A MASTER'S DEGREE COURSE OF STUDY DATABASE

by

GEORGE RICHARD HUGHES

B.A., Kansas University, 1973

M.A., University of Regina, 1974

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas
1979

Approved by:

[illegible]

Major Professor

TABLE OF CONTENTS

List of Figures

CHAPTER I

INTRODUCTION

1.1 History

There has been an ongoing effort in this University to automate certain student records. The college of Home Economics has developed a program which will verify what Bachelor's degree course requirements are being met by their major students. This program helps faculty members advise students in planning course loads for future semesters. This program was implemented using PL/1, with no provisions made for any sort of database management system (DBMS). [4]

Although several Home Economics departments at other institutions have expressed interest in duplicating this implementation, it suffers from being used in the batch mode only. Additionally, this program does not lend itself easily to modification, and given frequent curricula changes at both the departmental and College levels, this is a serious complaint.

Voelz and Garland [8] designed a prototype system using a DBMS that would contain University-wide academic records. This implementation utilized the Integrated Data Base Management System (IDMS) [2] , and was an adequate starting point. The commitment to a DBMS was important because the data can be described, stored, and manipulated independently of different users. Voelz and Garland's design proved to be too large, expensive, and wide ranging for any Computer Science Department applications.

In 1976, Long [4] constructed an IDMS database that extended upon the concepts of Voelz and Garland's scheme. Long was commissioned to make an economical system which would act as a counselling aid for the Computer Science faculty. Other requirements to be met by this database included Bachelor of Science requirements, major course requirements in Computer Science, some personal information on the student, and courses completed and/or currently enrolled in. This program also used the batch mode on an IBM 370 mainframe.

1.2 Preliminary Guidelines

The evolution of computer programs that automate both academic record keeping and student advising procedures continues. During discussions last Fall, the need was expressed for a prototype system which would go beyond the efforts mentioned above. The major contraints upon this system were:

1. the system must be available to interactive display terminal users
2. the DBMS must provide independence and security
3. the system would advise and help Master's students in Computer Science
4. the prototype must be cost effective

These constraints necessitated using the Department´s Interdata 8/32 minicomputer, and a conversational DBMS system INFO32 [3] , as the hardware and software support, respectively, for this project. The choice of the 8/32 reflected the cost constraint, as the overall database would be cheaper to operate on this machine compared with an implementation on the Kansas State Computing Center´s Itel AS-5. The smaller machine had the added benefits of terminal display mode, and convenient access for Computer Science students.

Other reasons for choosing the Interdata mini were that the Department would like to encourage the use of the 8/32 by out of town users via telephone links. Furthurmore, the Department has a continuing research interest in database applications for minicomputers over distributed networks.

INFO32 is a software product of Henco, Inc. [3]. Primarily a business oriented system, INFO32 uses English keywords as language command instructions for data entry and update, query formulation, and report writing. INFO32 was selected as the DBMS because it works well in a terminal display environment, was available on the 8/32, and lent itself easily to a frontend interface [5] . This interface will provide much of the data security and independence, as it masks the user from the actual INFO32 database. It will be discussed at a later time in this report.

1.3 Controlling Purpose

The controlling purpose of this report is to show the design and implementation of this Master´s degree database prototype project. We have already looked at the background of previous attempts in this area, and the renewed interest in providing new informational services to prospective and enrolled graduate students across the state.

In Section II, we will describe the specific requirements that this project will try to fulfill. In Section III, the database design will be discussed, and the assertions that can be made about the various data items. Other program parts within INFO32 that manipulate this database will also be described in this section.

We will outline the general data independence qualities that the backend interface between the user and the database will support. A summary of this report will be presented, as will a look at future directions for this type of work. Finally, the appendices document the user and supervisory manuals needed to modify or keep this implementation functioning.

CHAPTER II

SPECIFICATION REQUIREMENTS

2.1 Listing of Requirements

The impact of the initial constraints was felt mostly in the selection of physical devices and software packages. These constraints did not really effect the design or logical organization of the fledgling project. Section II will be an amplification of the overall purposes of the project, and a complete statement of specifications that will be realized in this implementation.

This project will be comprised of up-to-date, but un-official academic and personal facts about Master's students in Computer Science. Several different purposes are suggested by the specifications that follow. The first purpose is informational. Students can choose to view reports about admission requirements, equivalent experience in this discipline needed at the outset of graduate work, or other matters relating to the Master's program in Computer Science at Kansas State University.

