01 Jul 1986

Genesis of an Expert System For UMR Degree Auditing

Ruth Sue Dare

Arlan R. Dekock
Missouri University of Science and Technology, adekock@mst.edu

GENESIS OF AN EXPERT SYSTEM FOR UMR DEGREE AUDITING

Ruth Sue Dare* and A. R. DeKock

CSc-86-3

Department of Computer Science

University of Missouri-Rolla

Rolla, Missouri 65401 (314) 341-4491

*This report is substantially the M.S. thesis of the first author, completed July 1986.

ABSTRACT

This paper describes the features, design, and development of an expert system for degree auditing at the University of Missouri-Rolla. It summarizes artificial intelligence as it is known today while specifically addressing expert systems. It describes selected expert systems currently in existence.

The present audit procedure utilized at the University of Missouri-Rolla is discussed. A description is given of the design and development of an expert system, written in LISP, to conduct a degree audit. Finally there are concluding remarks which include an analysis of the system and a discussion of possible system enhancement.

ACKNOWLEDGEMENT

Sincere thanks is extended to Dr. Arlan DeKock for his advice and assistance during the development and writing of this thesis. Gratitude is also extended to my sister, Sherry L. Dare, for her encouragement and assistance during the typing of this document. Thanks is given the the Math and Computer Science Department at Northeast Missouri State University, Kirksville, Missouri for the use of computing facilities. The efforts of Dr. Bill Gillett and Dr. Myron Parry, as members of the thesis committee, are also greatly appreciated.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF TABLES

I. INTRODUCTION

A. SUMMARY OF ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) is a branch of computer science in which the objective is to create a computing system displaying traits comparable to behavior of intelligent humans. At first it may appear that all computers exhibit intelligent behavior in their performance of activities such as intricate calculations and extensive searches; however, let us define "intelligent behavior" as that which requires combining new and old information in a useful manner, utilizing rules of thumb and possibly intuition to achieve a meaningful response (Stevens, 1985). This paper does not concern itself with defining an expert when discussing expert systems. This definition is left to the reader.

The first AI programs were designed to play chess, solve puzzles, and perform translation of texts from one language to another. The birthdate of AI is the mid-1950's with Alan Turing named as the Father of AI (Barr, 1981).

Various problems are currently being researched within the realm of artificial intelligence including understanding natural language, general problem solving, game playing, vision and speech perception, robotics, theorem proving, and expert problem solving which is the topic of this thesis. Inherent in developing programs to solve problems of this

type is the ability for these systems to reason. Researchers hope that an intense look at intelligent systems will shed more light on the mental processes humans employ while solving problems.

Natural languages such as English, French, or Spanish are ones in which humans communicate by voice or the written page. Computers, however, deal more effectively with formal languages such as FORTRAN or COBOL where rigid syntax rules are applied. Creating systems capable of bridging the gap between formal language communication and that of natural language is one area being researched in AI. It is hoped that building computer systems to understand natural language will aid in more fully understanding the communication processes humans utilize.

General problem solving, another area of study in AI, pertains to the method individuals employ while doing an activity such as grocery shopping. The general problem solving program (GPS) was a beginning attempt to create a program capable of learning (Yazdani, 1984).

AI research is also being conducted in game playing. Work has been done in creating systems with the ability to play various games including checkers, chess, and tic-tac-toe. Games provide an ideal opportunity to study AI in that they afford the researcher an easy way to determine success by wins and losses while generally not requiring vast knowledge bases (Rich, 1983).

Vision and speech perception have also received attention from AI researchers. Here systems are created to manipulate objects "seen" oftentimes via a camera and objects "heard" via waveform analysis. Vision plays an integral role in developing the AI area of robotics.

Another area of action in artificial intelligence research is that of theorem proving or automatic deduction. A set of axioms provide the basis for a system to reason and prove theorems. The objective is to form a system capable of arriving at conclusions using the original axiom set, search procedures, and rules of inference. An expert system is a program or a suite of programs designed to accomplish a task that could only be correctly accomplished by an expert in the chosen field.

B. PSYCHOLOGICAL ISSUES

As the use of artificial intelligence abounds and expert systems are giving advice to professionals and laymen alike, several psychological issues begin to loom over the horizon. Some of these issues are recited in the ensuing paragraphs.

Moto-oka of the University of Tokyo and Chairman of the Advisory Committee of ICOT, Japan's Institute for New Generation Computer Technology, states:

> Computers are no longer used only for science and technology, business computations, and the automated control of industrial processes; they now have penetrated our daily

> lives and are becoming society's central nervous system. As the developed world moves from an industrial economy to an information economy, the social structure is undergoing its greatest change since the industrial revolution to adapt to the new technology (Moto-Oka, 1983).

Education is an area in which the impact of machine intelligence is being investigated. Expert systems are being constructed to tutor students at their own pace. The educational systems of today are embodied with more intelligence than were the traditional computer assisted instruction programs. The newer systems are attempting to be more tailored to each specific pupil's needs by accurately determining problem areas and concentrating in that realm. AI and expert systems in particular are viewed by some as ominous and threatening since machines will be performing tasks similar to those now accomplished by our most learned members of society.

There are several areas in which expert systems lack the ability to emulate their human counterparts. The computer lacks five senses with which to gather data and is forced to glean its data only from a keyboard. Expert systems also lack the emotional aspect of human decision making. Since they lack the human experiences of birth, growth, and death, their decisions are void of this element which human experts would naturally incorporate into a decision. Computers must methodically follow predetermined paths of reasoning while humans can experience flashes of

insight. Machines lack motivational drive and self-awareness. These deficiencies make it unlikely that AI applications, specifically expert systems, would ever replace humans as some now fear.

It has been stated that as technology increases there is a corresponding need for a similar rise in human contact (Naisbitt, 1983). The human touch can never be eliminated. This has never been more important to keep in mind than now with the use of expert systems on the rise.

An over dependence on computers may arise as the use of expert systems increases. Relying on a single system for advice in any one area could cause homogeneity of knowledge (Guterl, 1983). It is possible that fewer new insights would arise and fallacies increase from an overly accepted and used suite of programs.

C. REVIEW OF LITERATURE

Computers and communications technologies can be segmented into five generations. Early generations were characterized by their hardware components. Vacuum tubes, transistors, integrated circuits, and very large-scale integrated circuits describe the first four generations. The Japanese introduced the term "fifth generation computer" in one of their national projects. This fifth generation is characterized by software advances with an intense emphasis on artificial intelligence (Torrero, 1983).

According to Kahn, three main disciplines will make fifth generation computers possible. Microelectronics, artificial intelligence, and new hardware developments make up these disciplines. The microelectronics provide for creation of chips and wafers yielding fast and intricate devices. Techniques of AI produce systems that exhibit seemingly intelligent behavior. New hardware advances will make possible the creation of machines containing the new microelectronic elements on which these systems can be run (Kahn, 1983).

Determining exactly the approximate way of storing knowledge is an important task of the AI worker. A concise representation of knowledge is accomplished by a union of data structures and procedures to interpret the knowledge. Determining appropriate knowledge representation structures is part of the operations involved in knowledge engineering. Several kinds of knowledge need to be represented such as information about events, performance logics, and meta-knowledge. Meta-knowledge is using knowledge already obtained to produce still more knowledge (Barr, 1981).

According to Philip C. Treleaven:

> The nature of both data and processing tasks is changing. We are handling vast quantities of nonnumeric data, such as sentences, symbols, speech, graphics, and images, and processing is becoming less concerned with scientific calculation and more with artificial intelligence applications (Treleaven, 1984).

D. PROBLEM STATEMENT

This thesis concerns itself with developing an expert system for degree audits within the registrar's office at the University of Missouri-Rolla (UMR). It describes the design, development, and partial implementation of an expert system capable of conducting a check for fulfillment of graduation requirements known as a degree audit.

This project does not attempt to create a fine-tuned expert system but rather explores whether or not it is possible to begin an expert system for a portion of the degree auditing process at UMR. The undergraduate computer science degree requirements were selected as the sample set of graduation requirements on which efforts would be concentrated in designing the expert system. The task at hand is to encapsulate a complete set of rules describing the computer science degree auditing procedure.

II. DESCRIPTION OF EXPERT SYSTEMS

A. OVERVIEW

The aim of an expert system is to capture in computer code the skill of an expert in terms of both his formal training and the knowledge he has acquired through years of experience. Expert systems made their appearance in the mid-1970s (Hayes-Roth, 1983). They are known by other names. These include rule-based systems, expert consulting systems, and knowledge-based systems (Michie, 1982). Fields amenable to expert systems include medical diagnosis, analyzing scientific matter, and locating mineral deposits. General Electric has an expert system that points out problems that can occur simultaneously in a locomotive and suggests repair operations. This system is a computer-aided trouble-shooting system, Cats-1 (Kaplan, 1984).

Expert systems are generally only constructed for those areas where trained professional consultants are needed. The term expert system emphasizes system performance yielding results of a noted authority and not methodology; however, expert systems typically use methodic heuristic rules or rules of thumb obtained from consultations with the domain expert.

A program determined to be an expert system must be able to expand its own knowledge by reasoning as it proceeds through the designated processes. These systems generally

acquire the original information through an interactive session with the user. Once this original information is obtained, the system uses its knowledge base coupled with procedures within the inference engine to progress in an intelligent manner.

An expert system should be capable of explaining and justifying the decisions it makes and the lines of reasoning it followed in reaching a particular conclusion. The explanation facility of an expert system is vital for establishing trust in its results or conclusions.

The necessity for development of these inhuman mechanical experts becomes readily apparent in areas where experts are becoming fewer in number or where only one or two key individuals possess expertise vital to the successful operation of the institution. Experts willing to pass their expertise on to future generations free themselves from having to deal with matters they often consider trivial and furnish themselves the opportunity to concentrate on more challenging aspects of their expertise domain.

Expert systems appear to promise a more rapid advancement of knowledge. With the knowledge of today's experts captured in a computer program, tomorrow's experts are afforded the opportunity to climb on the shoulders of their predecessors and attain even greater heights by the added boost of an expert system.

Expert systems are created to accomplish a variety of types of activities. There are systems to interpret incoming data, predict certain outcomes, diagnose, plan, and design. Other expert systems assist in debugging, monitoring, control, repair, and instruction (Hayes-Roth, 1983).

Elaine Rich states:

> The goal of computer scientists working in the field of artificial intelligence is to produce programs that imitate human performance in a wide variety of "intelligent" tasks. For some expert systems, that goal is close to being obtained; for others, although we do not yet know how to construct programs that perform well on their own we can begin to build programs that significantly assist people in their performance of a task. In fact, long before we developed a "completely intelligent" system, we may expect to see responsibilities for problem-solving tasks gradually shift from the person to the machine (Rich, 1984).

B. STANDARD COMPONENTS OF AN EXPERT SYSTEM

An expert system is composed of a knowledge or rule base holding a set of rules often in the form IF <SITUATION> ===> THEN <ACTION>, an inference engine or control structure which manipulates the rules forming inferences, and input data or information pertaining to the situation at hand. The input data is often called a set of facts or a working data set. An interface program also exists which allows the user to understand program output and which translates user input to information understandable by the expert system.

The inference engine can be thought of as a type of theorem prover as it manipulates rules progressing to a viable conclusion. The knowledge base is domain specific. It contains rules directly applicable to the problem under consideration. Typically the inference engine is kept separate from the knowledge base. Often an inference engine void of domain-specific information can be used as the rule manipulator for several systems. This separation of inference engine and knowledge base also permits the knowledge base to grow and be modified.

Fully developed expert systems contain an explanation facility which allows a user to ask the system how it arrived at a specific conclusion or to display its line of reasoning. Confidence in using the expert system grows provided the user has the opportunity to ask the system for a display of how it arrived its results. This explanation could lead the user to a correct conclusion and reasoning pattern which might have been overlooked. An explanation facility also aids in debugging during system development.

The blackboard is a working area. In some systems it is called working memory. It contains the system's tentative solution and may be inspected during an audit session for needed information.

An ideal expert system contains a user interface which provides for communication between the system and user. It interprets questions posed by the user, his commands, and

any information he might volunteer. The user interface also formats information the system creates such as answers to questions, explanations of lines of reasoning or conclusions, and requests to the user for additional information.

C. SURVEY OF EXISTING EXPERT SYSTEMS

The earliest acknowledged expert system is Dendral originating in 1965 (Michie, 1979). It performs tasks of interpreting data and forming a hypothesis for identifying molecular structures from mass spectral and nuclear magnetic response data.

Casnet is an expert system which diagnoses, interprets, and provides therapy of glaucoma. It is highly efficient and used heavily (Hayes-Roth, 1983). Casnet utilizes reasoning under uncertainty.

Crib locates hardware and software faults in computer systems. The user gives the expert system an English-like description of the symptoms. Crib then performs a match between the observations and a database of known facts. It produces a report indicating whether a unit needs to be repaired or replaced.

Guidon is an instructional expert system dealing with the selection of antimicrobial therapy for patients with bacterial infections. It presents a case to the student to solve and analyzes his responses and queries during the

solution process. Guidon matches the similarity between the solution process it utilized and that of the student's.

