGE_REC2.LOC1;
        CONVERT_LOCATION ( TEMP_LOC, OLD_LOC1,
                           OLD_LOC2,OLD_LOC3);
        DELETE_CLASS (AREA_B,OLD_LOC2, OLD_LOC3);
        INSERT_CLASS (AREA_B,NEW_LOC2A, NEW_LOC2B,
```

```
        FILLED:= FALSE;
        TEMP_LOC:= STORAGE_REC2.LOC2;
        CONVERT_LOCATION(TEMP_LOC,HUM_LOC1,HUM_LOC2,
                         HUM_LOC3);
        CHECK_LOCATION (FILLED, STORAGE_REC2,HUM_LOC2,
                        HUM_LOC3,AREA_B,STOR_FILLED);
        IF FILLED THEN
           CLEAR_STORAGE ( STORAGE_REC1,
                           STORAGE_REC2);
        ELSE
          BEGIN
          MOVE_CLASS (AREA_B,HUM_LOC2, HUM_LOC3,
                      STORAGE_REC2);
          TEMP_LOC:= STORAGE_REC2.LOC1;
          CONVERT_LOCATION ( TEMP_LOC, OLD_LOC1,
                             OLD_LOC2,OLD_LOC3);
          DELETE_CLASS (AREA_B,OLD_LOC2, OLD_LOC3);
          INSERT_CLASS (AREA_B,NEW_LOC2A, NEW_LOC2B,
                        GRADE_DATA,GRAND_TOTAL);
          CLEAR_STORAGE (STORAGE_REC1, STORAGE_REC2);
          END;
        END
      ELSE
        CLEAR_STORAGE (STORAGE_REC1, STORAGE_REC2);
      END
    ELSE
    INSERT_CLASS (AREA_B, HUM_LOC2, HUM_LOC3,
                  GRADE_DATA,GRAND_TOTAL);
END
ELSE
 BEGIN
 FILLED:= FALSE;
 TEMP_LOC:= GRADE_DATA.LOC2;
 CONVERT_LOCATION (TEMP_LOC,HUM_LOC1, HUM_LOC2,
                   HUM_LOC3);
 NEW_LOC2A:= HUM_LOC2;
 NEW_LOC2B:= HUM_LOC3;
 CHECK_LOCATION (FILLED,STORAGE_REC2,HUM_LOC2,
                 HUM_LOC3,AREA_B,STOR_FILLED);
 IF FILLED THEN
    BEGIN
    MORE:= FALSE;
    CHECK_NUMBER (STORAGE_REC1.RPT, MORE);
    IF MORE THEN
       BEGIN
       TEMP_LOC:= STORAGE_REC1.LOC2;
       CONVERT_LOCATION (TEMP_LOC,HUM_LOC1,HUM_LOC2,
                         HUM_LOC3);
       CHECK_LOCATION (FILLED,STORAGE_REC1,HUM_LOC2,
                       HUM_LOC3,AREA_B,STOR_FILLED);
       IF FILLED THEN
         BEGIN
         MORE:= FALSE;
         CHECK_NUMBER(STORAGE_REC2.RPT, MORE);
         IF MORE THEN
           BEGIN
           TEMP_LOC:= STORAGE_REC2.LOC2;
           CONVERT_LOCATION ( TEMP_LOC, HUM_LOC1,
                              HUM_LOC2,HUM_LOC3);
           FILLED:= FALSE;
           CHECK_LOCATION (FILLED, STORAGE_REC2,
                  HUM_LOC2,HUM_LOC3,AREA_B,
                  STOR_FILLED);
           IF FILLED THEN
             CLEAR_STORAGE (STORAGE_REC1,
                            STORAGE_REC2);
           ELSE
             BEGIN
             MOVE_CLASS (AREA_B,STORAGE_REC2,
                         HUM_LOC2,HUM_LOC3);
             TEMP_LOC:= STORAGE_REC2.LOC1;
             CONVERT_LOCATION (TEMP_LOC, OLD_LOC1,
                               OLD_LOC2,OLD_LOC3);
             DELETE_CLASS (AREA_B,OLD_LOC2,
                           OLD_LOC3);
             INSERT_CLASS (AREA_B,NEW_LOC2A,
                 NEW_LOC2B,GRADE_DATA,GRAND_TOTAL);
             CLEAR_STORAGE ( STORAGE_REC1,
                             STORAGE_REC2);
             END;
           END
         ELSE
             CLEAR_STORAGE ( STORAGE_REC1,
```

```
                                        STORAGE_REC2);
                    ELSE
                      BEGIN
                      MOVE_CLASS (AREA_B,STORAGE_REC2,
                                  HUM_LOC2,HUM_LOC3);
                      TEMP_LOC:= STORAGE_REC2.LOC1;
                      CONVERT_LOCATION (TEMP_LOC, OLD_LOC1,
                                        OLD_LOC2,OLD_LOC3);
                      DELETE_CLASS (AREA_B,OLD_LOC2,
                                    OLD_LOC3);
                      INSERT_CLASS (AREA_B,NEW_LOC2A,
                          NEW_LOC2B,GRADE_DATA,GRAND_TOTAL);
                      CLEAR_STORAGE ( STORAGE_REC1,
                                      STORAGE_REC2);
                      END;
                    END
                  ELSE
                      CLEAR_STORAGE ( STORAGE_REC1,
                                      STORAGE_REC2);
              END
            ELSE
              BEGIN
              MOVE_CLASS (AREA_B,HUM_LOC2,HUM_LOC3,
                          STORAGE_REC1);
              TEMP_LOC:= STORAGE_REC1.LOC1;
              CONVERT_LOCATION (TEMP_LOC,OLD_LOC1, OLD_LOC2,
                                OLD_LOC3);
              DELETE_CLASS (AREA_B,OLD_LOC2,OLD_LOC3);
              INSERT_CLASS (AREA_B,NEW_LOC1A,NEW_LOC1B,
                            GRADE_DATA,GRAND_TOTAL);
              CLEAR_STORAGE(STORAGE_REC1, STORAGE_REC2);
              END;
          ELSE
            BEGIN
            MORE:= FALSE;
            CHECK_NUMBER (STORAGE_REC2.RPT, MORE);
            IF MORE THEN
              BEGIN
              FILLED:= FALSE;
              TEMP_LOC:= STORAGE_REC2.LOC2;
              CONVERT_LOCATION(TEMP_LOC,HUM_LOC1,HUM_LOC2,
                               HUM_LOC3);
              CHECK_LOCATION (FILLED, STORAGE_REC2,HUM_LOC2,
                              HUM_LOC3,AREA_B,STOR_FILLED);
              IF FILLED THEN
                 CLEAR_STORAGE ( STORAGE_REC1, STORAGE_REC2)
              ELSE
                BEGIN
                MOVE_CLASS (AREA_B,HUM_LOC2, HUM_LOC3,
                            STORAGE_REC2);
                TEMP_LOC:= STORAGE_REC2.LOC1;
                CONVERT_LOCATION ( TEMP_LOC, OLD_LOC1,
                                   OLD_LOC2,OLD_LOC3);
                DELETE_CLASS (AREA_B,OLD_LOC2, OLD_LOC3);
                INSERT_CLASS (AREA_B,NEW_LOC2A, NEW_LOC2B,
                              GRADE_DATA,GRAND_TOTAL);
                CLEAR_STORAGE (STORAGE_REC1, STORAGE_REC2);
                END;
              END
            ELSE
              CLEAR_STORAGE (STORAGE_REC1, STORAGE_REC2);
            END;
        END
        ELSE
          INSERT_CLASS (AREA_B, HUM_LOC2, HUM_LOC3,
                        GRADE_DATA,GRAND_TOTAL);
      END;
  END
    ELSE
      INSERT_CLASS ( AREA_B,HUM_LOC2,HUM_LOC3,GRADE_DATA,
                     GRAND_TOTAL);

END;
```

```
END
    ELSE
        INSERT_CLASS ( AREA_B,HUM_LOC2,HUM_LOC3,GRADE_DATA,
                       GRAND_TOTAL);

END;
```

```
PROCEDURE PART_C(GRADE : CLASS_REC; VAR SOC : SOCIAL;
                 VAR GRAND_TOTAL : INTEGER; COUNT : INTEGER;
                 VAR EXTRA_BASIC : BASIC_ARRAY);
CONST
  SOC_SCI_MAX := 12;
  ECON_MAX := 3;
  GEO_MAX := 3;
  POL_MAX := 3;
  PSY_MAX := 3;
  SOC_MAX := 3;

TYPE
  STR1 = PACKED ARRAY[1..1] OF CHAR;

VAR
  AREA : STR1;
  FOUND : BOOLEAN;


PROCEDURE SELECT_AREA(AREA : STR1; SOC : SOCIAL; GRADE : CLASS_REC;
          FOUND : BOOLEAN; VAR GRAND_TOTAL : INTEGER;
           COUNT : INTEGER; EXTRA_BASIC : BASIC_ARRAY);FORWARD;


PROCEDURE FIND_LOC(GRADE : CLASS_REC; T_AREA : STR1
                   VAR AREA : STR1);

BEGIN
  SUBSTR(GRADE.LOC1,2,1,AREA);
  IF AREA = T_AREA THEN
      SUBSTR(GRADE.LOC2,2,1,AREA);
  ELSE
    IF GRADE.RPT > "2" THEN
      BEGIN
        SUBSTR(GRADE.LOC2,2,1,AREA);
        IF AREA = T_AREA THEN
            SUBSTR(GRADE.LOC3,2,1,AREA);
        ELSE
          IF GRADE.RPT > "3" THEN
            BEGIN
              SUBSTR(GRADE.LOC3,2,1,AREA);
              IF AREA = T_AREA THEN
                  SUBSTR(GRADE.LOC4,2,1,AREA);
              ELSE
                IF GRADE.RPT > "4" THEN
                  BEGIN
                    SUBSTR(GRADE.LOC4,2,1,AREA);
                    IF AREA = T_AREA THEN
                        SUBSTR(GRADE.LOC5,2,1,AREA);
                    ELSE
                      AREA := " ";
                  END;
            END;
      END;
END;

PROCEDURE ECON_PLACE(VAR AREA : STR1; VAR SOC : SOCIAL;
                     VAR GRADE : CLASS_REC; VAR FOUND : BOOLEAN;
                     VAR GRAND_TOTAL : INTEGER; COUNT : INTEGER;
                     VAR EXTRA_BASIC : BASIC_ARRAY);
VAR
  T_GRADE : CLASS_REC;
  T_AREA  : STR1;