Secondly, this implementation will securely store courses completed or enrolled in for individual students. The credit hours and grade received for each course, and the semester when the class was taken can also be entered into the database. The student may, on demand, request that his grade point average be calculated, or have his list of courses evaluated for degree requirement satisfaction.

The third purpose of this project is to duplicate certain Departmental personal files. Faculty members can be spared the tedium of shuffling through file folders when checking on the progress of a graduate student.

These three purposes helped to shape the following required specifications:

1. Interactive capability for Manhattan, and out lying users
    - 1.a. access informational text files
    - 1.b. one academic record stored in the database per user
        - 1.b.1. user can create this record
        - 1.b.2. user can update items in this record
        - 1.b.3. user can read through parts of this created record
2. Database Administrator
    - 2.a. purges inactive records
    - 2.b. supervises the maintenance of the system
    - 2.c. modifies the system due to necessary changes
3. Interface that will monitor communication between the user and the INFO32 database
    - 3.a. controls user creation, updating, and reading

- 3.b. controls user access to informational text files
- 3.c. provides some type checking on data
- 3.d. provides for data independence and security

4. Individual Student records
    - 4.a. name, address, etc.
    - 4.b. city where enrolled
    - 4.c. means of financial support (only important if the student is a Computer Science Department employee)
    - 4.d. courses taken or currently enrolled in
        - 4.d.1. course number
        - 4.d.2. letter grade
        - 4.d.3. credit hours
        - 4.d.4. semester of enrollment

5. Queries allowed on the database record
    - 5.a. grade point average calculation
    - 5.b. student's position with regards to the 30 graduate credit hour minimum
    - 5.c. 'C' and 'D' credit hour deficiency
    - 5.d. curriculum checks for required courses
        - 5.d.1. core curriculum classes
        - 5.d.2. upper level class
        - 5.d.3. graduate seminar

6. Other Information
    - 6.a. Department admission requirements

6.b. Department MS requirements

6.c. Bachelor of Science equivalent experience one should have, or undergraduate classes to be made up upon acceptance into this Master's program

6.d. current and following semester schedules

6.e. Kansas State Computer Science faculty biographical sketches

6.f. courses acceptable for MS degree credit

6.f.1. pre-requisites

6.f.2. content of courses

6.f.3. credit hours for the course

7. No concurrent use of the database, at least for the time being

2.2 Relationships Between System Components

At this point, it would be best to explain the interrelationships between all the different parts of the system that will implement these specifications.

The MTM operating system in the Interdata 8/32 oversees and supervises any communication between the component parts. Its task is to provide the environment by which these parts can utilize the minicomputer's resources.

The DBMS, INFO32, controls all messages regarding the database itself. The INFO32 system only allows qualified users making legal queries to manipulate any part of the database.

This underscores the fact that the Fortran interface's [6] main job is the interception and interpretation of communications between a terminal user, INFO32, and ultimately, the INFO32 database COURSE. The interface lets the user choose from a limited set of options. The messages outlining these options are sent to the user, the user chooses a particular task, and the interface executes the necessary steps to carry out that task.

The interface, in this way, receives and gives messages to both the DBMS and the student user. The text files are chosen individually by the student, the interface learns of the choice, and then extracts for display the correct file.

Finally, the student user can only send messages directly to the interface, while receiving information from the DBMS programs, the text files, and the monitoring interface. The supervisory user can use the database by circumventing the interface, and dealing directly with INFO32. On the systems level, the supervisor can maintain and create any text files.

A graphical representation of the system components can be seen in Figure 2.1.