Expert system Headmed specifies drug treatment for a wide range of psychiatric disorders. It was designed to be used as a tutorial and a consulting aid.

The expert system Macsyma solves math problems in such areas as algebra and integration. It has no explanation capability and cannot reason with uncertainty. No search is involved. Pattern recognition allows the solution method to be selected. Both Dendral and Macsyma are reputed for surpassing most human experts in their respective areas.

Mycin diagnoses and suggests therapy for infectious blood diseases. Its major emphasis is in explanation. It is used in medical teaching but not for clinical work. Mycin initiated the development of Emycin which is an expert system giving assistance in building other expert systems. It uses backward chaining from the hypothesis.

Prospector interprets data from soil and geological deposits of minerals. It has an explanation capability and can reason with uncertainty. This system has already demonstrated itself to be useful. Puff, a system working with the detection of pulmonary function disorders, and R1 which configures DEC VAX systems were expert systems developed in more recent years (Hayes-Roth, 1983).

VM diagnoses and suggests therapy for postsurgical patients in an intensive care unit. It interprets data from

the monitoring system of a hospital. VM provides breathing assistance based upon the patient's history and measurements obtained from the monitors (Waterman, 1986).

The preceding discussion of existing expert systems is in no way inclusive of all existing systems. Rather it provides an overview of what knowledge engineers have been producing. Table I shows some of these systems.

TABLE I

SELECTED EXISTING EXPERT SYSTEMS

Name	Description	Researchers
Dendral	Identifies organic compounds	E.A. Feigenbaum
Casnet	Glaucoma management	S. Weiss C. Kulikowski
Crib	Finds hardware and software faults	T.R. Addis
Emycin	Develops rule-based expert system	W. Van Melle
Guidon	CAI in diagnosing complex problems in medicine and science	W.J. Clancey
Headmed	Psychopharmacology advisor	J.F. Heiser
Macsyma	Algebraic manipulation system	M.R. Genesereth
Mycin	Diagnoses infectious diseases	E. Shortliffe
Prospector	Evaluates mineral sites	P. Hart R. Duda
Vm	Diagnoses and suggests critical care for mechanical breathing	L.M. Fagan

III. KNOWLEDGE ENGINEERING

The process of constructing an expert system has been termed knowledge engineering. To build a system which emulates an expert, several conditions must exist. There needs to be at least one person noted for the task in which his expert knowledge is being sought. The expertise being codified must be special knowledge, experience, and judgment. The individual knowledgeable in the domain of interest must have the desire, the communication skills, experience, and judgments necessary to convey his expertise to the knowledge engineer (Gevarter, 1983). This last requirement can present a problem in that experts so familiar with the task being discussed often lack the patience and tolerance to effectively communicate with a knowledge engineer who may be unfamiliar with the topic being discussed.

The area in which expert knowledge is being obtained is termed the domain of interest. The expert giving his expertise is called the domain expert. Building an expert system requires from seven months for simple systems with existing tools to fifteen years for complex systems requiring much work in research and development (Hayes-Roth, 1984).

Often an expert's knowledge is illusive and lacks well-defined boundaries. The expert sometimes exercises his

skill using rules or heuristics of which even he is unaware. If a system is to be developed which is capable of exercising the degree of skill embodied in the expert, this expertise, both formal and unconscious in nature, has to be obtained, suitably represented, and manipulated effectively.

The task just described of obtaining an individual's expertise is known as knowledge abstraction. Determining the most appropriate way of representing or storing the knowledge is known as knowledge representation. Knowledge engineer is the title given the individual who works with the domain expert and then converts the expertise into rules comprising a workable expert system.

As the demand for creation of expert systems increases steadily, the need for an increased number of skilled knowledge engineers becomes more apparent. It requires a skilled engineer indeed who can converse with a domain expert and ask the appropriate questions to glean knowledge required for the knowledge base.

Once the knowledge engineer obtains facts and rules from the domain expert his next step is to determine an appropriate vocabulary for use within the system. He must select the right data structures for the information and create programs permitting new knowledge to be obtained and utilized as the program executes. A prime objective in the knowledge representation phase of system construction is to separate the data structures and vocabulary from program

logic and language. The separation increases understanding of the system and assists in extension and maintenance.

Edward A. Feigenbaum states his belief in the importance of knowledge engineering when he says:

> In the middle 1960s, after the first expert systems such as Dendral, were built AI underwent a shift to a knowledge-based paradigm. The principle is that, although one cannot do without inference, what method one uses is not all that important relative to the knowledge the program has. Thus, the important questions are related to knowledge--how to represent it in a machine, and how to acquire it from nature or people (Feigenbaum, 1983).

The knowledge engineer leads the process of building an expert system through five general states of identification, conceptualization, formalization, implementation, and testing (Hayes-Roth, 1983). In the identification stage both knowledge engineer and expert specify the problem and scope for system development. They decide whether extra experts will need to be consulted.

Conceptualization sees the knowledge engineer and domain expert together determining key concepts, information-flow, and relations necessary to solve the problem. They identify subtasks and problem solving strategies.

In the formalization stage, the knowledge engineer attempts to find a suitable match between the needed elements as determined in the conceptualization stage and existing knowledge engineering tools. At the conclusion of

this stage, a set of specifications exists describing how the problem can be represented using the selected knowledge engineering tool.

The knowledge engineer in the implementation stage organizes the gleaned knowledge so that rules and a control structure exist comprising a prototype program. This program should be in a form allowing execution and testing. During the testing stage the system's performance is evaluated. It is changed as directed by the domain experts.

Knowledge engineering involves integrating facts, beliefs, heuristics, and judgments acquired by an expert after he has consumed much formal learning and extensive experience. The knowledge engineer attempts to do away with "blind alleys" and redundant operations. He must make certain the knowledge is correct as he incorporates it into the knowledge base. He often must integrate more than one knowledge source which can cause intervening problems. Difficulties can also be created by the dynamic learning process as objects are arranged within the system.

According to William B. Gevarter of the National Aeronautics and Space Administration,

> ...researchers have found that amassing a large amount of knowledge, rather than sophisticated reasoning techniques, is responsible for most of the power of the system (Gevarter, 1983).

IV. SYSTEM CONSTRUCTION AND IMPLEMENTATION

A. SYSTEM TO BE STUDIED

Since the purpose of an expert system is to create a system which emulates a human expert engaged in his area of expertise, the degree audit expert at UMR, Mr. Lauren Peterson, was contacted and observed as he conducted an actual degree audit session. His auditing process was analyzed and imitated during the design phase of the expert system.

An expert in UMR graduation requirements has to determine those individuals meeting all graduation requirements. The procedure he utilizes is called a degree audit.

Currently at UMR the degree audit expert makes a final graduation check for approximately eight hundred to nine hundred engineering and science majors each year. This graduation check is made at any time the student requests and possibly occurs several semesters before graduation. The student is generally present during the audit session.

A file is maintained of those students who have had an audit. As a degree audit request arrives from a student, the first step is to see whether or not a previous audit has been completed. Two cards are kept in a file for each student who has had a check made. One card is an application for diploma and the other is a card containing

student number, name, degree, department, and expected graduation date. It also contains total credit hours, current semester hours, minimum required hours, total grade points, and overall grade point average. In addition to these areas there is an area for required courses, social science electives, and humanities electives. There are also free areas where the auditor lists extra courses not counting toward the degree, courses left to take, courses in which the student is currently enrolled, those in which he is pre-enrolled if any exist, and free electives.

As requirements begin to be checked, the expert utilizes a check sheet for the specific department in which the student is majoring. Currently twenty different check sheets are kept for bachelor of science degrees in engineering and science. These check sheets are taken from the undergraduate bulletin and arranged in an order appropriate to the expert conducting the audit.

The proposed degree audit system attempts to capture the knowledge and procedures used by this expert. Such a system would be an immense asset should the individual presently conducting degree audits become unavailable prior to a graduation ceremony. An automated checking system would greatly decrease monotony the expert may experience performing several hundred audits a year. An automated audit system helps reduce any human error that might exist in addition to eliminating the file holding both cards from

the audit process.

B. BASIC DESIGN

After observing the degree audit expert at work and discussing all aspects of the process, it was time for the knowledge engineering stage to begin. A suitable method of representing all graduation requirements and all courses taken had to be established. All courses a student has successfully completed, hereafter known as the transcript, is represented as a list of courses. Each course in turn is represented as a list of three elements. The first element is the department name. It is followed by the course number. The last element in the representation of each course is the number of semester credits a student receives upon successful completion of the course. If, for example, a student has completed only MA&ST 8 and CMPSC 83 his transcript would be represented as ((MA&ST 8 5)(CMPSC 83 3)). Further knowledge representation details are discussed in subsequent sections as needed.

The entire set of graduation requirements were inspected and ultimately divided into five classes based upon the nature of each requirement. The following classes emerged as a result of this process: pre-selected, logic, rule, logic rule, and free elective. A sixth class emerged to handle courses not counting toward the degree.

The pre-selected classification consists of exact,

specific courses which a student must take for graduation to occur. Pre-selected courses were appropriately named to facilitate the update process in the event of eminent bulletin changes such as changes in course number, credit hours amount, etc. Each requirement within the pre-selected classification set may consist of either a single course as in the block structured language requirement ((CMPSC 163 3)) or a group of courses as in the math requirement ((MA&ST 8 5)(MA&ST 21 5)(MA&ST 22 4)). The representation of a pre-selected requirement with multiple courses such as the math requirement indicates all courses given in the list are required.

Logic is the second classification. This type of requirement involves the logical operators "and" and "or". Here the student has a choice among a few specific courses. This choice then is the separating factor between the pre-selected classification and the logic classification.

A third classification is rules. Even though the pre-selected and logic requirements could be thought of as rules in the expert system, the term "rule" is being used throughout as a requirement type classification. A rule consists of a requirement which can be encoded in no way other than a verbal type description much as would be found in the UMR undergraduate bulletin. This is a requirement giving the student freedom in specific course selection but with some constraints attached. A rule typically involves a

certain number of hours be taken from a specific department or from a given set of courses. There may also be level constraints in a requirement of this type such as course number be over 300 or courses be from different areas or departments.

Another classification is the requirement which involves the logical operators "and" and "or" as in the logic classification but which also involves the rules classification. Thus, this type of requirement is termed the logic rule classification.

The free elective classification emerged where the student has total freedom in course selection. No constraint exists on the course a student takes except that he must take enough courses to reach the minimum number of total course hours necessary for graduation.

Finally, the non-countable classification was needed to complete the possible course types that could exist on a transcript. This handles the courses taken which do not count toward meeting the specific degree requirements.

C. DETAILED DESIGN

1. Inference engine structure. Each requirement classification has its own inference engine. The degree audit expert system (DAES) conducts its audit procedure by sequentially accessing the inference engines belonging to each of the six requirement classes. These are specifically

the functions F_R_NCT (finds and removes non-countable courses), X_PS (executes pre-selected requirements), X_L (executes the logic class of requirements), X_LOGIC_RULES (executes logic rules), X_RL (executes the rule requirements), and X_FE (executes the free elective check). The inference engine for each rule class is responsible for determining the actions needed as it receives its respective individual requirements. The inference engines are always accessed in the order given with the most restrictive requirements checked first progressing to a check of the least restrictive ones.

Each inference engine receives a list of lists containing all requirements belonging to its classification type. Each requirement list is sent sequentially to its inference engine which notes new facts, accumulates hours, deletes courses from the transcript, etc. as needed.

2. Non-countable courses. As the audit procedure begins, the first action that occurs is the location of all courses the student has taken but which do not count toward any degree requirement.

The function which finds and removes non-countable courses is called F_R_NCT. Input to this function is a list of courses which do not count toward the particular degree requirements being audited known as non-countable (NON_COUNTABLE)=((MA&ST 1 1) (MA&ST 2 5) (MA&ST 4 3) (MA&ST 6 2) (MILSC 10 1) (MILSC 20 1) (MILSC 30 1)(MILSC 40 1)).

These courses were determined during the knowledge engineering phase of system construction.

Non-countable courses found on the transcript, if any exist, are stored in the form of a list known as the non-countable list (N_CL). They are in turn removed by the function which removes a course from the transcript (RM_C) and recorded on the message board as not counting. The recording process is covered later. The message board is known as the blackboard (BLK_BD). It serves as DAES's "scratch pad" recording bits of information throughout the audit. BLK_BD is a list containing sublists. Each sublist represents some type of knowledge the system has acquired or learned as the audit session progresses. Functions described later inspect the BLK_BD to retrieve requirements not filled, why they were not satisfied, and courses not counting.

F_R_NCT invokes the function (SUM_HRS) which finds the sum of the hours of courses (those still on the transcript) which count toward the degree and saves the total number of countable hours (TCNTHRS) for later use in determining how many hours are needed of free electives.

3. Pre-selected requirements. Once the non-countable courses are removed from the transcript, the pre-selected course requirements are checked in much the same manner as the non-countable courses were; the transcript is searched for the presence of each pre-selected course. If it is

found on the transcript, it is removed and recorded as filling that pre-selected requirement. Each pre-selected requirement is known by a descriptive identifier name such as NUM_ANALYSIS representing numerical analysis. A list of the pre-selected course representations is given in Table II.