BEGIN
  T_AREA:= "1";
  IF SOC.ECON_TOTAL < ECON_MAX THEN
    BEGIN
      FOUND := TRUE;
      SOC.ECONOMICS.DEPT_CODE := GRADE.DEPT;
      SOC.ECONOMICS.COURSE_NUM := GRADE.COURSE;
      SOC.ECONOMICS.SEMESTER := GRADE.SEMESTER;
      SOC.ECONOMICS.CRSE_GRADE := GRADE.GRADE;
      SOC.ECONOMICS.CRSE_HOURS := GRADE.HOURS;
      SOC.ECONOMICS.RPT := GRADE.RPT;
      SOC.ECONOMICS.LOC1 := GRADE.LOC1;
      SOC.ECON_TOTAL := SOC.ECON_TOTAL+TRUNC(STOR(GRADE.HOURS));
      GRAND_TOTAL := GRAND_TOTAL+TRUNC(STOR(GRADE.HOURS));
      IF GRADE.RPT > "1" THEN
```

```
  T_GRADE : CLASS_REC;
  T_AREA  : STR1;

BEGIN
  T_AREA:= "1";
  IF SOC.ECON_TOTAL < ECON_MAX THEN
    BEGIN
      FOUND := TRUE;
      SOC.ECONOMICS.DEPT_CODE := GRADE.DEPT;
      SOC.ECONOMICS.COURSE_NUM := GRADE.COURSE;
      SOC.ECONOMICS.SEMESTER := GRADE.SEMESTER;
      SOC.ECONOMICS.CRSE_GRADE := GRADE.GRADE;
      SOC.ECONOMICS.CRSE_HOURS := GRADE.HOURS;
      SOC.ECONOMICS.RPT := GRADE.RPT;
      SOC.ECONOMICS.LOC1 := GRADE.LOC1;
      SOC.ECON_TOTAL := SOC.ECON_TOTAL+TRUNC(STOR(GRADE.HOURS));
      GRAND_TOTAL := GRAND_TOTAL+TRUNC(STOR(GRADE.HOURS));
      IF GRADE.RPT > "1" THEN
        BEGIN
          SOC.ECONOMICS.LOC2 := GRADE.LOC2;
          IF GRADE.RPT > "2" THEN
            BEGIN
              SOC.ECONOMICS.LOC3 := GRADE.LOC3;
              IF GRADE.RPT > "3" THEN
                BEGIN
                  SOC.ECONOMICS.LOC4 := GRADE.LOC4;
                  IF GRADE.RPT > "4" THEN
                    SOC.ECONOMICS.LOC5 := GRADE.LOC5;
                END;
            END;
        END;
    END
  ELSE
    IF GRADE.RPT > "1" THEN
      BEGIN
        FIND_LOC(GRADE,T_AREA,AREA);
        IF AREA <> " " THEN
          SELECT_AREA(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,
                      COUNT, EXTRA_BASIC);
      END
    ELSE
      IF SOC.ECONOMICS.RPT > "1" THEN
        BEGIN
          T_GRADE := GRADE;
          GRADE.DEPT := SOC.ECONOMICS.DEPT_CODE;
          GRADE.COURSE := SOC.ECONOMICS.COURSE_NUM;
          GRADE.SEMESTER := SOC.ECONOMICS.SEMESTER;
          GRADE.GRADE := SOC.ECONOMICS.CRSE_GRADE;
          GRADE.HOURS := SOC.ECONOMICS.CRSE_HOURS;
          GRADE.RPT := SOC.ECONOMICS.RPT;
          GRADE.LOC1 := SOC.ECONOMICS.LOC1;
          SOC.ECON_TOTAL := SOC.ECON_TOTAL-TRUNC(STOR(GRADE.HOURS));
          GRAND_TOTAL := GRAND_TOTAL-TRUNC(STOR(GRADE.HOURS));
          GRADE.LOC2 := SOC.ECONOMICS.LOC2;
          IF GRADE.RPT > "2" THEN
            BEGIN
              GRADE.LOC3 := SOC.ECONOMICS.LOC3;
              IF GRADE.RPT > "3" THEN
                BEGIN
                  GRADE.LOC4 := SOC.ECONOMICS.LOC4;
                  IF GRADE.RPT > "4" THEN
                    GRADE.LOC5 := SOC.ECONOMICS.LOC5
                  ELSE
                    GRADE.LOC5 := '    ';
                END
              ELSE
                BEGIN
                  GRADE.LOC4 := '    ';
                  GRADE.LOC5 := '    ';
                END;
            END
          ELSE
            BEGIN
              GRADE.LOC3 := '    ';
              GRADE.LOC4 := '    ';
              GRADE.LOC5 := '    ';
            END;
          FIND_LOC(GRADE,T_AREA,AREA);
          IF AREA <> " " THEN
            BEGIN
              SELECT_AREA(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,
                          COUNT,EXTRA_BASIC);
              IF FOUND THEN
```

```
                BEGIN
                  GRADE.LOC4 := '    ';
                  GRADE.LOC5 := '    ';
                END;
            END
          ELSE
            BEGIN
              GRADE.LOC3 := '    ';
              GRADE.LOC4 := '    ';
              GRADE.LOC5 := '    ';
            END;
          FIND_LOC(GRADE,T_AREA,AREA);
          IF AREA <> " " THEN
            BEGIN
              SELECT_AREA(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,
                          COUNT,EXTRA_BASIC);
              IF FOUND THEN
                BEGIN
                  SOC.ECONOMICS.DEPT_CODE := T_GRADE.DEPT;
                  SOC.ECONOMICS.COURSE_NUM := T_GRADE.COURSE;
                  SOC.ECONOMICS.SEMESTER := T_GRADE.SEMESTER;
                  SOC.ECONOMICS.CRSE_GRADE := T_GRADE.GRADE;
                  SOC.ECONOMICS.CRSE_HOURS := T_GRADE.HOURS;
                  SOC.ECONOMICS.RPT := T_GRADE.RPT;
                  SOC.ECONOMICS.LOC1 := T_GRADE.LOC1;
                  SOC.ECON_TOTAL :=
                    SOC.ECON_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                  GRAND_TOTAL :=
                    GRAND_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                  IF T_GRADE.RPT > "1" THEN
                    BEGIN
                      SOC.ECONOMICS.LOC2 := T_GRADE.LOC2;
                      IF T_GRADE.RPT > "2" THEN
                        BEGIN
                          SOC.ECONOMICS.LOC3 := T_GRADE.LOC3;
                          IF T_GRADE.RPT > "3" THEN
                            BEGIN
                              SOC.ECONOMICS.LOC4 := T_GRADE.LOC4;
                              IF T_GRADE.RPT > "4" THEN
                                SOC.ECONOMICS.LOC5 := T_GRADE.LOC5;
                            END;
                        END;
                    END;
                END
                ELSE
                  EXTRA_BASIC[COUNT]:= T_GRADE;
            END;
        END
        ELSE
          EXTRA_BASIC[COUNT]:= GRADE;
END;

PROCEDURE GEO_PLACE(VAR AREA : STR1; VAR SOC : SOCIAL;
                    VAR GRADE : CLASS_REC; VAR FOUND : BOOLEAN;
                    VAR GRAND_TOTAL : INTEGER; COUNT : INTEGER
                    VAR EXTRA_BASIC : BASIC_ARRAY);

VAR
  T_GRADE : CLASS_REC;
  T_AREA  : STR1;

BEGIN
  T_AREA:= "2";
  IF SOC.GEOGR_TOTAL < GEO_MAX THEN
    BEGIN
      FOUND := TRUE;
      SOC.GEOGRAPHY.DEPT_CODE := GRADE.DEPT;
      SOC.GEOGRAPHY.COURSE_NUM := GRADE.COURSE;
      SOC.GEOGRAPHY.SEMESTER := GRADE.SEMESTER;
      SOC.GEOGRAPHY.CRSE_GRADE := GRADE.GRADE;
      SOC.GEOGRAPHY.CRSE_HOURS := GRADE.HOURS;
      SOC.GEOGRAPHY.RPT := GRADE.RPT;
      SOC.GEOGRAPHY.LOC1 := GRADE.LOC1;
      SOC.GEOGR_TOTAL := SOC.GEO_TOTAL+TRUNC(STOR(GRADE.HOURS));
      GRAND_TOTAL := GRAND_TOTAL+TRUNC(STOR(GRADE.HOURS));
      IF GRADE.RPT > "1" THEN
        BEGIN
          SOC.GEOGRAPHY.LOC2 := GRADE.LOC2;
          IF GRADE.RPT > "2" THEN
            BEGIN
              SOC.GEOGRAPHY.LOC3 := GRADE.LOC3;
              IF GRADE.RPT > "3" THEN
                BEGIN
                  SOC.GEOGRAPHY.LOC4 := GRADE.LOC4;
```