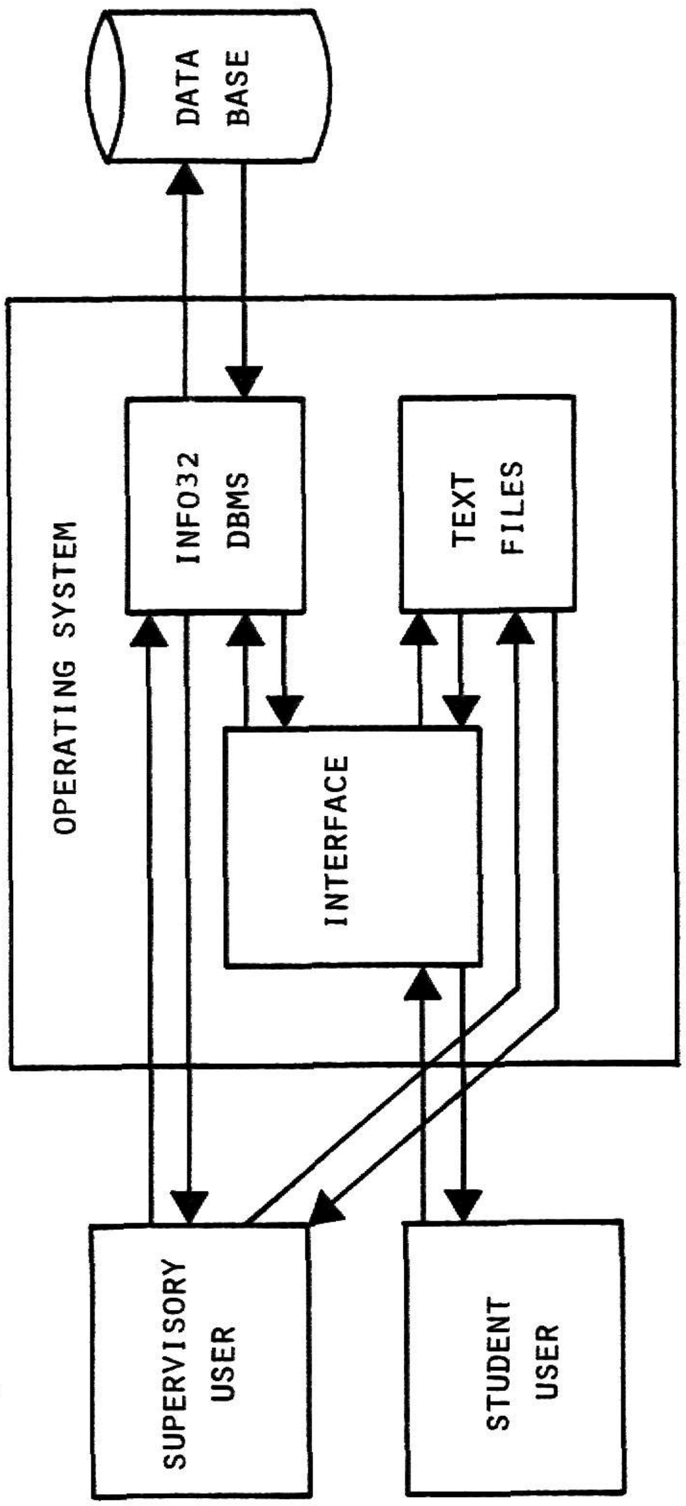
DATA BASE
OPERATING SYSTEM
INFO32 DBMS
TEXT FILES
INTERFACE
SUPERVISORY USER
STUDENT USER

FIGURE 2.1

CHAPTER III

Description of Functions

The next step is to put flesh on the skeleton of existing requirement specifications and constraints. This chapter contains a description of the logical organization of various program segments. The tasks involved here are either the database itself, or command stream programs within the INFO32 file directory which can act upon the currently selected student record.

The latter type of program includes GPA1, CHECK, CHECK2, and CLASSES. The more detailed functional specifications on these tasks will appear in the Appendix.

3.1 User Operations

There are four primary operations that a user, with the interface's help, can make on the database COURSE. A new student can enter the suggested personal information into his own record. Secondly, a continuing student can make updates on selected database items, by either changing the original entries, or entering information that is new, as would be the case after a completed semester.

At any point in the interactive session, the user can choose to view his database record in its entirety.

Finally, the user can request operations that query his record for curriculum checks and grade point average. These user operations suggest the following definitions for the database.

3.2 The Database

The database's chief feature is the reproduction and automation of a graduate student's Departmental file. To this end, many of the data items are fairly straight forward, such as first, last, and middle names; and street, city, state, and zip code for composing addresses. Although ZIP is not yet used for any calculations, it was thought that an integer and not a character representation would be advisable. We don't want to confuse the Postal Service with any 'ABCDE' zip codes.