If a pre-selected course has not been taken, it is recorded that this course is missing. There can be more than one course required for a given pre-selected requirement as in the math requirement or the English requirement. In this situation, the transcript searching process is repeated for the additional required courses. If some but not all courses of a given pre-selected requirement have been completed by the student, the completed courses are removed from the transcript and recorded as filling that requirement. The ones missing are recorded on BLK_BD as missing.

The recording process, necessary if the system is to be capable of explaining its actions, differs in the process employed to record the non-countable courses and the one used to capture the courses filling a requirement. In the non-countable case, the list of non-countable courses may immediately be recorded in its entirety. However, this is not necessarily so as the requirement descriptions become more complex. Therefore, the recording process for courses used in fulfilling the graduation requirements is

TABLE II

PRE-SELECTED REQUIREMENTS

IDENTIFIER	REPRESENTATION
CSOR	((CMPSC 1 1))
SCI_PROG	((CMPSC 73 2))
JCL	((CMPSC 74 3))
MCHLANG	((CMPSC 83 3))
BLK_STR_LANG	((CMPSC 163 3))
COBOL	((CMPSC 168 3))
ASSMBLR	((CMPSC 183 3))
NUM_ANALYSIS	((CMPSC 218 3))
DATA_STRUC	((CMPSC 253 3))
OPER_RES	((CMPSC 260 3))
EE	((ELENG 61 3) (ELENG 211 3))
MATH	((MA&ST 8 5) (MA&ST 21 5) (MA&ST 22 4))
ENGL	((ENGLSH 1 3) (ENGLSH 60 3))

accomplished by inspecting the transcript before and after a requirement has been processed and capturing the courses forming the difference between the two.

4. Logic requirements. After checking all pre-selected requirements the logic requirements are checked. These are the requirements involving specific courses as in the pre-selected requirements but where the student has some freedom in choosing the course or courses he/she takes. The identifier names are prefixed by the letter L to distinguish the requirements as logic requirements.

In the pre-selected requirements, multiple course descriptions in a requirement list implies an "and" connective. Separating the pre-selected requirements possessing an understood "and" from the logic requirements keeps the user from repeatedly typing the "and" connective while freeing the logic evaluator from uselessly searching for the "and" connective multiple times. In the logic requirements, the connectives "and" and "or" are explicitly stated in the list itself.

Each logical operator is preceded and followed by its two operands which may be a single course or a list of courses with a logical operator. All three of these elements form the requirement representation list.

The logic evaluator function LOG_EVAL can effectively process rules with multiple "and" and "or" connectives as

well as the simple rule with only the word "or" present. The algebra requirement ((MA&ST 203 3) OR (MA&ST 208 3)) and the lab science requirement (((CHEM 1 4) AND (CHEM 2 1)) OR ((LIFSCI 1 3) AND (LIFSCI 2 2))) are both examples of rule types handled by the logic evaluator. Table III contains the various types of "and/or" list combinations possible with LOG_EVAL.

The logic evaluator searches recursively, if need be, until it locates one of the keywords "and" or "or" preceded by a course list. Then the transcript is searched for the courses on each side of the keyword. If the course is on the transcript, the searching function returns that course. If it has not been taken, NIL is returned so that the appropriate computation on "or" or "and" may be computed to determine if the entire requirement has been satisfied. If a requirement is not entirely satisfied, those courses found are removed from the transcript and a message is recorded stating that requirement was not met. The courses used to fill or partially fill each requirement are recorded for future reference.

5. Logic rules. Once non-countable courses are removed, pre-selected courses checked, and logic requirements dealing with specific courses have been evaluated, requirements involving both rules and the logic connective words "and" and "or" are checked. An identifier for a logic rule is prefixed LR. Only one graduation

TABLE III

LOGIC REQUIREMENTS

IDENTIFIER	REPRESENTATION
L_STAT	((MA&ST 215 3) OR (MA&ST 343 3))
L_ALG	((MA&ST 203 3) OR (MA&ST 208 3))
L_LABSCI	(((CHEM 1 4) AND (CHEM 2 1)) OR ((LIFSCI 1 3) AND (LIFSCI 2 2)))
L_PHYS	((((PHYSCS 21 4) AND (PHYSCS 22 1)) OR (PHYSCS 23 4)) AND (((PHYSCS 25 4) AND (PHYSCS 26 1)) OR (PHYSCS 24 4))))

requirement fits the logic rule classification. This is the literature/speech requirement (LR_LIT_SPEECH)=(((HRS 3 FROM THE COURSES LIT) AND (HRS 3 FROM THE DEPT SP&MS)) OR (HRS 6 FROM THE COURSES LIT))).

To check a logic rule, a function called the logic recognizer (L_R) scans each rule locating keywords and searching the transcript in the manner described by the following discussion on rules. However, unlike the rule classification, DAES cannot automatically record a message such as there are not enough hours present since that may not be the case if the student has selected a requirement alternative. L_R must return either NIL or the courses which fit a requirement so that the logic rule evaluator can determine whether or not the requirement has been met. The task is thus more involved since courses partially filling a logic rule requirement may or may not need to be deleted from the transcript.

6. Rules. Next the simple (non-logic) rules are executed. These rules have been ordered such that the narrowest, more confining, requirements are checked before the broader requirements. Their identifiers are each prefixed by R for rule. Every rule of this type is intended to be "user-friendly" in nature and is represented as a list of keywords and numerals. Table IV contains a list of the recognizable rule keywords, their respective legal values, and an example of usage. All rule descriptions currently

TABLE IV

RULE KEYWORDS

KEYWORD	MEANING	LEGAL VALUES	EXAMPLE
LEVEL	course level number	positive integer	LEVEL 300
DEPT	department	any department name at UMR	DEPT PHIL
CRS	required number of courses	positive integer	CRS 2
WHERE_HRS	specified number of hours required from a specific level	positive integer	WHERE_HRS 6
HRS	number of hours	positive integer	HRS 12
DEPTS	list of department names	identifier containing a list of UMR department names	DEPTS SOC_SCI
COURSES	list of specific courses	identifier containing a list of course representations	COURSES CONST
AREAS	number of areas	positive integer	AREAS 2
NONCON-SUMABLE	check only for presence on transcript; leave located courses on transcript	no restriction	NONCONSUMABLE REQ

TABLE V

RULE REQUIREMENTS

IDENTIFIER	REPRESENTATION
R_CONST	(CRS 1 FROM THE COURSES CONST A NONCONSUMABLE REQUIREMENT)
R_PHIL	(HRS 3 FROM THE DEPT PHIL)
R_MATH	(HRS 3 OF LEVEL 22 OR GREATER FROM THE DEPT MA&ST)
R_SOCSCI	(HRS 9 FROM AREAS 2 OF THE DEPTS SOC_SCI)
R_SCIENG	(HRS 9 FROM THE DEPTS BS_NOT_CS)
R_CSC	(HRS 12 WHERE HRS 6 ARE OF LEVEL 300 OR GREATER FROM THE DEPT CMPSC)

used in this system are given in Table V.

Each rule may contain any of the needed keywords having meaning to the rule-recognizer function (R_R) which scans the rule to pick out these keywords. The only constraint on rule construction is that the value belonging to a keyword immediately follows its keyword. For example, the philosophy rule has the keywords HRS (hours) and DEPT (department) and is structured as (HRS 3 FROM THE DEPT PHIL). More words deemed necessary for readability could be inserted in the rule provided nothing comes between the keyword and its associated value.

The first rule executed is the one dealing with the constitution requirement (R_CONST)= (CRS 1 FROM THE COURSES CONST A NONCONSUMABLE REQUIREMENT). This is a non-consumable requirement so that a course on the transcript satisfying this requirement will not be deleted from the transcript at this time but will rather be left on the transcript so that it may fill another requirement for graduation. A non-consumable requirement necessitates only a check for presence on the transcript and by nature does not "consume" the course unlike rules without the non-consumable keyword.

All rules are evaluated by the rule recognizer. The constitution requirement rule has the keywords CRS, COURSES, and NONCONSUMABLE. The rule-recognizer locates these words. Then it searches the transcript to see if the appropriate

number of courses is on the transcript from the list of courses satisfying the constitution requirement which is given as a list of courses named CONST. If the appropriate number does not exist, the rule recognizer records this fact. If at least the needed number of courses is located on the transcript, the courses are not deleted from the transcript since this is a non-consumable requirement; however, the course or courses located on the transcript which fill the requirement are recorded as satisfying R_CONST for the explanation facility. If no courses in the CONST set of courses are located on the transcript, this fact is recorded.

The philosophy requirement (R_PHIL)=(HRS 3 FROM THE DEPT PHIL) is fired next. This is the least complex rule type. It requires the rule-recognizer to deal with only the DEPT and HRS keywords. All courses on the transcript from the philosophy department are located by the find-department function (F_D). If no courses exist on the transcript from the needed department, this is recorded. If courses are located, their semester hours are accumulated to determine if enough hours have been taken from the needed department. If so, only the needed number of semester hours are deleted from the transcript and recorded as satisfying the philosophy requirement. If not enough hours have been taken, an appropriate message is appended to BLK_BD. The courses partially filling the requirement are deleted from

the transcript and recorded as partially filling the requirement.

The math requirement rule (R_MATH)=(HRS 3 OF LEVEL 23 OR GREATER FROM THE DEPT MA&ST) gets attention next. This is a more logically complex rule to evaluate than was the philosophy rule in that the rule-recognizer has to deal with the LEVEL keyword in addition to the DEPT and HRS ones.

First the mathematics courses on the transcript are located by the department-finding function (F_D) which located the philosophy department courses. As described above, if no math courses are found, the system remembers this fact and continues with the next rule. If courses are located, a check for additional keywords is made. Since there are more keyword constraints other than simply DEPT and HRS, the hours amount is not yet tested.

The D_L is searched for those courses with the needed course number level. They are extracted and put on the found level list (F_LL). If no additional constraint is present on the number of hours needed from this specific level, the hours on F_LL are accumulated and checked to see if there are at least as many as the rule specifies. If so, the needed number of hours are deleted from the transcript and recorded as filling the R_MATH requirement. If not, the located courses on F_LL are deleted from the transcript pool and recorded as partially filling the R_MATH rule with a message stating not enough hours were found placed on the

blackboard.

The rule for the social science requirement (R_SOCSCI)=(HRS 9 FROM AREAS 2 OF THE DEPTS SOCSCI) requires the DEPTS, AREAS, and HRS keywords. The rule-recognizer locates these keywords. It invokes a function similar to the function used in R_PHIL and R_MATH to locate courses on the transcript from a single department except this function (F_SD) returns all courses found from all of the departments listed in the set of departments given in R_SOCSCI. The set of departments is called SOC_SCI and is represented as (ECONOM HIST POLYSCI PSYCH SOCIOL). This function returns a list of courses known as the set list (S_L) from the various needed departments set. All courses on S_L from each present department are enclosed within a list. Thus it's quite probable the function returns a list of lists.

If any courses were located, the areas constraint is dealt with. If the areas constraint is absent, the parentheses separating the courses of different departments in S_L are removed by the function which removes extra parentheses (RM_P). If the areas constraint does exist, a function exists which returns and saves for possible future reference the names of the various departments found (RETURN_UNIQ_DEPTS). The length of this unique department list is the count of the different areas in which the student has currently had courses. A function is invoked which captures one course from each unique department. This

list of courses from the unique areas is known as the area list (A_L).

Finally, the hours are accumulated on the S_L and tested against the required number of hours given in the rule. The same process occurs here for the hours check as previously described for other rules. The needed hours are removed from the transcript. Note that courses from the different areas present will be removed before the other courses since they physically appear first on S_L from which courses are removed, one by one from left to right.

The rule for the science and engineering requirement (R_SCIENG)=(HRS 9 FROM THE DEPTS BS_NOT_CS) is similar to the R_SOCSCI except the areas constraint is absent. The process for locating the various department courses and checking the hours is the same. Notice that a psychology course, for example, would accurately be selected by DAES to help fulfill the social science rule instead of the science and engineering rule, which it could help fulfill. This is because the social science rule will always be checked for fulfillment prior to the science and engineering one.

Perhaps the most complex rule of all is the rule for computer science courses because of the double constraint on hours. (R_CSC)=(HRS 12 WHERE HRS 6 ARE OF LEVEL 300 OR GREATER FROM THE DEPT CMPSC). All courses still on the transcript from the computer science department are located. The hours are accumulated next. Now courses only of the

appropriate level are extracted from the D_L and placed on the F_LL by the find level function (F_LEV). The hours on F_LL are accumulated. Since the WHERE_HRS keyword is present in this rule, this value is tested against the hours count on F_LL. The same sequence of actions occur in testing the where_hrs constraint as does in the hours test described previously. Care must be taken to remove only the needed amount of hours (WHERE_HRS) from the transcript. If not enough where_hrs exist, this deficiency is recorded. Courses deleted from the transcript for the where_hrs constraint are removed from the D_L. Finally, if any courses remain on D_L, they are removed as dictated by the hours constraint.