```
VAR
      SOC.GEOGRAPHY.DEPT_CODE := GRADE.DEPT;
      SOC.GEOGRAPHY.COURSE_NUM := GRADE.COURSE;
      SOC.GEOGRAPHY.SEMESTER := GRADE.SEMESTER;
      SOC.GEOGRAPHY.CRSE_GRADE := GRADE.GRADE;
      SOC.GEOGRAPHY.CRSE_HOURS := GRADE.HOURS;
      SOC.GEOGRAPHY.RPT := GRADE.RPT;
      SOC.GEOGRAPHY.LOC1 := GRADE.LOC1;
      SOC.GEOGR_TOTAL := SOC.GEO_TOTAL+TRUNC(STOR(GRADE.HOURS));
      GRAND_TOTAL := GRAND_TOTAL+TRUNC(STOR(GRADE.HOURS));
      IF GRADE.RPT > "1" THEN
        BEGIN
          SOC.GEOGRAPHY.LOC2 := GRADE.LOC2;
          IF GRADE.RPT > "2" THEN
            BEGIN
              SOC.GEOGRAPHY.LOC3 := GRADE.LOC3;
              IF GRADE.RPT > "3" THEN
                BEGIN
                  SOC.GEOGRAPHY.LOC4 := GRADE.LOC4;
                  IF GRADE.RPT > "4" THEN
                    SOC.GEOGRAPHY.LOC5 := GRADE.LOC5;
                END;
            END;
        END;
    END
  ELSE
    IF GRADE.RPT > "1" THEN
      BEGIN
        FIND_LOC(GRADE,T_AREA,AREA);
        IF AREA <> " " THEN
          SELECT_AREA(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,
                      COUNT, EXTRA_BASIC);
      END
    ELSE
      IF SOC.GEOGRAPHY.RPT > "1" THEN
        BEGIN
          T_GRADE := GRADE;
          GRADE.DEPT := SOC.GEOGRAPHY.DEPT_CODE;
          GRADE.COURSE := SOC.GEOGRAPHY.COURSE_NUM;
          GRADE.SEMESTER := SOC.GEOGRAPHY.SEMESTER;
          GRADE.GRADE := SOC.GEOGRAPHY.CRSE_GRADE;
          GRADE.HOURS := SOC.GEOGRAPHY.CRSE_HOURS;
          GRADE.RPT := SOC.GEOGRAPHY.RPT;
          GRADE.LOC1 := SOC.GEOGRAPHY.LOC1;
          SOC.GEOGR_TOTAL := SOC.GEOGR_TOTAL-
                     TRUNC(STOR(GRADE.HOURS));
          GRAND_TOTAL := GRAND_TOTAL-TRUNC(STOR(GRADE.HOURS));
          GRADE.LOC2 := SOCIAL.GEOGRAPHY.LOC2;
            IF GRADE.RPT > "2" THEN
              BEGIN
                GRADE.LOC3 := SOC.GEOGRAPHY.LOC3;
                IF GRADE.RPT > "3" THEN
                  BEGIN
                    GRADE.LOC4 := SOC.GEOGRAPHY.LOC4;
                    IF GRADE.RPT > "4" THEN
                      GRADE.LOC5 := SOC.GEOGRAPHY.LOC5
                    ELSE
                      GRADE.LOC5 := '    ';
                  END
                ELSE
                  BEGIN
                    GRADE.LOC4 := '    ';
                    GRADE.LOC5 := '    ';
                  END;
              END
            ELSE
              BEGIN
                GRADE.LOC3 := '    ';
                GRADE.LOC4 := '    ';
                GRADE.LOC5 := '    ';
              END;
          FIND_LOC(GRADE,T_AREA,AREA);
          IF AREA <> " " THEN
            BEGIN
              SELECT_AREA(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,
                          COUNT, EXTRA_BASIC);
              IF FOUND THEN
                BEGIN
                  SOC.GEOGRAPHY.DEPT_CODE := T_GRADE.DEPT;
                  SOC.GEOGRAPHY.COURSE_NUM := T_GRADE.COURSE;
                  SOC.GEOGRAPHY.SEMESTER := T_GRADE.SEMESTER;
                  SOC.GEOGRAPHY.CRSE_GRADE := T_GRADE.GRADE;
                  SOC.GEOGRAPHY.CRSE_HOURS := T_GRADE.HOURS;
                  SOC.GEOGRAPHY.RPT := T_GRADE.RPT;
```

```
            BEGIN
              GRADE.LOC3 := '    ';
              GRADE.LOC4 := '    ';
              GRADE.LOC5 := '    ';
            END;
        FIND_LOC(GRADE,T_AREA,AREA);
        IF AREA <> " " THEN
          BEGIN
            SELECT_AREA(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,
                        COUNT, EXTRA_BASIC);
            IF FOUND THEN
              BEGIN
                SOC.GEOGRAPHY.DEPT_CODE := T_GRADE.DEPT;
                SOC.GEOGPAPHY.COURSE_NUM := T_GRADE.COURSE;
                SOC.GEOGRAPHY.SEMESTER := T_GRADE.SEMESTER;
                SOC.GEOGRAPHY.CRSE_GRADE := T_GRADE.GRADE;
                SOC.GEOGRAPHY.CRSE_HOURS := T_GRADE.HOURS;
                SOC.GEOGRAPHY.RPT := T_GRADE.RPT;
                SOC.GEOGRAPHY.LOC1 := T_GRADE.LOC1;
                SOC.GEOGR_TOTAL :=
                  SOC.GEOGR_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                GRAND_TOTAL :=
                  GRAND_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                IF T_GRADE.RPT > "1" THEN
                  BEGIN
                    SOC.GEOGRAPHY.LOC2 := T_GRADE.LOC2;
                    IF T_GRADE.RPT > "2" THEN
                      BEGIN
                        SOC.GEOGRAPHY.LOC3 := T_GRADE.LOC3;
                        IF T_GRADE.RPT > "3" THEN
                          BEGIN
                            SOC.GEOGRAPHY.LOC4 := T_GRADE.LOC4;
                            IF T_GRADE.RPT > "4" THEN
                              SOC.GEOGRAPHY.LOC5 := T_GRADE.LOC5;
                          END;
                      END;
                  END;
              END
              ELSE
                EXTRA_BASIC[COUNT]:= T_GRADE;
          END;
      END
      ELSE
        EXTRA_BASIC[COUNT]:= GRADE;
END;

PROCEDURE POL_PLACE(VAR AREA : STR1; VAR SOC : SOCIAL;
                    VAR GRADE : CLASS_REC; VAR FOUND : BOOLEAN;
                    VAR GRAND_TOTAL : INTEGER; COUNT : INTEGER
                    VAR EXTRA_BASIC : BASIC_ARRAY);
VAR
  T_GRADE : CLASS_REC;
  T_AREA  : STR1;


BEGIN
  T_AREA:= "3";
  IF SOC.POL_SCI_TOTAL < POL_MAX THEN
    BEGIN
      FOUND := TRUE;
      SOC.POLITICAL_SCIENCE.DEPT_CODE := GRADE.DEPT;
      SOC.POLITICAL_SCIENCE.COURSE_NUM := GRADE.COURSE;
      SOC.POLITICAL_SCIENCE.SEMESTER := GRADE.SEMESTER;
      SOC.POLITICAL_SCIENCE.CRSE_GRADE := GRADE.GRADE;
      SOC.POLITICAL_SCIENCE.CRSE_HOURS := GRADE.HOURS;
      SOC.POLITICAL_SCIENCE.RPT := GRADE.RPT;
      SOC.POLITICAL_SCIENCE.LOC1 := GRADE.LOC1;
      SOC.POL_SCI_TOTAL := SOC.POL_SCI_TOTAL+TRUNC(STOR(GRADE.HOURS));
      GRAND_TOTAL := GRAND_TOTAL+TRUNC(STOR(GRADE.HOURS));
      IF GRADE.RPT > "1" THEN
        BEGIN
          SOC.POLITICAL_SCIENCE.LOC2 := GRADE.LOC2;
          IF GRADE.RPT > "2" THEN
            BEGIN
              SOC.POLITICAL_SCIENCE.LOC3 := GRADE.LOC3;
              IF GRADE.RPT > "3" THEN
                BEGIN
                  SOC.POLITICAL_SCIENCE.LOC4 := GRADE.LOC4;
                  IF GRADE.RPT > "4" THEN
                    SOC.POLITICAL_SCIENCE.LOC5 := GRADE.LOC5;
                END;
            END;
        END;
    END
```

```
VAR
    SOC.POL_SCI_TOTAL := SOC.POL_SCI_TOTAL+TRUNC(STOR(GRADE.HOURS));
    GRAND_TOTAL := GRAND_TOTAL+TRUNC(STOR(GRADE.HOURS));
    IF GRADE.RPT > "1" THEN
      BEGIN
        SOC.POLITICAL_SCIENCE.LOC2 := GRADE.LOC2;
        IF GRADE.RPT > "2" THEN
          BEGIN
            SOC.POLITICAL_SCIENCE.LOC3 := GRADE.LOC3;
            IF GRADE.RPT > "3" THEN
              BEGIN
                SOC.POLITICAL_SCIENCE.LOC4 := GRADE.LOC4;
                IF GRADE.RPT > "4" THEN
                  SOC.POLITICAL_SCIENCE.LOC5 := GRADE.LOC5;
              END;
          END;
      END;
  END
ELSE
  IF GRADE.RPT > "1" THEN
    BEGIN
      FIND_LOC(GRADE,T_AREA,AREA);
      IF AREA <> " " THEN
        SELECT_AREA(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,
                    COUNT, EXTRA_BASIC);
    END
  ELSE IF SOC.POLITICAL_SCIENCE.RPT > "1" THEN
    BEGIN
      T_GRADE := GRADE;
      GRADE.DEPT := SOC.POLITICAL_SCIENCE.DEPT_CODE;
      GRADE.COURSE := SOC.POLITICAL_SCIENCE.COURSE_NUM;
      GRADE.SEMESTER := SOC.POLITICAL_SCIENCE.SEMESTER;
      GRADE.GRADE := SOC.POLITICAL_SCIENCE.CRSE_GRADE;
      GRADE.HOURS := SOC.POLITICAL_SCIENCE.CRSE_HOURS;
      GRADE.RPT := SOC.POLITICAL_SCIENCE.RPT;
      GRADE.LOC1 := SOC.POLITICAL_SCIENCE.LOC1;
      SOC.POL_SCI_TOTAL := SOC.POL_SCI_TOTAL-
          TRUNC(STOR(GRADE.HOURS));
      GRAND_TOTAL := GRAND_TOTAL-TRUNC(STOR(GRADE.HOURS));
      GRADE.LOC2 := SOC.POLITICAL_SCIENCE.LOC2;
      IF GRADE.RPT > "2" THEN
        BEGIN
          GRADE.LOC3 := SOC.POLITICAL_SCIENCE.LOC3;
          IF GRADE.RPT > "3" THEN
            BEGIN
              GRADE.LOC4 := SOC.POLITICAL_SCIENCE.LOC4;
              IF GRADE.RPT > "4" THEN
                GRADE.LOC5 := SOC.POLITICAL_SCIENCE.LOC5
              ELSE
                GRADE.LOC5 := '    ';
            END
          ELSE
            BEGIN
              GRADE.LOC4 := '    ';
              GRADE.LOC5 := '    ';
            END;
        END
      ELSE
        BEGIN
          GRADE.LOC3 := '    ';
          GRADE.LOC4 := '    ';
          GRADE.LOC5 := '    ';
        END;
      FIND_LOC(GRADE,T_AREA,AREA);
      IF AREA <> " " THEN
        BEGIN
          SELECT_AREA(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,
                      COUNT, EXTRA_BASIC);
          IF FOUND THEN
            BEGIN
              SOC.POLITICAL_SCIENCE.DEPT_CODE := T_GRADE.DEPT;
              SOC.POLITICAL_SCIENCE.COURSE_NUM := T_GRADE.COURSE;
              SOC.POLITICAL_SCIENCE.SEMESTER := T_GRADE.SEMESTER;
              SOC.POLITICAL_SCIENCE.CRSE_GRADE := T_GRADE.GRADE;
              SOC.POLITICAL_SCIENCE.CRSE_HOURS := T_GRADE.HOURS;
              SOC.POLITICAL_SCIENCE.RPT := T_GRADE.RPT;
              SOC.POLITICAL_SCIENCE.LOC1 := T_GRADE.LOC1;
              SOC.POL_SCI_TOTAL :=
                SOC.POL_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
              GRAND_TOTAL :=
                GRAND_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
              IF T_GRADE.RPT > "1" THEN
                BEGIN
                  SOC.POLITICAL_SCIENCE.LOC2 := T_GRADE.LOC2;
```