The database now consists of 82 separate data items or data fields, which amount to a total of 373 bytes. As can be seen from Figure 3.1, more than two thirds (or 56) of these data fields consist of the record keeping apparatus for storing 14 different courses, grades, credit hours, and semester of enrollment.

The association of each CLASS-xx, GRADE-xx, HOURS-xx, and ENROL-xx is made by the last two character places on these data item names. For example, CLASS-09 is associated with GRADE-09, and with HOURS-09, and so on. The choice of

14 different courses was an arbitrary decision based on the fact that most graduate classes are three credit hour courses. Since most graduate candidates do not greatly exceed the 30 credit hour requirement , 14 classes should be sufficient for this amount of course work.

Approximately 45% of the database storage requirements are needed because INFO32 does not have any dynamic allocation feature, and so these 56 static fields must always be in place. An array implementation with looping would have definitely shortened the command stream programs, which for the most part, manipulate these fields of classroom achievement.

To complete the discussion of the 14 repetitive class instances, data items CLASS-01 through CLASS-14 were chosen to be of type integer, with a range of 1 to 999999. The reason for integer typing for this item was that logical comparisons like 'greater than' or 'equal to' were needed in determining if degree course requirements had been met.

Database Definition of COURSE

ITEM-NAME	TYPE	LENGTH	POSITION
LAST	Character	15	1 to 15
FIRST	Character	10	16 to 25
MIDDLE	Character	10	26 to 35
SSN	Character	9	36 to 44
STREET	Character	20	45 to 64
CITY	Character	15	65 to 79
STATE	Character	2	80 to 81
ZIP	Integer	5	82 to 86
TELEPHONE	Character	12	87 to 98
WHERE-ENROLLED	Character	12	99 to 110
FULL-TIME	Character	1	111 to 111
START-OF-PROGRAM	Date	8	112 to 119
MASTERS-EXAM	Character	1	120 to 120
DATE-MS-EXAM	Date	8	121 to 128
ORALS-PASSED	Character	1	129 to 129
DATE-OF-ORALS	Date	8	130 to 137
DEGREE-SOUGHT	Character	3	138 to 140
SUPPORT-CLASS	Character	5	141 to 145
SUPPORT-START	Date	8	146 to 153
SUPPORT-SOURCE	Character	10	154 to 163
SUPPORT-TENTH	Integer	3	164 to 166
ADVISOR	Character	15	167 to 181
MAJOR-PROF	Character	15	182 to 196
GPA	Numeric	4,2	197 to 200
TOTAL-HOURS	Integer	2	201 to 202
TOTAL-CREDITS	Integer	3	203 to 205
CLASS-01	Integer	6	206 to 211
GRADE-01	Character	1	212 to 212
HOURS-01	Integer	1	213 to 213
ENROL-01	Character	4	214 to 217
. . . .			
CLASS-14	Integer	6	362 to 367
GRADE-14	Character	1	368 to 368
HOURS-14	Integer	1	369 to 369
ENROL-14	Character	4	370 to 373

Figure 3.1

Six integer digits were used, as a course like C. S. 720 must be designated as its full course number-- 286720-- because 720 is not a unique identifier. Any other department might have a 720 course number, so consequently all six digits are present.

Items GRADE-01 to GRADE-14 were set to be one byte character variables, so that the usual alphabetic marks could be recorded. This GRADE-xx series is not without its problems, as any other letter can inadvertantly be entered. In the cases of ´I´ for incomplete, or ´W´ for withdraw, this incorrect data entry seems at least plausible.

Since the DBMS and interface do not check for this type of error in data validity, the command stream programs generally treat courses with grades of anything but ´A´, ´B´, ´C´, or ´D´ as courses that have yet to be completed. Maybe by being informed that a course he is sure he has completed has not been taken, a student will be moved to recheck the grade entry for this course.

HOURS-01 to HOURS-14 are all one digit integer variables that contain the credit hours for that particular course. It would have been a small matter to check these values for accuracy, as most classes are worth three credit hours. But since there is a curriculum change being considered whereby emphasis courses like C. S. 700 are expanded to four credit hours, this testing was not done.

ENROL-01 through ENROL-14 identify the semester in which a class was taken. A four character shorthand code is the suggested valid entry, as Fall 1978 becomes 'FA78'. Similarly, 'SP' for Spring and 'SU' for Summer are the other acceptable combinations for the first two data places. The last two places are just the last two digits for that year. No verification is done with this field, as entries like 'WI' for Winter, or several Intersession terms that defy classification are certainly possible for the database user.