7. Free-electives. After all non-countable courses are removed and the pre-selected, logic, logic rule, and rule requirements are checked, it is time to determine if enough total hours have been taken. This requirement is called the FREE_ELECTIVE_REQUIREMENT and is performed by a function called execute free elective (X_FE).

At the completion of executing all rules, the credit hours remaining for courses still on the transcript are accumulated and stored as hours left (HRSLEFT). The X_FE function determines the total amount of hours that may be removed (THRSRM) from the transcript. If the total number of countable hours (TCNTHRS) is less than 130, the blackboard receives a message that there are not enough free

elective hours and gives the number of hours the student still needs to take. Then it removes all courses on the transcript and records those courses as helping to fill the free elective requirement. If there are 130 or more total countable hours, the number of hours needed to satisfy the 130 hour requirement are removed and recorded as filling the FREE_ELECTIVE_REQUIREMENT. Any remaining courses are recorded on BLK_BD as being extra courses.

D. DAES USER INTERFACE

DAES's driver function (DEGREE_AUDIT) performs the degree audit and produces a report giving any messages from the blackboard stating deficiencies in requirements. If, at a later time, the user wishes to see the entire deficiency report again, it may be obtained by invoking the function which shows what remains unsatisfied (WHATS_NOT_FILLED). Table VI contains the valid user queries, keywords, and accompanying arguments.

A list known as HISTORY, created by the CREATE_HISTORY function, may be inspected at any time. HISTORY shows how DAES used each course on the transcript. It is represented as a long list whose elements consist of lists. Each of these sublists are comprised of the identifier, such as NON_COUNTABLE, OPER_RES, L_PHYS, LR_LIT_SPEECH, R_MATH, or FREE_ELECTIVE_REQUIREMENT, followed by a list of the courses used to help satisfy the requirement represented by each

TABLE VI

USER QUERIES

KEYWORD	ARGUMENT(S)	ACTION	EXAMPLE
HISTORY	none	shows which course filled each requirement	HISTORY
WHAT_FILLED	requirement name	shows courses which filled a specific requirement	(WHAT_FILLED R_MATH)
WHATS_NOT_FILLED	none	shows all requirements not satisfied	(WHATS_NOT_FILLED)
WHY	requirement name	shows why a requirement was or was not satisfied	(WHY R_SOCSCI)
BLK_BD	none	displays all messages and recorded information placed on the blackboard	BLK_BD
SHOW_NAMES	none	shows system names of requirements	(SHOW_NAMES)
BLD_TRANS	none	creates a new transcript	(BLD_TRANS)
ADD_TO_TRANS	none	adds any number of new courses to transcript	(ADD_TO_TRANS)
DELETE_FROM_TRANS	none	deletes any number of courses from the transcript	(DELETE_FROM_TRANS)

identifier. This gives the history of the entire degree audit session. If the user is interested only in what filled one specific requirement, he/she may obtain this information from DAES by invoking the function WHAT_FILLED and giving it the identifier name in question such as (WHAT_FILLED R_CSC).

In the event a user wishes more information concerning why a requirement was not met during the audit session, he/she may desire to query DAES as to why a requirement was not filled. This may be accomplished using the WHY function in conjunction with the identifier name in question such as (WHY R_SCIENG). This produces the rule associated with the science and engineering requirement prefixed with the words BECAUSE R_SCIENG REQUIRES. This can be especially helpful when used together with the WHAT_FILLED function to determine exactly why a rule failed in the event BLK_BD does not explain to user satisfaction.

Various other user interface features include BLK_BD, SHOW_NAMES, BLD_TRANS, ADD_TO_TRANS, and DELETE_FROM_TRANS. BLK_BD displays the blackboard DAES used during the audit session. The function SHOW_NAMES, invoked by (SHOW_NAMES) displays the requirement names as known by DAES. BLD_TRANS creates a new transcript. Assuming the transcript is to contain the course representations of MA&ST 8 and CMPSC 163, it could be built as follows:

(BLD_TRANS) <enter>

(MA&ST 8 5)(CMPSC 163 3) <enter>

ADD_TO_TRANS inserts any number of courses to an existing transcript. If the courses CMPSC 253 and HIST 112 are to be added to a transcript, it could be accomplished by

(ADD_TO_TRANS) <enter>

(CMPSC 253 3)(HIST 112 3) <enter>

DELETE_FROM_TRANS deletes any number of courses from an existing transcript. If the courses MA&ST 8 and HIST 112 are to be deleted from the transcript, the delete function would provide the necessary actions by:

(DELETE_FROM_TRANS) <enter>

(MA&ST 8 5)(HIST 112 3) <enter>

E. DEVELOPMENTAL TOOLS

The degree audit expert system was designed on an IBM PC/XT using the IQLISP (Integral Quality, 1983) interpreter, version 1.6. The minimum memory amount of 256K bytes is necessary. The LISP language, designed for list processing, lends itself well to this particular application. With each course represented as a list and the entire transcript represented as a list of courses, the search-match process so familiar in the audit procedure was greatly facilitated by LISP's built-in list manipulation functions.

The basic structure of LISP is the function. A LISP program consists of one function invocation followed by another. LISP's functions evaluated serially,

conditionally, iteratively or recursively provide a wide range of control structures all of which were utilized during implementation of the system.

LISP has been lauded for its unparalleled ability in expressing recursive algorithms and in manipulating dynamic data structures (Tucker, 1986). The degree audit system heavily utilizes the capability of working with dynamic data structures especially in the transcript and blackboard representations. LISP's prefix notation was used during the knowledge representation phase of rule construction.

V. RESULTS AND CONCLUSIONS

A. EXECUTION WITH ACTUAL TRANSCRIPT

During the development and implementation phases of system construction, each function was tested as it was created or enhanced. Eventually, the time came to test the system with an actual UMR student's transcript. A computer science major's transcript was selected at random. Each course was represented in a form DAES understands (i.e. (dept course-number hours)) and entered forming the transcript list.

This particular student had participated in the coop program. A decision needed to be made as to the representation form of a coop course. It was decided that coop courses be represented with course number 0 and hours amount 0 so that these hours would not be counted toward the computer science rule requirement. Also, the complication of waived courses became a reality since this student had a physics lab waived. The system was not modified to deal with the waived course; however, this modification could be accomplished by allowing the word WAIVED to be placed within the course representation and by creating a list of each student's waived courses. The waived course list could be checked as requirements are found unfulfilled to determine if they have been waived.

The final transcript form along with the system's

degree audit report may be seen in Appendix A. The system produced a correct analysis of degree attainment progress with, of course, the exception of the waived physics lab. It reported that the physics requirement had not been met when it actually had with only the waived lab missing.

The student had taken several courses which will not count toward the computer science degree. The system correctly located them and handled them in the appropriate manner. It noted with 100% accuracy which pre-selected courses were present and which were missing. It correctly discovered not enough hours had been taken for the computer science and science/engineering rules and that not enough free elective hours were present.

B. TRANSCRIPT MODIFICATIONS

The actual transcript was modified in several ways to determine if the system would respond in the appropriate manner. The first transcript modification run involved the following changes:

1. Addition of the courses
   (CMPSC 1 1) ,(ENGMGT 130 3), (PSYCH 50 3),
   (CMPSC 268 3), (CMPSC 293 3), (PHYSCS 23 4),
   and (MA&ST 343 3)
2. Deletion of the courses
   (ELENG 211 3), (MA&ST 22 4), (ENGLSH 60 3),
   (CHEM 2 1), (CMPSC 423 4), (CMPSC 0 0),

(MA&ST 208 3), (PHYSCS 21 4), and (MA&ST 215 3)

This modified transcript (called TR1) run is given in Appendix B. The degree audit report produced by DAES, the HISTORY list showing how it used the courses, and the final blackboard is also displayed in Appendix B. Other modified transcript runs are given in Appendix C through Appendix F.

It located a newly missing pre-selected course, namely data structures. Each of the three pre-selected requirements containing more than one course now has a course missing. It correctly located those courses present and noted the missing ones.

DAES realized and noted the algebra and lab science requirements were now unfulfilled. It placed the new engineering management and psychology courses in the correct category and noted the hour deficiency in R_SCIENG. It also correctly calculated the number of overall hours still to be taken. A different sequence of courses now satisfy the physics requirement. DAES correctly handled this situation. It accurately found R_CSC now satisfied.

The actual transcript was modified a second time (known as TR2) in various other ways to test additional system features. Here are the changes made to the original transcript for the second test run:

1. Addition of the courses
   (PHYSCS 22 1),(MA&ST 203 3),(LIFSCI 2 2), and
   (LIFSCI 1 3)

2. Deletion of the courses

(MA&ST 4 3),(MA&ST 6 2), (MILSC 10 1),

(MILSC 30 1),(MILSC 20 1),(MILSC 40 1),

(CMPSC 0 0),(CMPSC 423 4), (CMPSC 361 3),

(MA&ST 208 3),(CHEM 1 4),(CHEM 2 1)

The absence of non-countable courses was correctly handled. It accurately located the missing pre-selected courses and found that there were not enough hours for the R_SCIENG requirement. It correctly noted that there were not enough hours of the appropriate level for R_CSC and computed the correct number of hours needed to fulfill the total semester hours requirement. Results are in Appendix C.

The third transcript modification (called TR3) run consisted of the following changes with respect to the original transcript:

1. Addition of the courses

(ENGLSH 110 3), (PHYSCS 23 4), (PHYSCS 24 4),

(ECONOM 111 3), (ECONOM 215 3), (CMPSC 264 4),

(CMPSC 268 3), (CMPSC 101 3)

2. Deletion of the courses

(PHIL 15 3),(HIST 112 3),(HIST 176 3),

(PHYSCS 21 4),(PHYSCS 25 4),(PHYSCS 26 1),

(MA&ST 204 3),(CMPSC 0 0),(CMPSC 423 4)

In this run, missing pre-selected courses were located. It correctly noted that no courses were found to fulfill

R_CONST. The system realized that no philosophy courses had been taken and no math courses were left on the transcript to fulfill R_MATH. Enough hours for R_SOCSCI were present on the transcript but not from two different areas. The system appropriately dealt with this situation. It correctly noted more hours were needed for R_SCIENG. Finally it accurately calculated the number of hours remaining to reach the 130 hour requirement.

Different courses were used to satisfy the physics requirement. The system correctly handled this situation. It found R_CSC now satisfied using the correct number of semester hours with course number of level 300 and filling the remaining hours needed with the correct number of hours from courses in the computer science department. Here, extra computer science courses were placed on the transcript above the hours amount needed for R_CSC. The system accurately used the extra ones as free electives. Appendix D gives system results with this transcript.

The fourth test run, given in Appendix E, involved modifying the third transcript form just described (called TR4). Here are the changes made to the third transcript version:

1. Addition of the courses
   (CMPSC 1 1),(CMPSC 260 3),(PSYCH 50 3),
   (POLYSCI 90 3),(PHIL 10 3),(MA&ST 209 3), and
   (MA&ST 115 3)

2. Deletion of the course (MTENG 1 1)

Execution with this transcript found all pre-selected requirements met. The system filled R_PHIL with a different philosophy course than previously used. A different course was located to fill R_CONST. A math course was correctly used to help satisfy R_SCIENG. A different math course was used to fulfill R_MATH and different areas were used to meet the areas criteria in R_SOCSCI. The system accurately realized that all graduation requirements had been met and printed the expected congratulatory message.

Finally, the fifth transcript (named TR5) run involved adding the courses (PSYCH 52 3) and (SOCIOL 81 3) to the fourth transcript version. The system correctly output that all requirements had been met and recorded a message on the blackboard stating that there were extra courses taken and listing those courses used as extra ones. Execution results from using this transcript are given in Appendix F.

C. CONCLUDING REMARKS

During system development it became evident that it was indeed possible to construct a complete set of rules describing the UMR computer science degree audit procedure. The system is general enough that it could be extended to other departments within UMR.

Degree audit expert systems could be created for other institutions based upon the underlying structure of this

one. The five requirement types which emerged during the knowledge engineering phase of system development are basic to the graduation requirement types of virtually any institution. Thus, with some probable modifications, an expert system could be created to effectively handle other degree audits. The entire set of functions comprising the expert system is given in Appendix G.

VI. FUTURE DIRECTIONS

In the event the expert system needs to be extended to other departments, requirements would need to be represented in the five given requirement classes. A knowledge engineering phase would need to be completed for the requirements of each department. Should the requirements of a department need rules other than the type the rule recognizer would be able to deal with effectively, the R_R function could be modified to accommodate new rule types with a minimal amount of effort. This would entail adding the new keywords to the list of keywords found in the F_KEYWDS function. It would also necessitate adding a section of code to the rule-recognizer function specifying the actions needed for the new keywords.

There are several possible uses for a degree auditing expert system. It could provide assistance to advisors during enrollment periods. Should the system be put online so that each student could freely access it, he/she would have available at any time a report of individual progress toward fulfilling the UMR degree requirements. The system could be made to deal with waived courses. It also could be made to contain the grades of a student for each course on the transcript.