```
                    COUNT, EXTRA_BASIC);
            IF FOUND THEN
              BEGIN
                SOC.POLITICAL_SCIENCE.DEPT_CODE := T_GRADE.DEPT;
                SOC.POLITICAL_SCIENCE.COURSE_NUM := T_GRADE.COURSE;
                SOC.POLITICAL_SCIENCE.SEMESTER := T_GRADE.SEMESTER;
                SOC.POLITICAL_SCIENCE.CRSE_GRADE := T_GRADE.GRADE;
                SOC.POLITICAL_SCIENCE.CRSE_HOURS := T_GRADE.HOURS;
                SOC.POLITICAL_SCIENCE.RPT := T_GRADE.RPT;
                SOC.POLITICAL_SCIENCE.LOC1 := T_GRADE.LOC1;
                SOC.POL_SCI_TOTAL :=
                  SOC.POL_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                GRAND_TOTAL :=
                  GRAND_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                IF T_GRADE.RPT > "1" THEN
                  BEGIN
                    SOC.POLITICAL_SCIENCE.LOC2 := T_GRADE.LOC2;
                    IF T_GRADE.RPT > "2" THEN
                      BEGIN
                        SOC.POLITICAL_SCIENCE.LOC3 := T_GRADE.LOC3;
                        IF T_GRADE.RPT > "3" THEN
                          BEGIN
                            SOC.POLITICAL_SCIENCE.LOC4 :=
                              T_GRADE.LOC4;
                            IF T_GRADE.RPT > "4" THEN
                              SOC.POLITICAL_SCIENCE.LOC5 :=
                                T_GRADE.LOC5;
                          END;
                      END;
                  END;
              END
              ELSE
                EXTRA_BASIC[COUNT]:= T_GRADE;
          END;
      END
      ELSE
        EXTRA_BASIC[COUNT]:= GRADE;
END;

PROCEDURE PSY_PLACE(VAR AREA : STR1; VAR SOC : SOCIAL;
                    VAR GRADE : CLASS_REC; VAR FOUND : BOOLEAN;
                    VAR GRAND_TOTAL : INTEGER; COUNT : INTEGER;
                    VAR EXTRA_BASIC : BASIC_ARRAY);
VAR
  T_GRADE : CLASS_REC;
  T_AREA  : STR1;

BEGIN
  T_AREA:= "4";
  IF SOC.PSY_TOTAL < PSY_MAX THEN
    BEGIN
      FOUND := TRUE;
      SOC.PSYCHOLOGY.DEPT_CODE := GRADE.DEPT;
      SOC.PSYCHOLOGY.COURSE_NUM := GRADE.COURSE;
      SOC.PSYCHOLOGY.SEMESTER := GRADE.SEMESTER;
      SOC.PSYCHOLOGY.GRADE := GRADE.GRADE;
      SOC.PSYCHOLOGY.CRSE_HOURS := GRADE.HOURS;
      SOC.PSYCHOLOGY.RPT := GRADE.RPT;
      SOC.PSYCHOLOGY.LOC1 := GRADE.LOC1;
      SOC.PSY_TOTAL := SOC.PSY_TOTAL+TRUNC(STOR(GRADE.HOURS));
      GRAND_TOTAL := GRAND_TOTAL+TRUNC(STOR(GRADE.HOURS));
      IF GRADE.RPT > "1" THEN
        BEGIN
          SOC.PSYCHOLOGY.LOC2 := GRADE.LOC2;
          IF GRADE.RPT > "2" THEN
            BEGIN
              SOC.PSYCHOLOGY.LOC3 := GRADE.LOC3;
              IF GRADE.RPT > "3" THEN
                BEGIN
                  SOC.PSYCHOLOGY.LOC4 := GRADE.LOC4;
                  IF GRADE.RPT > "4" THEN
                    SOC.PSYCHOLOGY.LOC5 := GRADE.LOC5;
                END;
            END;
        END;
    END
  ELSE
    IF GRADE.RPT > "1" THEN
      BEGIN
        FIND_LOC(GRADE,T_AREA,AREA);
        IF AREA <> " " THEN
          SELECT_AREA(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,
                      COUNT, EXTRA_BASIC);
```

```
VAR
            SOC.PSYCHOLOGY.LOC3 := GRADE.LOC3;
            IF GRADE.RPT > "3" THEN
              BEGIN
                SOC.PSYCHOLOGY.LOC4 := GRADE.LOC4;
                IF GRADE.RPT > "4" THEN
                  SOC.PSYCHOLOGY.LOC5 := GRADE.LOC5;
              END;
          END;
      END;
  END
ELSE
  IF GRADE.RPT > "1" THEN
    BEGIN
      FIND_LOC(GRADE,T_AREA,AREA);
      IF AREA <> " " THEN
        SELECT_AREA(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,
                    COUNT, EXTRA_BASIC);
    END
  ELSE
    IF SOC.PSYCHOLOGY.RPT > "1" THEN
      BEGIN
        T_GRADE := GRADE;
        GRADE.DEPT := SOC.PSYCHOLOGY.DEPT_CODE;
        GRADE.COURSE := SOC.PSYCHOLOGY.COURSE_NUM;
        GRADE.SEMESTER := SOC.PSYCHOLOGY.SEMESTER;
        GRADE.GRADE := SOC.PSYCHOLOGY.CRSE_GRADE;
        GRADE.HOURS := SOC.PSYCHOLOGY.CRSE_HOURS;
        GRADE.RPT := SOC.PSYCHOLOGY.RPT;
        GRADE.LOC1 := SOC.PSYCHOLOGY.LOC1;
        SOC.PSY_TOTAL :=
          SOC.PSY_TOTAL - TRUNC(STOR(SOC.PSYCHOLOGY.HOURS));
        GRAND_TOTAL := GRAND_TOTAL - TRUNC(STOR(GRADE.HOURS));
        GRADE.LOC2 := SOC.PSYCHOLOGY.LOC2;
        IF SOC.PSYCHOLOGY.RPT > "2" THEN
          BEGIN
            GRADE.LOC3 := SOC.PSYCHOLOGY.LOC3;
            IF SOC.PSYCHOLOGY.RPT > "3" THEN
              BEGIN
                GRADE.LOC4 := SOC.PSYCHOLOGY.LOC4;
                IF SOC.PSYCHOLOGY.RPT > "4" THEN
                  GRADE.LOC5 := SOC.PSYCHOLOGY.LOC5
                ELSE
                  GRADE.LOC5 := '    ';
              END
            ELSE
              BEGIN
                GRADE.LOC4 := '    ';
                GRADE.LOC5 := '    ';
              END;
          END
        ELSE
          BEGIN
            GRADE.LOC3 := '    ';
            GRADE.LOC4 := '    ';
            GRADE.LOC5 := '    ';
          END;
        FIND_LOC(GRADE,T_AREA,AREA);
        IF AREA <> " " THEN
          BEGIN
            SELECT_AREA(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,
                        COUNT, EXTRA_BASIC);
            IF FOUND THEN
              BEGIN
                SOC.PSYCHOLOGY.DEPT_CODE := T_GRADE.DEPT;
                SOC.PSYCHOLOGY.COURSE_NUM := T_GRADE.COURSE;
                SOC.PSYCHOLOGY.SEMESTER := T_GRADE.SEMESTER;
                SOC.PSYCHOLOGY.CRSE_GRADE := T_GRADE.GRADE;
                SOC.PSYCHOLOGY.CRSE_HOURS := T_GRADE.HOURS;
                SOC.PSYCHOLOGY.RPT := T_GRADE.RPT;
                SOC.PSYCHOLOGY.LOC1 := T_GRADE.LOC1;
                SOC.PSY_TOTAL :=
                  SOC.PSY_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                GRAND_TOTAL :=
                  GRAND_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                IF T_GRADE.RPT > "1" THEN
                  BEGIN
                    SOC.PSYCHOLOGY.LOC2 := T_GRADE.LOC2;
                    IF T_GRADE.RPT > "2" THEN
                      BEGIN
                        SOC.PSYCHOLOGY.LOC3 := T_GRADE.LOC3;
                        IF T_GRADE.RPT > "3" THEN
                          BEGIN
                            SOC.PSYCHOLOGY.LOC4 := T_GRADE.LOC4;
```

```
                    SOC.PSYCHOLOGY.SEMESTER := T_GRADE.SEMESTER;
                    SOC.PSYCHOLOGY.CRSE_GRADE := T_GRADE.GRADE;
                    SOC.PSYCHOLOGY.CRSE_HOURS := T_GRADE.HOURS;
                    SOC.PSYCHOLOGY.RPT := T_GRADE.RPT;
                    SOC.PSYCHOLOGY.LOC1 := T_GRADE.LOC1;
                    SOC.PSY_TOTAL :=
                      SOC.PSY_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                    GRAND_TOTAL :=
                      GRAND_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                    IF T_GRADE.RPT > "1" THEN
                      BEGIN
                        SOC.PSYCHOLOGY.LOC2 := T_GRADE.LOC2;
                        IF T_GRADE.RPT > "2" THEN
                          BEGIN
                            SOC.PSYCHOLOGY.LOC3 := T_GRADE.LOC3;
                            IF T_GRADE.RPT > "3" THEN
                              BEGIN
                                SOC.PSYCHOLOGY.LOC4 := T_GRADE.LOC4;
                                IF T_GRADE.RPT > "4" THEN
                                  SOC.PSYCHOLOGY.LOC5 := T_GRADE.LOC5;
                              END;
                          END;
                      END;
                  END
                  ELSE
                    EXTRA_BASIC[COUNT]:= T_GRADE;
              END;
          END
          ELSE
            EXTRA_BASIC[COUNT]:= GRADE;
END;

PROCEDURE SOC_PLACE(VAR AREA : STR1; VAR SOC : SOCIAL;
                    VAR GRADE : CLASS_REC; VAR FOUND : BOOLEAN;
                    VAR GRAND_TOTAL : INTEGER; COUNT : INTEGER
                    VAR EXTRA_BASIC : BASIC_ARRAY);

VAR
  T_GRADE : CLASS_REC;
  T_AREA  : STR1;