The personal information like name and address is not too controversial. Telephone, the items that comprise the address, and last name are updatable, as people move, and in a sometimes related action, change their last name.

Having the SSN, or social security number, as a length nine character string item in this database could be the source for data being compromised, and could even cause legal headaches. The initial design of this database used the SSN as the key by which a user would identify his existing record. The fallacy in using SSN as the key to selecting this database record is that (especially at a school like Kansas State where social security numbers are also student identification numbers) a social security number can often become public knowledge.

An unscrupulous person could then jeopardize a database record containing the SSN that he had somehow obtained. The response to this serious problem was to have the interface generate and display a five character hash key based on an individual's social security number upon record creation.

All subsequent referals to this record would only be allowed by the entering of this hash key at the beginning of the conversation. There is great room for improvement in the area of protecting this data.

The WHERE-ENROLLED variable was included so that Manhattan, Leavenworth, Wichita, or Kansas City students could be differientiated. Full time status is a simple one character ´Y´ or ´N´ entry, with presumably more Department attention being given to full time students.

In the case of FULL-TIME, as well as the other one character items MASTERS-EXAM and ORALS-PASSED, the ´Y´ or ´N´ type checking is done at data entry by the interface. An ´N´ response to these last two items will cause the interface to branch around DATE-MS-EXAM and DATE-OF-ORALS, respectively. To wit: if one hasn´t passed the orals exam, there isn´t much point in asking when that exam took place.

INFO32 does make provisions for date types. All dates are checked for validity by this system, with the format being MM/DD/YY, or 6/27/79. This will be the only practical type checking for dates because of the problem encountered when this prototype reaches tne twenty-first century. The answer would be to utilize INFO32´s 10 character date types. As for now, the saving of two bytes on each date is justification enough for this definition.

The DATE-MS-EXAM and the START-OF-PROGRAM dates can be used for gauging the degree progress of a student whose graduate Computer Science career has been done exclusively at Kansas State. A transfer student could distort the credibility of these two items by entering accurate dates of his accomplishments at other schools.

The DEGREE-SOUGHT item could be helpful in segregating Master's students, and PhD hopefuls. The advisor and MAJOR-PROF categories could possibly be redundant. But in the case of a student having a different advisor and major professor, this difference will be noted.

The SUPPORT- series of data items are primarily intended for use by the Department head. The SUPPORT-SOURCE, SUPPORT- CLASS, and SUPPORT-TENTH items can be used for bookkeeping, accounting summaries per source of funding, or even performance evaluation of SUPPORT-CLASS types like 'G.T.A' or 'G.R.A'.

Finally, no type checking is done on TOTAL-HOURS, GPA, or TOTAL-CREDITS, other than the usual integer versus character checking INFO32 always does.

The GPA1 program will figure a new value for TOTAL-HOURS, TOTAL-CREDITS, and GPA, and return the new value to the student's tuple. A final note on GPA and real numbers: INFO32 must count the decimal point as a digit place. In order to get the familar grade point form of x.xx, GPA had to be defined as a number of length 4, with two decimal digits.

In summation, the COURSE database is a static entity containing 82 different data items. It contains no links, as would be the case in a network model, and consists of one schema and one subschema. This database is one big collection of these different data items and relationships. An individual's record represents the academic and personal facts of one student, keyed upon by social security number, and protected by the hash key password.

This design decision of making one database with many records was necessary because INFO32 doesn't provide for making duplicate database copies off of a model database definition for every different user. INFO32 does not allow any concurrent operations, so the database can now be accessed by only one user at a time.

3.3 Program GPA1

Program GPA1 is a command stream program embedded in the INFO32 DBMS. It is a sequential program using IF statements in processing the selected database record for grade point average, total grade credits, and total credit hours. Additionally, the credit hours of grades 'C' and 'D', and the relationship of total credits of that user to the 30 graduate credit hour requirement will be computed. It is the job of the interface to select just that record of the database that is relevant to the current user of GPA1's functions.

The GPA1 section will only keep a running total of total credit hours and total grade credits for these courses at least at the 600 level in Computer Science. The C. S. 891

intensive course for non-major graduate students is the lone exception to this rule.