It could automatically make application of diploma for those students passing the audit and create the final

graduation roster. The system could also be made interactive so that the user could enter various information. This might be advantageous to a student desiring to see the varying effects of taking different course sequences.

BIBLIOGRAPHY

Barr, Avron, Bennett, James, and Clancey, William, "Transfer of Expertise: A Theme for AI Research," Stanford Heuristic Programming Project, HPP-79-11, March 1979.

Barr, Avron and Feigenbaum, Edward A., The Handbook of Artificial Intelligence, Vol. 1, Los Altos, Cailfornia, William Kaufmann, Inc., 1981, pp. 3-11, 143-152.

Barstow, David R. "Maxims for Knowledge Engineering," Schlumberger-Doll Research Labortory, SDR AI Memo No. 10, May 1981.

Buchanan, Bruce G. and Duda, Richard O., "Principles of Rule-Based Expert Systems," Heuristic Programming Project, HPP-82-14, August 1982.

Cohen, Paul R. and Feigenbaum, Edward A., The Handbook of Artificial Intelligence, Vol. III, Los Altos, California, William Kaufmann, Inc., 1982, pp. 1-105.

Feigenbaum, E.A., "The Art of Artificial Intelligence: Themes and Case Studies of Knowledge Engineering," Stanford Heuristic Programming Project, HPP-77-25, August 1977.

Feigenbaum, Edward A., "Artificial Intelligence," IEEE Spectrum, Vol. 20, No. 11, November 1983, pp. 77-78.

Feigenbaum, Edward A., "Artificial Intelligence Research,

What is it? What has it achieved? Where is it going?," Stanford University Report, UMC-HSRC 8617-7.

Gevarter, William B., "Expert Systems: Limited but Powerful," IEEE Spectrum, Vol. 20, No. 8, August 1983, pp. 39-45.

Guterl, Fred, "Next Generation Impacts," IEEE Spectrum, Vol. 20, No. 11, November 1983, pp. 111-117.

Hartley, Roger T., "CRIB: Computer Fault-finding through Knowledge Engineering," Computer, Vol. 17, No. 3, March 1984, pp. 76-83.

Hayes-Roth, Frederick, "Codifying Human Knowledge for Machine Reading," IEEE Spectrum, Vol. 20, No. 11, November 1983, pp. 79-81.

Hayes-Roth, Frederick, Waterman, Donald A., Lenat, Douglas B., Building Expert Systems, Reading, Massachusetts, Addison-Wesley Publishing Company, Inc., 1983, pp. 7, 14, 16-19, 38-41.

Integral Quality, Trademark found in IQLISP Reference Manual, 1983.

Kahn, Robert E., "A New Generation in Computing," IEEE Spectrum, Vol. 20, No. 11, November 1983, pp. 36-41.

Kaplan, Gadi, "Industrial Electronics," IEEE Spectrum, Vol. 21, No. 1, January 1984, pp. 70-71.

Lerner, Eric J., "Technology and the Military: DOD's Darpa at 25," IEEE Spectrum, Vol. 20, No. 8, August 1983, pp. 70-73.

Michie, Donald, Introductory Readings in Expert Systems, New York, New York, Gordon and Breach Science Publishers, 1982, pp. 1-50.

Moto-Oka, Tohru, "Fifth-Generation Computer Systems: A Japanese Project," Computer, Vol. 17, No. 3, March 1984, pp. 6-13.

Moto-Oka, Tohru, "The Fifth Generation: A Quantum Jump in Friendliness," IEEE Spectrum, Vol. 20, No. 11, November 1983, pp. 46-47.

Naisbitt, John, "Computer Trends," IEEE Spectrum, Vol. 20, No. 11, November 1983, pp. 110-111.

Nii, H. Penny, "An Introduction to Knowledge Engineering, Blackboard Model, and AGE," Stanford Heuristic Programming Project, HPP-80-29, March 1980.

Rich, Elaine, Artificial Intelligence, New York, New York, McGraw-Hill Book Company, 1983, pp. 3-22, 32-35, 113.

Rich, Elaine, "The Gradual Expansion of Artificial Intelligence," Computer, Vol. 17, No. 5, May 1984, pp. 4-12.

Stevens, Lawrence, Artificial Intelligence: In Search for the Perfect Machine, Hasbrouck, New Jersey, Hayden Book Company, 1985, pp. 3-6.

Torrero, Edward A., "Tomorrow's Computers," IEEE Spectrum, Vol. 20, No. 11, November 1983, pp. 34-35.

Treleaven, Philip C., "Future Computers: Logic, Data Flow..., Control Flow?", Computer, Vol. 17, No. 3,

March 1984, pp. 47-55.

Tucker, Allen B., Programming Languages, New York, New York, McGraw-Hill Book Company, 1986, pp. 317-349.

Waterman, Donald A., A Guide to Expert Systems, Reading, Massachusetts, Addison Wesley Book Company, 1986, pp. 249-248.

Yau, Stephen S., "Japanese Computer Technology and Culture," Computer, Vol. 17, No. 3, March 1984, pp. 4-5.

Yazdani, Masoud, Narayanan, Ajit, Aritficial Intelligence: Human Effects, New York, New York, John Wiley and Sons, 1984, pp. 200-201.

Yeh, Raymond, "Software Engineering," IEEE Spectrum, Vol. 20, No. 11, November 1983, pp. 91-94.

VITA

Ruth Sue Dare was born on November 3, 1960 in Kirksville, Missouri. She graduated from Kirksville Senior High School in Kirksville, Missouri in 1979. She received a Bachelor of Science Degree in Computer Science and a Bachelor of Science in Education Degree in Secondary Mathematics from Northeast Missouri State University (NMSU), Kirksville, Missouri, where she graduated Summa Cum Laude and Valedictorian in May 1983. While attending NMSU she was initiated into 2 honor societies: Kappa Mu Epsilon and Alpha Phi Sigma.

She has been enrolled in the Graduate School of the University of Missouri-Rolla since May 1983 and held the position of Graduate Teaching Assistant in Computer Science from August 1983 through July 1984. She has held a Chancellor's Fellowship for the duration of her graduate work. She is currently at Northeast Missouri State University where she's been teaching since August 1984.

APPENDIX A

EXECUTION WITH ACTUAL TRANSCRIPT

ROLLA_TRANS

```
((MA&ST 4 3) (MA&ST 6 2) (CHEM 1 4) (CHEM 2 1) (ENGTC 1~
0 3) (HIST 176 3) (MA&ST 8 5) (MTENG 1 1) (CHEM 3 3) (C~
MPSC 73 2) (MA&ST 21 5) (MILSC 10 1) (PHYSCS 21 4) (ENG~
LSH 1 3) (HIST 112 3) (CMPSC 74 3) (ECONOM 110 3) (MA&S~
T 22 4) (MILSC 30 1) (PHYED 103 1) (PHYSCS 25 4) (PHYSC~
S 26 1) (CMPSC 83 3) (CMPSC 163 3) (ENGLSH 106 3) (MA&S~
T 204 3) (MILSC 20 1) (MILSC 40 1) (PHIL 15 3) (CMPSC 1~
83 3) (CMPSC 218 3) (CMPSC 253 3) (ELENG 61 3) (SP&MS 8~
5 3) (CMPSC 0 0) (ENGLSH 60 3) (MA&ST 208 3) (CMPSC 168
 3) (CMPSC 349 3) (CMPSC 361 3) (ELENG 211 3) (MA&ST 21~
5 3) (SP&MS 283 3) (CMPSC 0 0) (CMPSC 423 4))
```

DEGREE AUDIT REPORT

```
(MISSING THE COURSE (S) ((CMPSC 1 1)))
(MISSING THE COURSE (S) ((CMPSC 260 3)))
(REQUIREMENTS NOT MET FOR L_PHYS)
(NOT ENOUGH HOURS FOR R_SCIENG)
(NOT ENOUGH HOURS FOR R_CSC)
(NOT ENOUGH HOURS FOR FREE_ELECTIVE_REQUIREMENT SINCE 1~
9 MORE HOURS ARE NEEDED)
```

HISTORY

```
((NON_COUNTABLE ((MA&ST 4 3) (MA&ST 6 2) (MILSC 10 1) (
MILSC 20 1) (MILSC 30 1) (MILSC 40 1))) (CSOR NIL) (SCI~
_PROG ((CMPSC 73 2))) (JCL ((CMPSC 74 3))) (MCHLANG ((C~
MPSC 83 3))) (BLK_STR_LANG ((CMPSC 163 3))) (COBOL ((CM~
PSC 168 3))) (ASSMBLR ((CMPSC 183 3))) (NUM_ANALYSIS ((
CMPSC 218 3))) (DATA_STRUC ((CMPSC 253 3))) (OPER_RES N~
IL) (EE ((ELENG 61 3) (ELENG 211 3))) (MATH ((MA&ST 8 5
) (MA&ST 21 5) (MA&ST 22 4))) (ENGL ((ENGLSH 1 3) (ENGL~
SH 60 3))) (L_STAT ((MA&ST 215 3))) (L_ALG ((MA&ST 208
3))) (L_LABSCI ((CHEM 1 4) (CHEM 2 1))) (L_PHYS ((PHYSC~
S 21 4) (PHYSCS 25 4) (PHYSCS 26 1))) (LR_LIT_SPEECH ((
ENGLSH 106 3) (SP&MS 283 3))) (R_CONST ((HIST 112 3) (H~
IST 176 3))) (R_PHIL ((PHIL 15 3))) (R_MATH ((MA&ST 204
 3))) (R_SOCSCI ((HIST 176 3) (HIST 112 3) (ECONOM 110
3))) (R_SCIENG ((MTENG 1 1) (CHEM 3 3))) (R_CSC ((CMPSC
 0 0) (CMPSC 349 3) (CMPSC 361 3) (CMPSC 0 0) (CMPSC 42~
3 4))) (FREE_ELECTIVE_REQUIREMENT ((ENGTC 10 3) (PHYED
103 1) (SP&MS 85 3))))
```

BLK_BD

((THESE COURSES DIDN'T COUNT ((MA&ST 4 3) (MA&ST 6 2) (
MILSC 10 1) (MILSC 20 1) (MILSC 30 1) (MILSC 40 1))) (M~
ISSING THE COURSE (S) ((CMPSC 1 1))) (ALL REQUIREMENTS
MET FOR SCI_PROG) (ALL REQUIREMENTS MET FOR JCL) (ALL R~
EQUIREMENTS MET FOR MCHLANG) (ALL REQUIREMENTS MET FOR
BLK_STR_LANG) (ALL REQUIREMENTS MET FOR COBOL) (ALL REQ~
UIREMENTS MET FOR ASSMBLR) (ALL REQUIREMENTS MET FOR NU~
M_ANALYSIS) (ALL REQUIREMENTS MET FOR DATA_STRUC) (MISS~
ING THE COURSE (S) ((CMPSC 260 3))) (ALL REQUIREMENTS M~
ET FOR EE) (ALL REQUIREMENTS MET FOR MATH) (ALL REQUIRE~
MENTS MET FOR ENGL) (ALL REQUIREMENTS MET FOR L_STAT) (
ALL REQUIREMENTS MET FOR L_ALG) (ALL REQUIREMENTS MET F~
OR L_LABSCI) (REQUIREMENTS NOT MET FOR L_PHYS) (ALL REQ~
UIREMENTS MET FOR LR_LIT_SPEECH) (NOT ENOUGH HOURS FOR
R_SCIENG) (NOT ENOUGH HOURS FOR R_CSC) (NOT ENOUGH HOUR~
S FOR FREE_ELECTIVE_REQUIREMENT SINCE 19 MORE HOURS ARE
NEEDED))