BEGIN
  T_AREA:= "5"
  IF SOC.SOC_TOTAL < SOC_MAX THEN
    BEGIN
      FOUND := TRUE;
      SOC.SOCIOLOGY.DEPT_CODE := GRADE.DEPT;
      SOC.SOCIOLOGY.COURSE_NUM := GRADE.COURSE;
      SOC.SOCIOLOGY.SEMESTER := GRADE.SEMESTER;
      SOC.SOCIOLOGY.CRSE_GRADE := GRADE.GRADE;
      SOC.SOCIOLOGY.CRSE_HOURS := GRADE.HOURS;
      SOC.SOCIOLOGY.RPT := GRADE.RPT;
      SOC.SOCIOLOGY.LOC1 := GRADE.LOC1;
      SOC.SOC_TOTAL := SOC.SOC_TOTAL+TRUNC(STOR(GRADE.HOURS));
      GRAND_TOTAL := GRAND_TOTAL+TRUNC(STOR(GRADE.HOURS));
      IF GRADE.RPT > "1" THEN
        BEGIN
          SOC.SOCIOLOGY.LOC2 := GRADE.LOC2;
          IF GRADE.RPT > "2" THEN
            BEGIN
              SOC.SOCIOLOGY.LOC3 := GRADE.LOC3;
              IF GRADE.RPT > "3" THEN
                BEGIN
                  SOC.SOCIOLOGY.LOC4 := GRADE.LOC4;
                  IF GRADE.RPT > "4" THEN
                    SOC.SOCIOLOGY.LOC5 := GRADE.LOC5;
                END;
            END;
        END;
    END
  ELSE
    IF GRADE.RPT > "1" THEN
      BEGIN
        FIND_LOC(GRADE,T_AREA,AREA);
        IF AREA <> " " THEN
          SELECT_AREA(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,
                      COUNT, EXTRA_BASIC);
      END
    ELSE IF SOC.SOCIOLOGY.RPT > "1" THEN
      BEGIN
        T_GRADE := GRADE;
        GRADE.DEPT := SOC.SOCIOLOGY.DEPT_CODE;
        GRADE.COURSE := SOC.SOCIOLOGY.COURSE_NUM;
        GRADE.SEMESTER := SOC.SOCIOLOGY.SEMESTER;
```

```
VAR
            END;
        END;
    END
  ELSE
    IF GRADE.RPT > "1" THEN
      BEGIN
        FIND_LOC(GRADE,T_AREA,AREA);
        IF AREA <> " " THEN
          SELECT_AREA(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,
                      COUNT, EXTRA_BASIC);
      END
    ELSE IF SOC.SOCIOLOGY.RPT > "1" THEN
      BEGIN
        T_GRADE := GRADE;
        GRADE.DEPT := SOC.SOCIOLOGY.DEPT_CODE;
        GRADE.COURSE := SOC.SOCIOLOGY.COURSE_NUM;
        GRADE.SEMESTER := SOC.SOCIOLOGY.SEMESTER;
        GRADE.GRADE := SOC.SOCIOLOGY.CRSE_GRADE;
        GRADE.HOURS := SOC.SOCIOLOGY.CRSE_HOURS;
        GRADE.RPT := SOC.SOCIOLOGY.RPT;
        GRADE.LOC1 := SOC.SOCIOLOGY.LOC1;
        SOC.SOC_TOTAL := SOC.SOC_TOTAL-TRUNC(STOR(GRADE.HOURS));
        GRAND_TOTAL := GRAND_TOTAL-TRUNC(STOR(GRADE.HOURS));
        GRADE.LOC2 := SOC.SOCIOLOGY.LOC2;
        IF SOC.SOCIOLOGY.RPT > "2" THEN
          BEGIN
            GRADE.LOC3 := SOC.SOCIOLOGY.LOC3;
            IF SOC.SOCIOLOGY.RPT > "3" THEN
              BEGIN
                GRADE.LOC4 := SOC.SOCIOLOGY.LOC4;
                IF SOC.SOCIOLOGY.RPT > "4" THEN
                  GRADE.LOC5 := SOC.SOCIOLOGY.LOC5
                ELSE
                  GRADE.LOC5 := '    ';
              END
            ELSE
              BEGIN
                GRADE.LOC4 := '    ';
                GRADE.LOC5 := '    ';
              END;
          END
        ELSE
          BEGIN
            GRADE.LOC3 := '    ';
            GRADE.LOC4 := '    ';
            GRADE.LOC5 := '    ';
          END;
        FIND_LOC(GRADE,T_AREA,AREA);
        IF AREA <> " " THEN
          BEGIN
            SELECT_AREA(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,
                        COUNT, EXTRA_BASIC);
            IF FOUND THEN
              BEGIN
                SOC.SOCIOLOGY.DEPT_CODE := T_GRADE.DEPT;
                SOC.SOCIOLOGY.COURSE_NUM := T_GRADE.COURSE;
                SOC.SOCIOLOGY.SEMESTER := T_GRADE.SEMESTER;
                SOC.SOCIOLOGY.CRSE_GRADE := T_GRADE.GRADE;
                SOC.SOCIOLOGY.CRSE_HOURS := T_GRADE.HOURS;
                SOC.SOCIOLOGY.RPT := T_GRADE.RPT;
                SOC.SOCIOLOGY.LOC1 := T_GRADE.LOC1;
                SOC.SOC_TOTAL := SOC.SOC_TOTAL+
                                 TRUNC(STOR(T_GRADE.HOURS));
                GRAND_TOTAL :=
                  GRAND_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                IF T_GRADE.RPT > "1" THEN
                  BEGIN
                    SOC.SOCIOLOGY.LOC2 := T_GRADE.LOC2;
                    IF T_GRADE.RPT > "2" THEN
                      BEGIN
                        SOC.SOCIOLOGY.LOC3 := T_GRADE.LOC3;
                        IF T_GRADE.RPT > "3" THEN
                          BEGIN
                            SOC.SOCIOLOGY.LOC4 := T_GRADE.LOC4;
                            IF T_GRADE.RPT > "4" THEN
                              SOC.SOCIOLOGY.LOC5 := T_GRADE.LOC5;
                          END;
                      END;
                  END;
              END
              ELSE
                EXTRA_BASIC[COUNT]:= T_GRADE;
          END;
```

```
                IF T_GRADE.RPT > "1" THEN
                  BEGIN
                    SOC.SOCIOLOGY.LOC2 := T_GRADE.LOC2;
                    IF T_GRADE.RPT > "2" THEN
                      BEGIN
                        SOC.SOCIOLOGY.LOC3 := T_GRADE.LOC3;
                        IF T_GRADE.RPT > "3" THEN
                          BEGIN
                            SOC.SOCIOLOGY.LOC4 := T_GRADE.LOC4;
                            IF T_GRADE.RPT > "4" THEN
                               SOC.SOCIOLOGY.LOC5 := T_GRADE.LOC5;
                          END;
                      END;
                  END;
              END
              ELSE
                EXTRA_BASIC[COUNT]:= T_GRADE;
          END;
      END
      ELSE
        EXTRA_BASIC[COUNT]:= GRADE;
END;

PROCEDURE SELECT_AREA;
VAR
  TEMP : CHAR;

BEGIN
  SUBSTR(AREA,1,1,TEMP);
  CASE TEMP OF
    '1' : ECON_PLACE(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,COUNT,
               EXTRA_BASIC);
    '2' : GEO_PLACE(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,COUNT,
               EXTRA_BASIC);
    '3' : POL_PLACE(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,COUNT,
               EXTRA_BASIC);
    '4' : PSY_PLACE(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,COUNT,
               EXTRA_BASIC);
    '5' : SOC_PLACE(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,COUNT,
               EXTRA_BASIC);
  END;
END;


BEGIN
   FOUND := FALSE;
   IF SOC.SOC_SCI_TOTAL < SOC_SCI_MAX THEN
   BEGIN
      SUBSTR(GRADE.LOC1,2,1,AREA);
      SELECT_AREA(AREA,SOC,GRADE,FOUND,GRAND_TOTAL,COUNT,
                  EXTRA_BASIC);
   END
   ELSE
     EXTRA_BASIC[COUN]:= GRADE;
END;


PROCEDURE PART_D(GRADE : CLASS_REC; VAR SCI : SCIENCE ;
                 VAR GRAND_TOTAL : INTEGER; COUNT :INTEGER;
                 VAR EXTRA_BASIC : BASIC_ARRAY);

CONST
  SCI_MATH_MAX := 12;
  BIO_MAX := 3;
  PHY_MAX := 3;
  MAT_MAX := 3;
  ELEC_MAX := 3;

TYPE
  STR1 = PACKED ARRAY[1..1] OF CHAR;

VAR
  T_AREA : STR1;
  AREA   : STR1;
  FOUND  : BOOLEAN;

PROCEDURE SELECT_AREA2(AREA:STR1;SCI:SCIENCE;GRADE:CLASS_REC;
       FOUND:BOOLEAN;VAR GRAND_TOTAL:INTEGER;COUNT : INTEGER;
        VAR EXTRA_BASIC : BASIC_ARRAY);FORWARD;

PROCEDURE FIND_LOC(GRADE : CLASS_REC; T_AREA : STR1;
                   VAR AREA:STR1);

BEGIN
```

```
TYPE
  STR1 = PACKED ARRAY[1..1] OF CHAR;

VAR
  T_AREA : STR1;
  AREA   : STR1;
  FOUND  : BOOLEAN;

PROCEDURE SELECT_AREA2(AREA:STR1;SCI:SCIENCE;GRADE:CLASS_REC;
       FOUND:BOOLEAN;VAR GRAND_TOTAL:INTEGER;COUNT : INTEGER;
        VAR EXTRA_BASIC : BASIC_ARRAY);FORWARD;

PROCEDURE FIND_LOC(GRADE : CLASS_REC; T_AREA : STR1;
                   VAR AREA:STR1);

BEGIN
    SUBSTR(GRADE.LOC1,2,1,AREA);
    IF AREA = T_AREA THEN
        SUBSTR(GRADE.LOC2,2,1,AREA);
    ELSE
      IF GRADE.RPT > "2" THEN
        BEGIN
          SUBSTR(GRADE.LOC2,2,1,AREA);
          IF AREA = T_AREA THEN
              SUBSTR(GRADE.LOC3,2,1,AREA);
          ELSE
            IF GRADE.RPT > "3" THEN
             BEGIN
                SUBSTR(GRADE.LOC3,2,1,AREA);
                IF AREA = T_AREA THEN
                    SUBSTR(GRADE.LOC4,2,1,AREA);
                ELSE
                  IF GRADE.RPT > "4" THEN
                    BEGIN
                      SUBSTR(GRADE.LOC4,2,1,AREA);
                      IF AREA = T_AREA THEN
                          SUBSTR(GRADE.LOC5,2,1,AREA);
                      ELSE
                        AREA:= " ";
                    END;
             END;
        END;
  END;