These two running totals will only be made on a course whose GRADE-xx is explicitly an 'A', 'B', 'C', 'D', or 'F'. In the last case, hours of 'F' must be added to total credit hours temporarily for grade point calculations, but left out of total credit hours when that variable is subtracted from the magic number of 30. If 'F' credit hours were not subtracted at this point, the user with any 'F' marks would get a distorted picture of how close to graduation he might be.

Thus the GPA data is limited to completed graduate courses in Computer Science at Kansas State. This differs from the official admissions and record computation by only graduate courses completed in other departments. This might seem to be a trivial matter, since at most six credit hours can be taken outside of the Computer Science curriculum.

I did not see any easy way in INFO32 to account for these possible credits from other departments. INFO32 has no character string functions with which acceptable graduate courses could be excised for GPA1's attention. So I choose to eliminate from GPA1's consideration any classes which did not conform to the domain of Computer Science graduate courses, because of the awkward nature of type integer for CLASS-xx that would adversely affect performance, and because of the absence of character string manipulation in INFO32 that could have been helpful.

Also unresolved is the problem of accepting for credit courses completed at other institutions. The Kansas State Graduate Office is, of course, the final arbitrar in these matters, so perhaps database users of this type should be instructed to write that office. As it stands now, there is no allowance for accurately interpretting course numbers from other schools that may gain entry in a student´s tuple, although these transfer credits will not be included in GPA1´s processing.

On the subject of lower grades, a successful Master´s candidate must have at least three times the number of credit hours of ´A´ or ´B´ hours, as he has in credit hours of ´C´ or ´D´. GPA1 will make known this deficiency due to too many ´C´ or ´D´ hours. A warning message will be displayed by GPA1 in the event that a student´s grades are lacking in this way.

Other display messages given by this program are the grade point average itself, and how many credits remain for graduation for that person. GPA, TOTAL-CREDITS, and TOTAL-HOURS are all returned back in updated form to the student´s record in COURSE. A more detailed presentation, and a source code listing of each of these programs can be found in the Appendix.

3.4 Program CHECK

The CHECK command stream program has responsibility in testing a user's list of classes for completion of the major emphasis courses in Computer Science. The four emphasis courses are C.S. 640, Introduction to Software Engineering; C.S. 700, Translator Design I; C.S. 720, Operating Systems II; and C.S. 760 or C.S. 761, Database Management Systems.

This program was divided with the other curriculum checking program CHECK2, because with only IF statements at my disposal, the program was getting rather long. To provide some degree of modularization, and improve response time, CHECK2 will test for completion of the graduate seminar, as well as the upper level graduate class requirements.

CHECK can take advantage of the fact that courses will be entered by the interface beginning with CLASS-01, and will proceed in order to CLASS-14. This allows for a nesting arrangement of IF statements which test for any numerical value while walking through the sequence of CLASS-xx items.

In other words, if three classes have been entered, the IF test for CLASS-04 will fail, control will 'drop out', and no other CLASS-xx items will be evaluated. In very unscientific testing, it was observed that this nesting approach was found to cut response time for this program by about half, as compared to evaluating all 14 CLASS-xx fields whether they need to be tested or not.

After the CHECK process determines that a CLASS-xx item is active, it next examines the GRADE-xx value in the same level of association, and determines if a passing mark has been recorded. No assumptions about completion of a class are made about either classes currently enrolled in, or incomplete classes.

Once there is an active class with a passing grade, it is then a matter of testing that class for each of the required emphasis courses. If there is a match under these circumstances, a counter is incremented in the appropriate fashion for that course. The logical details of this operation are spelled out in the Appendix.

After evaluating as many classes as are active, CHECK will output a message to the student describing exactly what emphasis classes he has completed, and what emphasis classes remain to be finished.

3.5 Program CHECK2

Program CHECK2 is similar in form to CHECK. A nesting strategy is again employed that only tests those CLASS-xx items that contain values. A passing grade in an active course is also demanded before any further tests are made. The courses of interest to this program are the graduate seminar, C.S. 897, and a dozen or so courses that have C.S. graduate courses as pre-requisites.

The courses in this latter group that do satisfy the upper level requirement are C.S. 725, Computer Networks; C.S. 750, Advanced Computer Architecture Experiments; C.S.