APPENDIX B

EXECUTION WITH FIRST MODIFIED TRANSCRIPT

```
TR1

((CMPSC 1 1) (MA&ST 4 3) (MA&ST 6 2) (CHEM 1 4) (ENGTC
10 3) (HIST 176 3) (MA&ST 8 5) (CMPSC 260 3) (MTENG 1 1
) (CHEM 3 3) (CMPSC 73 2) (ENGMGT 130 3) (MA&ST 21 5) (
MILSC 10 1) (PHYSCS 23 4) (ENGLSH 1 3) (HIST 112 3) (CM~
PSC 74 3) (ECONOM 110 3) (MILSC 30 1) (PSYCH 50 3) (PHY~
ED 103 1) (PHYSCS 25 4) (PHYSCS 26 1) (CMPSC 83 3) (CMP~
SC 163 3) (ENGLSH 106 3) (MA&ST 204 3) (MILSC 20 1) (MI~
LSC 40 1) (PHIL 15 3) (CMPSC 183 3) (CMPSC 218 3) (ELEN~
G 61 3) (SP&MS 85 3) (CMPSC 168 3) (CMPSC 349 3) (CMPSC
 361 3) (MA&ST 343 3) (SP&MS 283 3) (CMPSC 268 3) (CMPS~
C 293 3))


DEGREE AUDIT REPORT

(MISSING THE COURSE (S) ((CMPSC 253 3)))
(MISSING THE COURSE (S) ((ELENG 211 3)))
(MISSING THE COURSE (S) ((MA&ST 22 4)))
(MISSING THE COURSE (S) ((ENGLSH 60 3)))
(REQUIREMENTS NOT MET FOR L_ALG)
(REQUIREMENTS NOT MET FOR L_LABSCI)
(NOT ENOUGH HOURS FOR R_SCIENG)
(NOT ENOUGH HOURS FOR FREE_ELECTIVE_REQUIREMENT SINCE 2~
4 MORE HOURS ARE NEEDED)


HISTORY

((NON_COUNTABLE ((MA&ST 4 3) (MA&ST 6 2) (MILSC 10 1) (
MILSC 20 1) (MILSC 30 1) (MILSC 40 1))) (CSOR ((CMPSC 1
 1))) (SCI_PROG ((CMPSC 73 2))) (JCL ((CMPSC 74 3))) (M~
CHLANG ((CMPSC 83 3))) (BLK_STR_LANG ((CMPSC 163 3))) (
COBOL ((CMPSC 168 3))) (ASSMBLR ((CMPSC 183 3))) (NUM_A~
NALYSIS ((CMPSC 218 3))) (DATA_STRUC NIL) (OPER_RES ((C~
MPSC 260 3))) (EE ((ELENG 61 3))) (MATH ((MA&ST 8 5) (M~
A&ST 21 5))) (ENGL ((ENGLSH 1 3))) (L_STAT ((MA&ST 343
 3))) (L_ALG NIL) (L_LABSCI ((CHEM 1 4))) (L_PHYS ((PHYS~
CS 23 4) (PHYSCS 25 4) (PHYSCS 26 1))) (LR_LIT_SPEECH (
(ENGLSH 106 3) (SP&MS 283 3))) (R_CONST ((HIST 112 3) (
HIST 176 3))) (R_PHIL ((PHIL 15 3))) (R_MATH ((MA&ST 20~
4 3))) (R_SOCSCI ((HIST 176 3) (ECONOM 110 3) (PSYCH 50
 3))) (R_SCIENG ((MTENG 1 1) (CHEM 3 3) (ENGMGT 130 3))
) (R_CSC ((CMPSC 349 3) (CMPSC 361 3) (CMPSC 268 3) (CM~
PSC 293 3))) (FREE_ELECTIVE_REQUIREMENT ((ENGTC 10 3) (
HIST 112 3) (PHYED 103 1) (SP&MS 85 3))))
```

BLK_BD

((THESE COURSES DIDN'T COUNT ((MA&ST 4 3) (MA&ST 6 2) (
MILSC 10 1) (MILSC 20 1) (MILSC 30 1) (MILSC 40 1))) (A~
LL REQUIREMENTS MET FOR CSOR) (ALL REQUIREMENTS MET FOR
 SCI_PROG) (ALL REQUIREMENTS MET FOR JCL) (ALL REQUIREM~
ENTS MET FOR MCHLANG) (ALL REQUIREMENTS MET FOR BLK_STR~
_LANG) (ALL REQUIREMENTS MET FOR COBOL) (ALL REQUIREMEN~
TS MET FOR ASSMBLR) (ALL REQUIREMENTS MET FOR NUM_ANALY~
SIS) (MISSING THE COURSE (S) ((CMPSC 253 3))) (ALL REQU~
IREMENTS MET FOR OPER_RES) (MISSING THE COURSE (S) ((EL~
ENG 211 3))) (MISSING THE COURSE (S) ((MA&ST 22 4))) (M~
ISSING THE COURSE (S) ((ENGLSH 60 3))) (ALL REQUIREMENT~
S MET FOR L_STAT) (REQUIREMENTS NOT MET FOR L_ALG) (REQ~
UIREMENTS NOT MET FOR L_LABSCI) (ALL REQUIREMENTS MET F~
OR L_PHYS) (ALL REQUIREMENTS MET FOR LR_LIT_SPEECH) (NO~
T ENOUGH HOURS FOR R_SCIENG) (NOT ENOUGH HOURS FOR FREE~
_ELECTIVE_REQUIREMENT SINCE 24 MORE HOURS ARE NEEDED))

APPENDIX C

EXECUTION WITH SECOND MODIFIED TRANSCRIPT

```
TR2

((PHYSCS 22 1) (CHEM 1 4) (CHEM 2 1) (ENGTC 10 3) (HIST
 176 3) (MA&ST 8 5) (MTENG 1 1) (CHEM 3 3) (CMPSC 73 2)
 (MA&ST 21 5) (PHYSCS 21 4) (ENGLSH 1 3) (HIST 112 3) (
CMPSC 74 3) (ECONOM 110 3) (MA&ST 22 4) (PHYED 103 1) (
PHYSCS 25 4) (PHYSCS 26 1) (CMPSC 83 3) (CMPSC 163 3) (
ENGLSH 106 3) (MA&ST 204 3) (PHIL 15 3) (CMPSC 183 3) (
CMPSC 218 3) (CMPSC 253 3) (ELENG 61 3) (SP&MS 85 3) (E~
NGLSH 60 3) (MA&ST 208 3) (CMPSC 168 3) (CMPSC 349 3) (
ELENG 211 3) (MA&ST 215 3) (SP&MS 283 3))


DEGREE AUDIT REPORT

(MISSING THE COURSE (S) ((CMPSC 1 1)))
(MISSING THE COURSE (S) ((CMPSC 260 3)))
(NOT ENOUGH HOURS FOR R_SCIENG)
(NOT ENOUGH WHERE_HRS FOR R_CSC)
(NOT ENOUGH HOURS FOR FREE_ELECTIVE_REQUIREMENT SINCE 2~
5 MORE HOURS ARE NEEDED)


HISTORY

((NON_COUNTABLE NIL) (CSOR NIL) (SCI_PROG ((CMPSC 73 2)
)) (JCL ((CMPSC 74 3))) (MCHLANG ((CMPSC 83 3))) (BLK_S~
TR_LANG ((CMPSC 163 3))) (COBOL ((CMPSC 168 3))) (ASSMB~
LR ((CMPSC 183 3))) (NUM_ANALYSIS ((CMPSC 218 3))) (DAT~
A_STRUC ((CMPSC 253 3))) (OPER_RES NIL) (EE ((ELENG 61
3) (ELENG 211 3))) (MATH ((MA&ST 8 5) (MA&ST 21 5) (MA&~
ST 22 4))) (ENGL ((ENGLSH 1 3) (ENGLSH 60 3))) (L_STAT
((MA&ST 215 3))) (L_ALG ((MA&ST 208 3))) (L_LABSCI ((CH~
EM 1 4) (CHEM 2 1))) (L_PHYS ((PHYSCS 22 1) (PHYSCS 21
4) (PHYSCS 25 4) (PHYSCS 26 1))) (LR_LIT_SPEECH ((ENGLS~
H 106 3) (SP&MS 283 3))) (R_CONST ((HIST 112 3) (HIST 1~
76 3))) (R_PHIL ((PHIL 15 3))) (R_MATH ((MA&ST 204 3)))
 (R_SOCSCI ((HIST 176 3) (HIST 112 3) (ECONOM 110 3)))
(R_SCIENG ((MTENG 1 1) (CHEM 3 3))) (R_CSC ((CMPSC 349
3))) (FREE_ELECTIVE_REQUIREMENT ((ENGTC 10 3) (PHYED 10~
3 1) (SP&MS 85 3))))


BLK_BD

((MISSING THE COURSE (S) ((CMPSC 1 1))) (ALL REQUIREMEN~
TS MET FOR SCI_PROG) (ALL REQUIREMENTS MET FOR JCL) (AL~
L REQUIREMENTS MET FOR MCHLANG) (ALL REQUIREMENTS MET F~
OR BLK_STR_LANG) (ALL REQUIREMENTS MET FOR COBOL) (ALL
```

```
REQUIREMENTS MET FOR ASSMBLR) (ALL REQUIREMENTS MET FOR
 NUM_ANALYSIS) (ALL REQUIREMENTS MET FOR DATA_STRUC) (M~
ISSING THE COURSE (S) ((CMPSC 260 3))) (ALL REQUIREMENT~
S MET FOR EE) (ALL REQUIREMENTS MET FOR MATH) (ALL REQU~
IREMENTS MET FOR ENGL) (ALL REQUIREMENTS MET FOR L_STAT
) (ALL REQUIREMENTS MET FOR L_ALG) (ALL REQUIREMENTS ME~
T FOR L_LABSCI) (ALL REQUIREMENTS MET FOR L_PHYS) (ALL
REQUIREMENTS MET FOR LR_LIT_SPEECH) (NOT ENOUGH HOURS F~
OR R_SCIENG) (NOT ENOUGH WHERE_HRS FOR R_CSC) (NOT ENOU~
GH HOURS FOR FREE_ELECTIVE_REQUIREMENT SINCE 25 MORE HO~
URS ARE NEEDED))
```

APPENDIX D

EXECUTION WITH THRID MODIFIED TRANSCRIPT

```
TR3

((CHEM 1 4) (ENGLSH 110 3) (CHEM 2 1) (ENGTC 10 3) (ECO~
NOM 111 3) (MA&ST 8 5) (MTENG 1 1) (CHEM 3 3) (CMPSC 73
 2) (MA&ST 21 5) (ECONOM 215 3) (PHYSCS 23 4) (ENGLSH 1
 3) (CMPSC 74 3) (ECONOM 110 3) (MA&ST 22 4) (PHYED 103
 1) (PHYSCS 24 4) (CMPSC 83 3) (CMPSC 163 3) (ENGLSH 10~
6 3) (CMPSC 183 3) (CMPSC 218 3) (CMPSC 253 3) (ELENG 6~
1 3) (SP&MS 85 3) (CMPSC 101 3) (ENGLSH 60 3) (MA&ST 20~
8 3) (CMPSC 168 3) (CMPSC 349 3) (CMPSC 361 3) (ELENG 2~
11 3) (MA&ST 215 3) (SP&MS 283 3) (CMPSC 264 3) (CMPSC
268 3))


DEGREE AUDIT REPORT

(MISSING THE COURSE (S) ((CMPSC 1 1)))
(MISSING THE COURSE (S) ((CMPSC 260 3)))
(NO COURSES FOUND FOR R_CONST)
(NO COURSES IN THE NEEDED DEPT FOR R_PHIL)
(NO COURSES IN THE NEEDED DEPT FOR R_MATH)
(NOT ENOUGH AREAS FOR R_SOCSCI)
(NOT ENOUGH HOURS FOR R_SCIENG)
(NOT ENOUGH HOURS FOR FREE_ELECTIVE_REQUIREMENT SINCE 1~
8 MORE HOURS ARE NEEDED)


HISTORY

((NON_COUNTABLE NIL) (CSOR NIL) (SCI_PROG ((CMPSC 73 2)
)) (JCL ((CMPSC 74 3))) (MCHLANG ((CMPSC 83 3))) (BLK_S~
TR_LANG ((CMPSC 163 3))) (COBOL ((CMPSC 168 3))) (ASSMB~
LR ((CMPSC 183 3))) (NUM_ANALYSIS ((CMPSC 218 3))) (DAT~
A_STRUC ((CMPSC 253 3))) (OPER_RES NIL) (EE ((ELENG 61
3) (ELENG 211 3))) (MATH ((MA&ST 8 5) (MA&ST 21 5) (MA&~
ST 22 4))) (ENGL ((ENGLSH 1 3) (ENGLSH 60 3))) (L_STAT
((MA&ST 215 3))) (L_ALG ((MA&ST 208 3))) (L_LABSCI ((CH~
EM 1 4) (CHEM 2 1))) (L_PHYS ((PHYSCS 23 4) (PHYSCS 24
4))) (LR_LIT_SPEECH ((ENGLSH 110 3) (ENGLSH 106 3))) (R~
_CONST NIL) (R_PHIL NIL) (R_MATH NIL) (R_SOCSCI ((ECONO~
M 111 3) (ECONOM 215 3) (ECONOM 110 3))) (R_SCIENG ((MT~
ENG 1 1) (CHEM 3 3))) (R_CSC ((CMPSC 101 3) (CMPSC 349
3) (CMPSC 361 3) (CMPSC 264 3))) (FREE_ELECTIVE_REQUIRE~
MENT ((ENGTC 10 3) (PHYED 103 1) (SP&MS 85 3) (SP&MS 28~
3 3) (CMPSC 268 3))))
```