PROCEDURE BIO_PLACE(VAR AREA : STR1; VAR SCI : SCIENCE;
                    VAR GRADE : CLASS_REC; VAR FOUND : BOOLEAN;
                    VAR GRAND_TOTAL : INTEGER; COUNT : INTEGER
                    VAR EXTRA_BASIC: BASIC_ARRAY);

VAR
  T_GRADE : CLASS_REC;
  T_AREA  : STR1;

BEGIN
    IF SCI.BIO_TOTAL < BIO_MAX THEN
      BEGIN
        FOUND := TRUE;
        SCI.BIOLOGY.DEPT_CODE := GRADE.DEPT;
        SCI.BIOLOGY.COURSE_NUM := GRADE.COURSE;
        SCI.BIOLOGY.SEMESTER := GRADE.SEMESTER;
        SCI.BIOLOGY.CRSE_GRADE := GRADE.GRADE;
        SCI.BIOLOGY.CRSE_HOURS := GRADE.HOURS;
        SCI.BIOLOGY.RPT := GRADE.RPT;
        SCI.BIOLOGY.LOC1 := GRADE.LOC1;
        SCI.BIO_TOTAL := SCI.BIO_TOTAL+TRUNC(STOR(GRADE.HOURS));
        GRAND_TOTAL := GRAND_TOTAL+TRUNC(STOR(GRADE.HOURS));
        IF GRADE.RPT > "1" THEN
          BEGIN
            SCI.BIOLOGY.LOC2 := GRADE.LOC2;
            IF GRADE.RPT > "2" THEN
              BEGIN
                SCI.BIOLOGY.LOC3 := GRADE.LOC3;
                IF GRADE.RPT > "3" THEN
                  BEGIN
                    SCI.BIOLOGY.LOC4 := GRADE.LOC4;
                    IF GRADE.RPT > "4" THEN
                      SCI.BIOLOGY.LOC5 := GRADE.LOC5;
                  END;
              END;
          END;
      END;
```

```
      SCI.BIOLOGY.RPT := GRADE.RPT;
      SCI.BIOLOGY.LOC1 := GRADE.LOC1;
      SCI.BIO_TOTAL := SCI.BIO_TOTAL+TRUNC(STOR(GRADE.HOURS));
      GRAND_TOTAL := GRAND_TOTAL+TRUNC(STOR(GRADE.HOURS));
      IF GRADE.RPT > "1" THEN
        BEGIN
          SCI.BIOLOGY.LOC2 := GRADE.LOC2;
          IF GRADE.RPT > "2" THEN
            BEGIN
              SCI.BIOLOGY.LOC3 := GRADE.LOC3;
              IF GRADE.RPT > "3" THEN
                BEGIN
                  SCI.BIOLOGY.LOC4 := GRADE.LOC4;
                  IF GRADE.RPT > "4" THEN
                    SCI.BIOLOGY.LOC5 := GRADE.LOC5;
                END;
            END;
        END;
    END
  ELSE
    IF GRADE.RPT > "1" THEN
      BEGIN
        FIND_LOC(GRADE,T_AREA,AREA);
        IF AREA <> " " THEN
          SELECT_AREA2(AREA,SCI,GRADE,FOUND,GRAND_TOTAL,
                       COUNT, EXTRA_BASIC);
      END
    ELSE
      IF SCI.BIOLOGY.RPT > "1" THEN
        BEGIN
          T_GRADE := GRADE;
          GRADE.DEPT := SCI.BIOLOGY.DEPT_CODE;
          GRADE.COURSE := SCI.BIOLOGY.COURSE_NUM;
          GRADE.SEMESTER := SCI.BIOLOGY.SEMESTER;
          GRADE.GRADE := SCI.BIOLOGY.GRADE;
          GRADE.HOURS := SCI.BIOLOGY.CRSE_HOURS;
          GRADE.RPT := SCI.BIOLOGY.RPT;
          GRADE.LOC1 := SCI.BIOLOGY.LOC1;
          SCI.BIO_TOTAL := SCI.BIO_TOTAL-TRUNC(STOR(GRADE.HOURS));
          GRAND_TOTAL := GRAND_TOTAL-TRUNC(STOR(GRADE.HOURS));
          GRADE.LOC2 := SCI.BIOLOGY.LOC2;
          IF GRADE.RPT > "2" THEN
            BEGIN
              GRADE.LOC3 := SCI.BIOLOGY.LOC3;
              IF GRADE.RPT > "3" THEN
                BEGIN
                  GRADE.LOC4 := SCI.BIOLOGY.LOC4;
                  IF GRADE.RPT > "4" THEN
                    GRADE.LOC5 := SCI.BIOLOGY.LOC5
                  ELSE
                    GRADE.LOC5 := '    ';
                END
              ELSE
                BEGIN
                  GRADE.LOC4 := '    ';
                  GRADE.LOC5 := '    ';
                END;
            END
          ELSE
            BEGIN
              GRADE.LOC3 := '    ';
              GRADE.LOC4 := '    ';
              GRADE.LOC5 := '    ';
            END;
          FIND_LOC(GRADE,T_AREA,AREA);
          IF AREA <> " " THEN
            BEGIN
              SELECT_AREA2(AREA,SCI,GRADE,FOUND,GRAND_TOTAL,
                           COUNT,EXTRA_BASIC);
              IF FOUND THEN
                BEGIN
                  SCI.BIOLOGY.DEPT_CODE := T_GRADE.DEPT;
                  SCI.BIOLOGY.COURSE_NUM := T_GRADE.COURSE;
                  SCI.BIOLOGY.SEMESTER := T_GRADE.SEMESTER;
                  SCI.BIOLOGY.CRSE_GRADE := T_GRADE.GRADE;
                  SCI.BIOLOGY.CRSE_HOURS := T_GRADE.HOURS;
                  SCI.BIOLOGY.RPT := T_GRADE.RPT;
                  SCI.BIOLOGY.LOC1 := T_GRADE.LOC1;
                  SCI.BIO_TOTAL := SCI.BIO_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                  GRAND_TOTAL :=
                    GRAND_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                  IF T_GRADE.RPT > "1" THEN
                    BEGIN
                      SCI.BIOLOGY.LOC2 := T_GRADE.LOC2;
```

```
VAR
                SELECT_AREA2(AREA,SCI,GRADE,FOUND,GRAND_TOTAL,
                             COUNT,EXTRA_BASIC);
              IF FOUND THEN
                BEGIN
                  SCI.BIOLOGY.DEPT_CODE := T_GRADE.DEPT;
                  SCI.BIOLOGY.COURSE_NUM := T_GRADE.COURSE;
                  SCI.BIOLOGY.SEMESTER := T_GRADE.SEMESTER;
                  SCI.BIOLOGY.CRSE_GRADE := T_GRADE.GRADE;
                  SCI.BIOLOGY.CRSE_HOURS := T_GRADE.HOURS;
                  SCI.BIOLOGY.RPT := T_GRADE.RPT;
                  SCI.BIOLOGY.LOC1 := T_GRADE.LOC1;
                  SCI.BIO_TOTAL := SCI.BIO_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                  GRAND_TOTAL :=
                    GRAND_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                  IF T_GRADE.RPT > "1" THEN
                    BEGIN
                      SCI.BIOLOGY.LOC2 := T_GRADE.LOC2;
                      IF T_GRADE.RPT > "2" THEN
                        BEGIN
                          SCI.BIOLOGY.LOC3 := T_GRADE.LOC3;
                          IF T_GRADE.RPT > "3" THEN
                            BEGIN
                              SCI.BIOLOGY.LOC4 := T_GRADE.LOC4;
                              IF T_GRADE.RPT > "4" THEN
                                SCI.BIOLOGY.LOC5 := T_GRADE.LOC5;
                            END;
                        END;
                    END;
                END
                ELSE
                  EXTRA_BASIC[COUNT]:= T_GRADE;
            END;
        END
        ELSE
          EXTRA_BASIC[COUNT]:= GRADE;
  END;

PROCEDURE PHY_PLACE(VAR AREA : STR1; VAR SCI : SCIENCE;
                    VAR GRADE : CLASS_REC; VAR FOUND : BOOLEAN;
                    VAR GRAND_TOTAL : INTEGER; COUNT : INTEGER;
                    VAR EXTRA_BASIC : BASIC_ARRAY);


VAR
  T_GRADE : CLASS_REC;
  T_AREA  : STR1;

BEGIN
  IF SCI.PHY_SCI_TOTAL < PHY_MAX THEN
    BEGIN
      FOUND := TRUE;
      SCI.PHYSICAL_SCIENCE.DEPT_CODE := GRADE.DEPT;
      SCI.PHYSICAL_SCIENCE.COURSE_NUM := GRADE.COURSE;
      SCI.PHYSICAL_SCIENCE.SEMESTER := GRADE.SEMESTER;
      SCI.PHYSICAL_SCIENCE.CRSE_GRADE := GRADE.GRADE;
      SCI.PHYSICAL_SCIENCE.CRSE_HOURS := GRADE.HOURS;
      SCI.PHYSICAL_SCIENCE.RPT := GRADE.RPT;
      SCI.PHYSICAL_SCIENCE.LOC1 := GRADE.LOC1;
      SCI.PHY_SCI_TOTAL := SCI.PHY_SCI_TOTAL+TRUNC(STOR(GRADE.HOURS));
      GRAND_TOTAL := GRAND_TOTAL+TRUNC(STOR(GRADE.HOURS));
      IF GRADE.RPT > "1" THEN
        BEGIN
          SCI.PHYSICAL_SCIENCE.LOC2 := GRADE.LOC2;
          IF GRADE.RPT > "2" THEN
            BEGIN
              SCI.PHYSICAL_SCIENCE.LOC3 := GRADE.LOC3;
              IF GRADE.RPT > "3" THEN
                BEGIN
                  SCI.PHYSICAL_SCIENCE.LOC4 := GRADE.LOC4;
                  IF GRADE.RPT > "4" THEN
                    SCI.PHYSICAL_SCIENCE.LOC5 := GRADE.LOC5;
                END;
            END;
        END;
    END
  ELSE
    IF GRADE.RPT > "1" THEN
      BEGIN
        FIND_LOC(GRADE,T_AREA,AREA);
        IF AREA <> " " THEN
          SELECT_AREA2(AREA,SCI,GRADE,FOUND,GRAND_TOTAL,
                       COUNT,EXTRA_BASIC);
      END
```