BLK_BD

((MISSING THE COURSE (S) ((CMPSC 1 1))) (ALL REQUIREMEN~
TS MET FOR SCI_PROG) (ALL REQUIREMENTS MET FOR JCL) (AL~
L REQUIREMENTS MET FOR MCHLANG) (ALL REQUIREMENTS MET F~
OR BLK_STR_LANG) (ALL REQUIREMENTS MET FOR COBOL) (ALL
REQUIREMENTS MET FOR ASSMBLR) (ALL REQUIREMENTS MET FOR
NUM_ANALYSIS) (ALL REQUIREMENTS MET FOR DATA_STRUC) (M~
ISSING THE COURSE (S) ((CMPSC 260 3))) (ALL REQUIREMENT~
S MET FOR EE) (ALL REQUIREMENTS MET FOR MATH) (ALL REQU~
IREMENTS MET FOR ENGL) (ALL REQUIREMENTS MET FOR L_STAT
) (ALL REQUIREMENTS MET FOR L_ALG) (ALL REQUIREMENTS ME~
T FOR L_LABSCI) (ALL REQUIREMENTS MET FOR L_PHYS) (ALL
REQUIREMENTS MET FOR LR_LIT_SPEECH) (NO COURSES FOUND F~
OR R_CONST) (NO COURSES IN THE NEEDED DEPT FOR R_PHIL)
(NO COURSES IN THE NEEDED DEPT FOR R_MATH) (NOT ENOUGH
AREAS FOR R_SOCSCI) (NOT ENOUGH HOURS FOR R_SCIENG) (NO~
T ENOUGH HOURS FOR FREE_ELECTIVE_REQUIREMENT SINCE 18 M~
ORE HOURS ARE NEEDED))

APPENDIX E

EXECUTION WITH FOURTH MODIFIED TRANSCRIPT

TR4

```
((CMPSC 1 1) (CHEM 1 4) (ENGLSH 110 3) (CHEM 2 1) (ENGT~
C 10 3) (CMPSC 260 3) (ECONOM 111 3) (MA&ST 8 5) (PSYCH
 50 3) (POLYSCI 90 3) (CHEM 3 3) (CMPSC 73 2) (MA&ST 21
 5) (PHIL 10 3) (ECONOM 215 3) (PHYSCS 23 4) (ENGLSH 1
3) (CMPSC 74 3) (ECONOM 110 3) (MA&ST 209 3) (MA&ST 22
4) (PHYED 103 1) (PHYSCS 24 4) (MA&ST 115 3) (CMPSC 83
3) (CMPSC 163 3) (ENGLSH 106 3) (CMPSC 183 3) (CMPSC 21~
8 3) (CMPSC 253 3) (ELENG 61 3) (SP&MS 85 3) (CMPSC 101
 3) (ENGLSH 60 3) (MA&ST 208 3) (CMPSC 168 3) (CMPSC 34~
9 3) (CMPSC 361 3) (ELENG 211 3) (MA&ST 215 3) (SP&MS 2~
83 3) (CMPSC 264 3) (CMPSC 268 3))
```

DEGREE AUDIT REPORT

CONGRATULATIONS!!!

ALL GRADUATION REQUIREMENTS HAVE BEEN FULFILLED!!

HISTORY

```
((NON_COUNTABLE NIL) (CSOR ((CMPSC 1 1))) (SCI_PROG ((C~
MPSC 73 2))) (JCL ((CMPSC 74 3))) (MCHLANG ((CMPSC 83 3
))) (BLK_STR_LANG ((CMPSC 163 3))) (COBOL ((CMPSC 168 3
))) (ASSMBLR ((CMPSC 183 3))) (NUM_ANALYSIS ((CMPSC 218
 3))) (DATA_STRUC ((CMPSC 253 3))) (OPER_RES ((CMPSC 26~
0 3))) (EE ((ELENG 61 3) (ELENG 211 3))) (MATH ((MA&ST
8 5) (MA&ST 21 5) (MA&ST 22 4))) (ENGL ((ENGLSH 1 3) (E~
NGLSH 60 3))) (L_STAT ((MA&ST 215 3))) (L_ALG ((MA&ST 2~
08 3))) (L_LABSCI ((CHEM 1 4) (CHEM 2 1))) (L_PHYS ((PH~
YSCS 23 4) (PHYSCS 24 4))) (LR_LIT_SPEECH ((ENGLSH 110
3) (ENGLSH 106 3))) (R_CONST ((POLYSCI 90 3))) (R_PHIL
((PHIL 10 3))) (R_MATH ((MA&ST 209 3))) (R_SOCSCI ((ECO~
NOM 111 3) (PSYCH 50 3) (POLYSCI 90 3))) (R_SCIENG ((CH~
EM 3 3) (ECONOM 215 3) (ECONOM 110 3))) (R_CSC ((CMPSC
101 3) (CMPSC 349 3) (CMPSC 361 3) (CMPSC 264 3))) (FRE~
E_ELECTIVE_REQUIREMENT ((ENGTC 10 3) (PHYED 103 1) (MA&~
ST 115 3) (SP&MS 85 3) (SP&MS 283 3) (CMPSC 268 3))))
```

BLK_BD

```
((ALL REQUIREMENTS MET FOR CSOR) (ALL REQUIREMENTS MET
FOR SCI_PROG) (ALL REQUIREMENTS MET FOR JCL) (ALL REQUI~
REMENTS MET FOR MCHLANG) (ALL REQUIREMENTS MET FOR BLK_~
STR_LANG) (ALL REQUIREMENTS MET FOR COBOL) (ALL REQUIRE~
MENTS MET FOR ASSMBLR) (ALL REQUIREMENTS MET FOR NUM_AN~
```

```
ALYSIS) (ALL REQUIREMENTS MET FOR DATA_STRUC) (ALL REQU~
IREMENTS MET FOR OPER_RES) (ALL REQUIREMENTS MET FOR EE
) (ALL REQUIREMENTS MET FOR MATH) (ALL REQUIREMENTS MET
 FOR ENGL) (ALL REQUIREMENTS MET FOR L_STAT) (ALL REQUI~
REMENTS MET FOR L_ALG) (ALL REQUIREMENTS MET FOR L_LABS~
CI) (ALL REQUIREMENTS MET FOR L_PHYS) (ALL REQUIREMENTS
 MET FOR LR_LIT_SPEECH) (THESE WERE EXTRA COURSES NIL))
```

APPENDIX F

EXECUTION WITH FIFTH MODIFIED TRANSCRIPT

```
TR5

((SOCIOL 81 3) (CMPSC 1 1) (CHEM 1 4) (ENGLSH 110 3) (C~
HEM 2 1) (ENGTC 10 3) (CMPSC 260 3) (ECONOM 111 3) (MA&~
ST 8 5) (PSYCH 52 3) (PSYCH 50 3) (POLYSCI 90 3) (CHEM
3 3) (CMPSC 73 2) (MA&ST 21 5) (PHIL 10 3) (ECONOM 215
3) (PHYSCS 23 4) (ENGLSH 1 3) (CMPSC 74 3) (ECONOM 110
3) (MA&ST 209 3) (MA&ST 22 4) (PHYED 103 1) (PHYSCS 24
4) (MA&ST 115 3) (CMPSC 83 3) (CMPSC 163 3) (ENGLSH 106
 3) (CMPSC 183 3) (CMPSC 218 3) (CMPSC 253 3) (ELENG 61
 3) (SP&MS 85 3) (CMPSC 101 3) (ENGLSH 60 3) (MA&ST 208
 3) (CMPSC 168 3) (CMPSC 349 3) (CMPSC 361 3) (ELENG 21~
1 3) (MA&ST 215 3) (SP&MS 283 3) (CMPSC 264 3) (CMPSC 2~
68 3))


DEGREE AUDIT REPORT

CONGRATULATIONS!!!

ALL GRADUATION REQUIREMENTS HAVE BEEN FULFILLED!!


HISTORY

((NON_COUNTABLE NIL) (CSOR ((CMPSC 1 1))) (SCI_PROG ((C~
MPSC 73 2))) (JCL ((CMPSC 74 3))) (MCHLANG ((CMPSC 83 3
))) (BLK_STR_LANG ((CMPSC 163 3))) (COBOL ((CMPSC 168 3
))) (ASSMBLR ((CMPSC 183 3))) (NUM_ANALYSIS ((CMPSC 218
 3))) (DATA_STRUC ((CMPSC 253 3))) (OPER_RES ((CMPSC 26~
0 3))) (EE ((ELENG 61 3) (ELENG 211 3))) (MATH ((MA&ST
8 5) (MA&ST 21 5) (MA&ST 22 4))) (ENGL ((ENGLSH 1 3) (E~
NGLSH 60 3))) (L_STAT ((MA&ST 215 3))) (L_ALG ((MA&ST 2~
08 3))) (L_LABSCI ((CHEM 1 4) (CHEM 2 1))) (L_PHYS ((PH~
YSCS 23 4) (PHYSCS 24 4))) (LR_LIT_SPEECH ((ENGLSH 110
3) (ENGLSH 106 3))) (R_CONST ((POLYSCI 90 3))) (R_PHIL
((PHIL 10 3))) (R_MATH ((MA&ST 209 3))) (R_SOCSCI ((ECO~
NOM 111 3) (PSYCH 52 3) (POLYSCI 90 3))) (R_SCIENG ((CH~
EM 3 3) (ECONOM 215 3) (ECONOM 110 3))) (R_CSC ((CMPSC
101 3) (CMPSC 349 3) (CMPSC 361 3) (CMPSC 264 3))) (FRE~
E_ELECTIVE_REQUIREMENT ((SOCIOL 81 3) (ENGTC 10 3) (PSY~
CH 50 3) (PHYED 103 1) (MA&ST 115 3) (SP&MS 85 3))))


BLK_BD

((ALL REQUIREMENTS MET FOR CSOR) (ALL REQUIREMENTS MET
FOR SCI_PROG) (ALL REQUIREMENTS MET FOR JCL) (ALL REQUI~
REMENTS MET FOR MCHLANG) (ALL REQUIREMENTS MET FOR BLK_~
```

```
STR_LANG) (ALL REQUIREMENTS MET FOR COBOL) (ALL REQUIRE~
MENTS MET FOR ASSMBLR) (ALL REQUIREMENTS MET FOR NUM_AN~
ALYSIS) (ALL REQUIREMENTS MET FOR DATA_STRUC) (ALL REQU~
IREMENTS MET FOR OPER_RES) (ALL REQUIREMENTS MET FOR EE
) (ALL REQUIREMENTS MET FOR MATH) (ALL REQUIREMENTS MET
 FOR ENGL) (ALL REQUIREMENTS MET FOR L_STAT) (ALL REQUI~
REMENTS MET FOR L_ALG) (ALL REQUIREMENTS MET FOR L_LABS~
CI) (ALL REQUIREMENTS MET FOR L_PHYS) (ALL REQUIREMENTS
 MET FOR LR_LIT_SPEECH) (THESE WERE EXTRA COURSES ((SP&~
MS 283 3) (CMPSC 268 3))))
```