```
            SCI.PHYSICAL_SCIENCE.RPT := GRADE.RPT;
            IF GRADE.RPT > "3" THEN
              BEGIN
                SCI.PHYSICAL_SCIENCE.LOC4 := GRADE.LOC4;
                IF GRADE.RPT > "4" THEN
                  SCI.PHYSICAL_SCIENCE.LOC5 := GRADE.LOC5;
              END;
          END;
      END;
  END
ELSE
  IF GRADE.RPT > "1" THEN
    BEGIN
      FIND_LOC(GRADE,T_AREA,AREA);
      IF AREA <> " " THEN
        SELECT_AREA2(AREA,SCI,GRADE,FOUND,GRAND_TOTAL,
                     COUNT,EXTRA_BASIC);
    END
  ELSE
    IF SCI.PHYSICAL_SCIENCE.RPT > "1" THEN
      BEGIN
        T_GRADE := GRADE;
        GRADE.DEPT := SCI.PHYSICAL_SCIENCE.DEPT_CODE;
        GRADE.COURSE := SCI.PHYSICAL_SCIENCE.COURSE_NUM;
        GRADE.SEMESTER := SCI.PHYSICAL_SCIENCE.SEMESTER;
        GRADE.GRADE := SCI.PHYSICAL_SCIENCE.CRSE_GRADE;
        GRADE.HOURS := SCI.PHYSICAL_SCIENCE.CRSE_HOURS;
        GRADE.RPT := SCI.PHYSICAL_SCIENCE.RPT;
        GRADE.LOC1 := SCI.PHYSICAL_SCIENCE.LOC1;
        SCI.PHY_SCI_TOTAL := SCI.PHY_SCI_TOTAL-
                 TRUNC(STOR(GRADE.HOURS));
        GRAND_TOTAL := GRAND_TOTAL-TRUNC(STOR(GRADE.HOURS));
        GRADE.LOC2 := SCI.PHYSICAL_SCIENCE.LOC2;
        IF GRADE.RPT > "2" THEN
          BEGIN
            GRADE.LOC3 := SCI.PHYSICAL_SCIENCE.LOC3;
            IF GRADE.RPT > "3" THEN
              BEGIN
                GRADE.LOC4 := SCI.PHYSICAL_SCIENCE.LOC4;
                IF GRADE.RPT > "4" THEN
                  GRADE.LOC5 := SCI.PHYSICAL_SCIENCE.LOC5
                ELSE
                  GRADE.LOC5 := '    ';
              END
            ELSE
              BEGIN
                GRADE.LOC4 := '    ';
                GRADE.LOC5 := '    ';
              END;
          END
        ELSE
          BEGIN
            GRADE.LOC3 := '    ';
            GRADE.LOC4 := '    ';
            GRADE.LOC5 := '    ';
          END;
        FIND_LOC(GRADE,T_AREA,AREA);
        IF AREA <> " " THEN
          BEGIN
           SELECT_AREA2(AREA,SCI,GRADE,FOUND,GRAND_TOTAL,
                        COUNT,EXTRA_BASIC);
            IF FOUND THEN
              BEGIN
                SCI.PHYSICAL_SCIENCE.DEPT_CODE := T_GRADE.DEPT;
                SCI.PHYSICAL_SCIENCE.COURSE_NUM := T_GRADE.COURSE;
                SCI.PHYSICAL_SCIENCE.SEMESTER := T_GRADE.SEMESTER;
                SCI.PHYSICAL_SCIENCE.CRSE_GRADE := T_GRADE.GRADE;
                SCI.PHYSICAL_SCIENCE.CRSE_HOURS := T_GRADE.HOURS;
                SCI.PHYSICAL_SCIENCE.RPT := T_GRADE.RPT;
                SCI.PHYSICAL_SCIENCE.LOC1 := T_GRADE.LOC1;
                SCI.PHY_SCI_TOTAL :=
                  SCI.PHY_SCI_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                GRAND_TOTAL :=
                  GRAND_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                IF T_GRADE.RPT > "1" THEN
                  BEGIN
                    SCI.PHYSICAL_SCIENCE.LOC2 := T_GRADE.LOC2;
                    IF T_GRADE.RPT > "2" THEN
                      BEGIN
                        SCI.PHYSICAL_SCIENCE.LOC3 := T_GRADE.LOC3;
                        IF T_GRADE.RPT > "3" THEN
                          BEGIN
                            SCI.PHYSICAL_SCIENCE.LOC4 :=
                              T_GRADE.LOC4;
```

```
VAR
                    SCI.PHYSICAL_SCIENCE.CRSE_HOURS := T_GRADE.HOURS;
                    SCI.PHYSICAL_SCIENCE.RPT := T_GRADE.RPT;
                    SCI.PHYSICAL_SCIENCE.LOC1 := T_GRADE.LOC1;
                    SCI.PHY_SCI_TOTAL :=
                      SCI.PHY_SCI_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                    GRAND_TOTAL :=
                      GRAND_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                    IF T_GRADE.RPT > "1" THEN
                      BEGIN
                        SCI.PHYSICAL_SCIENCE.LOC2 := T_GRADE.LOC2;
                        IF T_GRADE.RPT > "2" THEN
                          BEGIN
                            SCI.PHYSICAL_SCIENCE.LOC3 := T_GRADE.LOC3;
                            IF T_GRADE.RPT > "3" THEN
                              BEGIN
                                SCI.PHYSICAL_SCIENCE.LOC4 :=
                                  T_GRADE.LOC4;
                                IF T_GRADE.RPT > "4" THEN
                                  SCI.PHYSICAL_SCIENCE.LOC5 :=
                                    T_GRADE.LOC5;
                              END;
                          END;
                      END;
                  END
                  ELSE
                    EXTRA_BASIC[COUNT]:= T_GRADE;
              END;
          END
          ELSE
            EXTRA_BASIC[COUNT]:= GRADE;
END;

PROCEDURE MAT_PLACE(VAR AREA : STR1; VAR SCI : SCIENCE;
                    VAR GRADE : CLASS_REC; VAR FOUND : BOOLEAN;
                    VAR GRAND_TOTAL : INTEGER; COUNT : INTEGER;
                    VAR EXTRA_BASIC : BASIC_ARRAY);

VAR
  T_GRADE : CLASS_REC;
  T_AREA  : STR1;

BEGIN
  IF SCI.SCI_MATH_TOTAL < MAT_MAX THEN
    BEGIN
      FOUND := TRUE;
      SCI.MATH.DEPT_CODE := GRADE.DEPT;
      SCI.MATH.COURSE_NUM := GRADE.COURSE;
      SCI.MATH.SEMESTER := GRADE.SEMESTER;
      SCI.MATH.CRSE_GRADE := GRADE.GRADE;
      SCI.MATH.CRSE_HOURS := GRADE.HOURS;
      SCI.MATH.RPT := GRADE.RPT;
      SCI.MATH.LOC1 := GRADE.LOC1;
      SCI.MATH_TOTAL := SCI.MATH_TOTAL+TRUNC(STOR(GRADE.HOURS));
      GRAND_TOTAL := GRAND_TOTAL+TRUNC(STOR(GRADE.HOURS));
      IF GRADE.RPT > "1" THEN
        BEGIN
          SCI.MATH.LOC2 := GRADE.LOC2;
          IF GRADE.RPT > "2" THEN
            BEGIN
              SCI.MATH.LOC3 := GRADE.LOC3;
              IF GRADE.RPT > "3" THEN
                BEGIN
                  SCI.MATH.LOC4 := GRADE.LOC4;
                  IF GRADE.RPT > "4" THEN
                    SCI.MATH.LOC5 := GRADE.LOC5;
                END;
            END;
        END;
    END
    ELSE
      IF GRADE.RPT > "1" THEN
        BEGIN
          FIND_LOC(GRADE,T_AREA,AREA);
          IF AREA <> " " THEN
            SELECT_AREA2(AREA,SCI,GRADE,FOUND,GRAND_TOTAL,
                         COUNT,EXTRA_BASIC);
        END
      ELSE IF SCI.MATH.RPT > "1" THEN
        BEGIN
          T_GRADE := GRADE;
          GRADE.DEPT := SCI.MATH.DEPT_CODE;
          GRADE.COURSE := SCI.MATH.COURSE_NUM;
          GRADE.SEMESTER := SCI.MATH.SEMESTER;
          GRADE.GRADE := SCI.MATH.CRSE_GRADE;
```

```
                END;
            END;
        END;
    END
  ELSE
    IF GRADE.RPT > "1" THEN
      BEGIN
        FIND_LOC(GRADE,T_AREA,AREA);
        IF AREA <> " " THEN
          SELECT_AREA2(AREA,SCI,GRADE,FOUND,GRAND_TOTAL,
                       COUNT,EXTRA_BASIC);
      END
    ELSE IF SCI.MATH.RPT > "1" THEN
      BEGIN
        T_GRADE := GRADE;
        GRADE.DEPT := SCI.MATH.DEPT_CODE;
        GRADE.COURSE := SCI.MATH.COURSE_NUM;
        GRADE.SEMESTER := SCI.MATH.SEMESTER;
        GRADE.GRADE := SCI.MATH.CRSE_GRADE;
        GRADE.HOURS := SCI.MATH.CRSE_HOURS;
        GRADE.RPT := SCI.MATH.RPT;
        GRADE.LOC1 := SCI.MATH.LOC1;
        SCI.MATH_TOTAL := SCI.MATH_TOTAL-TRUNC(STOR(GRADE.HOURS));
        GRAND_TOTAL := GRAND_TOTAL-TRUNC(STOR(GRADE.HOURS));
        GRADE.LOC2 := SCI.MATH.LOC2;
        IF GRADE.RPT > "2" THEN
          BEGIN
            GRADE.LOC3 := SCI.MATH.LOC3;
            IF GRADE.RPT > "3" THEN
              BEGIN
                GRADE.LOC4 := SCI.MATH.LOC4;
                IF GRADE.RPT > "4" THEN
                  GRADE.LOC5 := SCI.MATH.LOC5
                ELSE
                  GRADE.LOC5 := '    ';
              END
            ELSE
              BEGIN
                GRADE.LOC4 := '    ';
                GRADE.LOC5 := '    ';
              END;
          END
        ELSE
          BEGIN
            GRADE.LOC3 := '    ';
            GRADE.LOC4 := '    ';
            GRADE.LOC5 := '    ';
          END;
        FIND_LOC(GRADE,T_AREA,AREA);
        IF AREA <> " " THEN
          BEGIN
            SELECT_AREA2(AREA,SCI,GRADE,FOUND,GRAND_TOTAL,
                         COUNT,EXTRA_BASIC);
            IF FOUND THEN
              BEGIN
                SCI.MATH.DEPT_CODE := T_GRADE.DEPT;
                SCI.MATH.COURSE_NUM := T_GRADE.COURSE;
                SCI.MATH.SEMESTER := T_GRADE.SEMESTER;
                SCI.MATH.CRSE_GRADE := T_GRADE.GRADE;
```

```
                SCI.MATH.CRSE_HOURS := T_GRADE.HOURS;
                SCI.MATH.RPT := T_GRADE.RPT;
                SCI.MATH.LOC1 := T_GRADE.LOC1;
                SCI.MATH_TOTAL := SCI.MATH_TOTAL+
                        TRUNC(STOR(T_GRADE.HOURS));
                GRAND_TOTAL :=
                  GRAND_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                IF T_GRADE.RPT > "1" THEN
                  BEGIN
                    SCI.MATH.LOC2 := T_GRADE.LOC2;
                    IF T_GRADE.RPT > "2" THEN
                      BEGIN
                        SCI.MATH.LOC3 := T_GRADE.LOC3;
                        IF T_GRADE.RPT > "3" THEN
                          BEGIN
                            SCI.MATH.LOC4 := T_GRADE.LOC4;
                            IF T_GRADE.RPT > "4" THEN
                              SCI.MATH.LOC5 := T_GRADE.LOC5;
                          END;
                      END;
                  END;
              END
              ELSE
                EXTRA_BASIC[COUNT]:= T_GRADE;
          END;
      END
      ELSE
        EXTRA_BASIC[COUNT]:= GRADE;
END;

PROCEDURE ELEC_PLACE(VAR AREA : STR1; VAR SCI : SCIENCE;
                     VAR GRADE : CLASS_REC; VAR FOUND : BOOLEAN;
                     VAR GRAND_TOTAL : INTEGER);
VAR
  T_GRADE : CLASS_REC;
  T_AREA  : STR1;

BEGIN
  IF SCI.ELEC_TOTAL < ELEC_MAX THEN
    BEGIN
      FOUND := TRUE;
      SCI.MATH_ELECTIVES.DEPT_CODE := GRADE.DEPT;
      SCI.MATH_ELECTIVES.COURSE_NUM := GRADE.COURSE;
      SCI.MATH_ELECTIVES.SEMESTER := GRADE.SEMESTER;
      SCI.MATH_ELECTIVES.CRSE_GRADE := GRADE.GRADE;
      SCI.MATH_ELECTIVES.CRSE_HOURS := GRADE.HOURS;
      SCI.MATH_ELECTIVES.RPT := GRADE.RPT;
      SCI.MATH_ELECTIVES.LOC1 := GRADE.LOC1;
      SCI.ELEC_TOTAL := SCI.ELEC_TOTAL+TRUNC(STOR(GRADE.HOURS));
      GRAND_TOTAL := GRAND_TOTAL+TRUNC(STOR(GRADE.HOURS));
      IF GRADE.RPT > "1" THEN
        BEGIN
          SCI.MATH_ELECTIVES.LOC2 := GRADE.LOC2;
          IF GRADE.RPT > "2" THEN
            BEGIN
              SCI.MATH_ELECTIVES.LOC3 := GRADE.LOC3;
              IF GRADE.RPT > "3" THEN
                BEGIN
                  SCI.MATH_ELECTIVES.LOC4 := GRADE.LOC4;
                  IF GRADE.RPT > "4" THEN
                    SCI.MATH_ELECTIVES.LOC5 := GRADE.LOC5;
                END;
            END;
        END;
    END
  ELSE
    IF GRADE.RPT > "1" THEN
      BEGIN
        FIND_LOC(GRADE,T_AREA,AREA);
        IF AREA <> " " THEN
          SELECT_AREA2(AREA,SCI,GRADE,FOUND,GRAND_TOTAL,
                       COUNT,EXTRA_BASIC);
      END
    ELSE
      IF SCI.MATH_ELECTIVES.RPT > "1" THEN
        BEGIN
          T_GRADE := GRADE;
          GRADE.DEPT_CODE := SCI.MATH_ELECTIVES.DEPT;
          GRADE.COURSE_NUM := SCI.MATH_ELECTIVES.COURSE;
          GRADE.SEMESTER := SCI.MATH_ELECTIVES.SEMESTER;
```

```
          END;
        END;
    END
  ELSE
    IF GRADE.RPT > "1" THEN
      BEGIN
        FIND_LOC(GRADE,T_AREA,AREA);
        IF AREA <> " " THEN
          SELECT_AREA2(AREA,SCI,GRADE,FOUND,GRAND_TOTAL,
                       COUNT,EXTRA_BASIC);
      END
    ELSE
      IF SCI.MATH_ELECTIVES.RPT > "1" THEN
        BEGIN
          T_GRADE := GRADE;
          GRADE.DEPT_CODE := SCI.MATH_ELECTIVES.DEPT;
          GRADE.COURSE_NUM := SCI.MATH_ELECTIVES.COURSE;
          GRADE.SEMESTER := SCI.MATH_ELECTIVES.SEMESTER;
          GRADE.GRADE := SCI.MATH_ELECTIVES.CRSE_GRADE;
          GRADE.HOURS := SCI.MATH_ELECTIVES.CRSE_HOURS;
          GRADE.RPT := SCI.MATH_ELECTIVES.RPT;
          GRADE.LOC1 := SCI.MATH_ELECTIVES.LOC1;
          SCI.ELEC_TOTAL :=
            SCI.ELEC_TOTAL-TRUNC(STOR(SCI.MATH_ELECTIVES.HOURS));
          GRAND_TOTAL := GRAND_TOTAL-TRUNC(STOR(GRADE.HOURS));
          GRADE.LOC2 := SCI.MATH_ELECTIVES.LOC2;
          IF SCI.MATH_ELECTIVES.RPT > "2" THEN
            BEGIN
              GRADE.LOC3 := SCI.MATH_ELECTIVES.LOC3;
              IF SCI.MATH_ELECTIVES.RPT > "3" THEN
                BEGIN
                  GRADE.LOC4 := SCI.MATH_ELECTIVES.LOC4;
                  IF SCI.MATH_ELECTIVES.RPT > "4" THEN
                    GRADE.LOC5 := SCI.MATH_ELECTIVES.LOC5
                  ELSE
                    GRADE.LOC5 := '    ';
                END
              ELSE
                BEGIN
                  GRADE.LOC4 := '    ';
                  GRADE.LOC5 := '    ';
                END;
            END
          ELSE
            BEGIN
              GRADE.LOC3 := '    ';
              GRADE.LOC4 := '    ';
              GRADE.LOC5 := '    ';
            END;
          FIND_LOC(GRADE,T_AREA,AREA);
          I AREA <> " " THEN
            BEGIN
              SELECT_AREA2(AREA,SCI,GRADE,FOUND,GRAND_TOTAL,
                           COUNT,EXTRA_BASIC);
              IF FOUND THEN
                BEGIN
                  SCI.MATH_ELECTIVES.DEPT_CODE := T_GRADE.DEPT;
                  SCI.MATH_ELECTIVES.COURSE_NUM := T_GRADE.COURSE;
                  SCI.MATH_ELECTIVES.SEMESTER := T_GRADE.SEMESTER;
                  SCI.MATH_ELECTIVES.CRSE_GRADE := T_GRADE.GRADE;
                  SCI.MATH_ELECTIVES.CRSE_HOURS :=
                    T_GRADE.HOURS;
                  SCI.MATH_ELECTIVES.RPT := T_GRADE.RPT;
                  SCI.MATH_ELECTIVES.LOC1 := T_GRADE.LOC1;
                  SCI.ELEC_TOTAL :=
                    SCI.ELEC_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                  GRAND_TOTAL :=
                    GRAND_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                  IF T_GRADE.RPT > "1" THEN
                    BEGIN
                      SCI.MATH_ELECTIVES.LOC2 := T_GRADE.LOC2;
                      IF T_GRADE.RPT > "2" THEN
                        BEGIN
                          SCI.MATH_ELECTIVES.LOC3 := T_GRADE.LOC3;
                          IF T_GRADE.RPT > "3" THEN
                            BEGIN
                              SCI.MATH_ELECTIVES.LOC4 := T_GRADE.LOC4;
                              IF T_GRADE.RPT > "4" THEN
                                SCI.MATH_ELECTIVES.LOC5 :=
                                  T_GRADE.LOC5;
                            END;
                        END;
                    END;
                END
```

```
                    GRAND_TOTAL :=
                      GRAND_TOTAL+TRUNC(STOR(T_GRADE.HOURS));
                    IF T_GRADE.RPT > "1" THEN
                      BEGIN
                        SCI.MATH_ELECTIVES.LOC2 := T_GRADE.LOC2;
                        IF T_GRADE.RPT > "2" THEN
                          BEGIN
                            SCI.MATH_ELECTIVES.LOC3 := T_GRADE.LOC3;
                            IF T_GRADE.RPT > "3" THEN
                              BEGIN
                                SCI.MATH_ELECTIVES.LOC4 := T_GRADE.LOC4;
                                IF T_GRADE.RPT > "4" THEN
                                  SCI.MATH_ELECTIVES.LOC5 :=
                                    T_GRADE.LOC5;
                              END;
                          END;
                      END;
                  END
                  ELSE
                    EXTRA_BASIC[COUNT]:= T_GRADE;
              END;
          END
          ELSE
            EXTRA_BASIC[COUNT]:= GRADE;
END;


PROCEDURE SELECT_AREA2;
VAR
  TEMP : CHAR;

BEGIN
  SUBSTR(AREA,1,1,TEMP);
  CASE TEMP OF
    '1' : BIO_PLACE(AREA,SCI,GRADE,FOUND,GRAND_TOTAL,COUNT,
                    EXTRA_BASIC);
    '2' : PHY_PLACE(AREA,SCI,GRADE,FOUND,GRAND_TOTAL,COUNT,
                    EXTRA_BASIC);
    '3' : MAT_PLACE(AREA,SCI,GRADE,FOUND,GRAND_TOTAL,COUNT,
                    EXTRA_BASIC);
    '4' : ELEC_PLACE(AREA,SCI,GRADE,FOUND,GRAND_TOTAL,COUNT,
                    EXTRA_BASIC);
    END;
END;


BEGIN
   FOUND := FALSE;
   IF SCI.SCI_MATH_TOTAL < SCI_MATH_MAX THEN
   BEGIN SUBSTR(GRADE.LOC1,2,1,AREA);
     SELECT_AREA2(AREA,SCI,GRADE,FOUND,GRAND_TOTAL,COUNT,
   END
   ELSE
     EXTRA_BASIC[COUNT]:= GRADE;
END;
```