APPENDIX G

DAES SOURCE CODE

```
~
~  RULES
~
(PRINT 'RULES)

(PACKAGE RULES
    (DEF PNTR W_N_F REPORT SK DEGREE_AUDIT F_R_NCT EVAL_NO_NILS SUM_HRS
    RECORD_CRS_USD FIND_CRS_USD X_PS X_L LOG_EVAL EVAL_GIVING_NILS
    VAPPEND VOR VAND X_RL F_KEYWDS R_R F_D F_R_CNT_HRS F_RM F_SD
    F_R_ALL RM_P MODIFY RM_C R_HRS_CK R_CRS_CK RETURN_UNIQ_DEPTS RM_DC
    R_XHRS X_LOGIC_RULES EVAL_LR L_R X_FE WHAT_FILLED WHATS_NOT_FILLED
    CREATE_HISTORY BLD_HIST WHY WRITER EVAL_WHY BLD_EXPL WHY_LR EDIT_EXPR
    WHY_RULES SHOW_NAMES PP_WRITER BLD_TRANS ADD_TO_TRANS CK_DUP_ADDS
    DELETE_FROM_TRANS)
    (SETQ TR1 TR2 TR3 TR4 TR5 ROLLA_TRANS NON_COUNTABLE PRE_SEL CSOR SCI_PROG
    JCL MCHLANG BLK_STR_LANG COBOL ASSMBLR NUM_ANALYSIS DATA_STRUC OPER_RES EE
    MATH ENGL LOGIC_LIST L_STAT L_ALG L_LABSCI L_PHYS LR_LIT_SPEECH LIT
    RULE_LIST R_PHIL R_CONST CONST R_MATH R_SOCSCI R_SCIENG BS_NOT_CS R_CSC
    SOC_SCI FREE_ELECTIVE_REQUIREMENT TRANS BLK_BD HISTORY))

(DEF 'PNTR      ~PRINTS PROGRAM RESULTS TO A FILE
    '[LAMBDA ()         ~INSTEAD OF THE SCREEN
         (MAPC '[LAMBDA (X)
                    (PRINTC X RESULTS)]
             (W_N_F))
         (COND
             [(NULL REPORT)     ~ALL REQUIREMENTS HAVE BEEN MET
               (PRINTC 'CONGRATULATIONS!!! RESULTS)
               (SK 2)
               (MAPC '[LAMBDA (X)
                          (PRINC X RESULTS)]
                   '(
```

```
                    ALL\ GRADUATION\ REQUIREMENTS\ HAVE\ BEEN\ FULFILLED~
!!)))]))]

(DEF 'W_N_F     ~PLACES WHAT'S NOT BEEN FILLED IN A FILE CONTAINING
    '[LAMBDA ()          ~PROGRAM RESULTS
         (SUBSET '[LAMBDA (CBB)
                       (SETQ REPORT
                           (COND
                               [(SOME '[LAMBDA (CL)
                                           (COND
                                               [(MEMB CL CBB)
                                                 CBB]
                                               [T
                                                 NIL])]
                                      '(NO NOT MISSING))
                                 CBB]
                                [T
                                 NIL]))] BLK_BD)])

(DEF 'REPORT    ~FORMATS OUTPUT FILE REPORT OF INTERACTIVE
    '[LAMBDA ()          ~PROGRAM RESULTS
         (SETQ RESULTS            ~CREATE OUTPUT FILE
             (OUTPUT "RESULTS"))
         (LINELENGTH 55 RESULTS)     ~SET LINE LENGTH OF OUTPUT FILE
         (SETQ TRANS ROLLA_TRANS)   ~SELECT ACTUAL TRANSCRIPT
         (PRINTC 'ROLLA_TRANS RESULTS)
         (SK 2)
         (PRINTC TRANS RESULTS)
         (SK 3)
         (DEGREE_AUDIT)
         (PRINTC 'DEGREE\ AUDIT\ REPORT RESULTS)
         (SK 2)
         (PNTR)
         (SK 3)
         (PRINTC 'HISTORY RESULTS)
         (SK 2)
```

```
(PRINTC HISTORY RESULTS)
(SK 3)
(PRINTC 'BLK_BD RESULTS)
(SK 2)
(PRINTC BLK_BD RESULTS)
(SK 3)
(SETQ TRANS TR1)        ~SELECT FIRST MODIFIED TRANSCRIPT
(PRINTC 'TR1 RESULTS)
(SK 2)
(PRINTC TRANS RESULTS)
(SK 3)
(DEGREE_AUDIT)
(PRINTC 'DEGREE\ AUDIT\ REPORT RESULTS)
(SK 2)
(PNTR)
(SK 3)
(PRINTC 'HISTORY RESULTS)
(SK 2)
(PRINTC HISTORY RESULTS)
(SK 3)
(PRINTC 'BLK_BD RESULTS)
(SK 2)
(PRINTC BLK_BD RESULTS)
(SK 3)
(SETQ TRANS TR2)        ~SELECT SECOND MODIFIED TRANSCRIPT
(PRINTC 'TR2 RESULTS)
(SK 2)
(PRINTC TRANS RESULTS)
(SK 3)
(DEGREE_AUDIT)
(PRINTC 'DEGREE\ AUDIT\ REPORT RESULTS)
(SK 2)
(PNTR)
(SK 3)
(PRINTC 'HISTORY RESULTS)
(SK 2)
```

```
(PRINTC HISTORY RESULTS)
(SK 3)
(PRINTC 'BLK_BD RESULTS)
(SK 2)
(PRINTC BLK_BD RESULTS)
(SK 3)
(SETQ TRANS TR3)        ~SELECT THRID MODIFIED TRANSCRIPT
(PRINTC 'TR3 RESULTS)
(SK 2)
(PRINTC TRANS RESULTS)
(SK 3)
(DEGREE_AUDIT)
(PRINTC 'DEGREE\ AUDIT\ REPORT RESULTS)
(SK 2)
(PNTR)
(SK 3)
(PRINTC 'HISTORY RESULTS)
(SK 2)
(PRINTC HISTORY RESULTS)
(SK 3)
(PRINTC 'BLK_BD RESULTS)
(SK 2)
(PRINTC BLK_BD RESULTS)
(SK 3)
(SETQ TRANS TR4)        ~SELECT FOURTH MODIFIED TRANSCRIPT
(PRINTC 'TR4 RESULTS)
(SK 2)
(PRINTC TRANS RESULTS)
(SK 3)
(DEGREE_AUDIT)
(PRINTC 'DEGREE\ AUDIT\ REPORT RESULTS)
(SK 2)
(PNTR)
(SK 3)
(PRINTC 'HISTORY RESULTS)
(SK 2)
```

```
        (PRINTC HISTORY RESULTS)
        (SK 3)
        (PRINC 'BLK_BD RESULTS)
        (SK 2)
        (PRINTC BLK_BD RESULTS)
        (SK 3)
        (SETQ TRANS TR5)        ~SELECT FIFTH MODIFIED TRANSCRIPT
        (PRINTC 'TR5 RESULTS)
        (SK 2)
        (PRINTC TRANS RESULTS)
        (SK 3)
        (DEGREE_AUDIT)
        (PRINTC 'DEGREE\ AUDIT\ REPORT RESULTS)
        (SK 2)
        (PNTR)
        (SK 3)
        (PRINTC 'HISTORY RESULTS)
        (SK 2)
        (PRINTC HISTORY RESULTS)
        (SK 3)
        (PRINTC 'BLK_BD RESULTS)
        (SK 2)
        (PRINTC BLK_BD RESULTS)])

(DEF 'SK        ~SKIPS ANY NUMBER OF BLANK LINES ON THE
    '[LAMBDA (SKIP_NUM)         ~OUTPUT FILE
        (FOR I FROM 1 TO SKIP_NUM
            (TERPRI " " RESULTS))])

(DEF 'DEGREE_AUDIT      ~DRIVER FUNCTION
    '[LAMBDA ()
        [PROG ()
            (MAPC 'PRINC
                '(CHECKING\ FOR\ NON_COUNTABLE\ COURSES))
            (TERPRI)
            (F_R_NCT NON_COUNTABLE)    ~LOCATE ANY COURSES NOT COUNTING
```

```
(MAPC 'PRINC
    '(EXECUTING\ PRE_SELECTED\ REQUIREMENTS))
(TERPRI)
(X_PS)                          ~EXECUTE PRE-SELECTED REQUIREMENTS
(MAPC 'PRINC
    '(EXECUTING\ LOGIC\ REQUIREMENTS))
(TERPRI)
(X_L)                           ~EXECUTE LOGIC REQUIREMENTS
(MAPC 'PRINC
    '(EXECUTING\ LOGIC_RULES))
(TERPRI)
(X_LOGIC_RULES LR_LIT_SPEECH)       ~EXECUTE LOGIC RULES
(MAPC 'PRINC
    '(EXECUTING\ RULES))
(TERPRI)
(X_RL)                          ~EXECUTE RULE LIST
(MAPC 'PRINC
    '(CHECKING\ FREE\ ELECTIVES))
(X_FE)                          ~EXECUTE FREE ELECTIVE CHECK
(FOR LINESPACE FROM 1 TO 15
    (TERPRI))
(MAPC 'PRINC
    '(DEGREE\ AUDIT\ REPORT))
(TERPRI)
(MAPC 'PRINT                    ~PRINT FINDINGS
    (WHATS_NOT_FILLED))
(COND
    [(NULL REPORT)
      (PRINT 'CONGRATULATIONS!!!)
      (TERPRI)
      (TERPRI)
      (MAPC 'PRINC
          '(
           ALL\ GRADUATION\ REQUIREMENTS\ HAVE\ BEEN\ FULFI~
LLED!!))
      (TERPRI)])])
```

```
(DEF 'F_R_NCT           ~FIND AND REMOVE NON-COUNTABLE COURSES
    '[LAMBDA (LIST)
         (SETQ BLK_BD NIL)
         (COND
             [(SETQ N_CL        ~LOCATE AND STORE NON-COUNTABLE COURSES
                  (EVAL_NO_NILS LIST))
               (SETQ R_HRSCT 0)
               (MAPC 'RM_C N_CL) ~REMOVE NON-COUNTABLE COURSES FROM TRANS AND
               (MODIFY '(THESE COURSES DIDN'T COUNT) N_CL)]  ~RECORD THEM
             [T
               'ALL_COURSES_COUNT])
         (SETQ VNON_COUNTABLE N_CL)
         (SETQ SUMHRS 0)
         (SETQ TCNTHRS          ~FIND NUMBER OF TOTAL COUNTABLE HOURS
             (SUM_HRS TRANS))])

(DEF 'EVAL_NO_NILS      ~SEARCH TRANS WITHOUT RETURNING NIL IF COURSE IS
    '[LAMBDA (REQ)                                  ~NOT LOCATED
         (SUBSET '[LAMBDA (CR)
                     (SUBSET '[LAMBDA (CT)
                                 (COND
                                     [(EQUAL CR CT)
                                       CR]
                                     [T
                                       NIL])] TRANS)] REQ)])

(DEF 'SUM_HRS           ~RETURN HOURS SUM OF ANY LIST
    '[LAMBDA (LIST)
         (COND
             [(CAR LIST)
               (SETQ SUMHRS
                   (+ SUMHRS
                       (CAR (LAST (CAR LIST)))))
               (SUM_HRS (CDR LIST))]
             [T
```

```
              SUMHRS])])

(DEF 'RECORD_CRS_USD    ~RECORD COURSES USED TO FULFILL A REQUIREMENT
    '[LAMBDA (REQ_NAME)
        (SETQ IDENT
            (VAPPEND REQ_NAME))
        (SET IDENT NIL)
        (FOR COUNTER FROM 1 TO ~DETERMINE HOW MANY COURSES WERE USED
            (- (LENGTH TRANSB)          ~TO FULFILL THIS REQUIREMENT
               (LENGTH TRANS))
            (FIND_CRS_USD))])

(DEF 'FIND_CRS_USD      ~ASSISTS IN RECORDING PROCESS
    '[LAMBDA ()
        (COND
            [(SETQ CRM
                (CAR (SOME '[LAMBDA (CL)
                                (EVERY '[LAMBDA (CT)
                                           (COND
                                              [(EQUAL CL CT)
                                                NIL]
                                              [T
                                                CL])] TRANS)]
                           TRANSB)))
              (SETQ TRANSB
                 (F_RM CRM TRANSB))
              (SET IDENT
                 (APPEND (EVAL IDENT)
                    (LIST CRM)))]
            [T
              'NOTHING_TO_DO])])

(DEF 'X_PS      ~EXECUTE PRE-SELECTED REQUIREMENTS
    '[LAMBDA ()
        (SETQ TRANSB TRANS)
        (MAPC '[NLAMBDA (CPS)
```

```
                    (SETQ R_NAME CPS)
                    (COND
                        [(SETQ F_CL     ~SEARCH FOR COURSE OR COURSES
                            (EVAL_NO_NILS (EVAL CPS)))
                          (SETQ NOT_TAKEN
                              (EVAL R_NAME))
                          (MAPC '[LAMBDA (COURSE)
                                    (SETQ NOT_TAKEN
                                        (F_RM COURSE NOT_TAKEN))] F_CL)
                          (COND
                              [(EQUAL NOT_TAKEN NIL)      ~RECORD RESULTS
                                (MODIFY '(ALL REQUIREMENTS MET FOR)
                                R_NAME)
                                (MAPC 'RM_C
                                    (EVAL R_NAME))]
                              [T
                                (MODIFY '(MISSING THE COURSE
                                                (S)) NOT_TAKEN)
                                (MAPC 'RM_C F_CL)]}]
                        [T
                          (MODIFY '(MISSING THE COURSE
                                        (S))
                              (EVAL R_NAME))])
                    (RECORD_CRS_USD CPS)] PRE_SEL)])

(DEF 'X_L               ~EXECUTE LOGIC REQUIREMENTS
    '[LAMBDA ()
        (MAPC '[NLAMBDA (CLL)
                    (SETQ LIST_HOLDER NIL)
                    (SETQ F_CL NIL)
                    (COND
                        [(SETQ C_L      ~SEARCH AND STORE COURSES FOUND
                            (F_R_ALL (RM_P (LOG_EVAL (EVAL CLL))) NIL)
                         )
                          (MAPC 'RM_C C_L)
                          (MODIFY '(ALL REQUIREMENTS MET FOR) CLL)]
```

```
                        [T
                          (MODIFY '(REQUIREMENTS NOT MET FOR) CLL)
                          (SETQ F_CL
                              (F_R_ALL (RM_P F_CL) NIL))
                          (MAPC 'RM_DC F_CL)
                          (MAPC 'RM_C F_CL)])
                    (RECORD_CRS_USD CLL)] LOGIC_LIST)])

(DEF 'LOG_EVAL          ~EVALUATES LOGIC REQUIREMENTS
    '[LAMBDA (CLLR)
        (COND
            [(ATOM (CAR CLLR))
              (EVAL_GIVING_NILS (LIST CLLR))]
            [(AND (ATOM (CAAR CLLR))
                  (NULL (CDR CLLR)))
              (EVAL_GIVING_NILS CLLR)]
            [(MEMB (CADR CLLR)
                   '(OR AND))
              (APPLY (VAPPEND (CADR CLLR))
                  (LIST (CAR (LOG_EVAL (CDDR CLLR)))
                        (CAR (LOG_EVAL (LIST (CAR CLLR))))))]
            [T
              (LIST (APPLY (VAPPEND (CADAR CLLR))
                           (LIST (CAR (LOG_EVAL (CDDAR CLLR)))
                                 (CAR (LOG_EVAL (LIST (CAAR CLLR))))))])])

(DEF 'EVAL_GIVING_NILS  ~SEARCHES TRANS AND RETURNS NIL IF COURSE IS NOT FOUND
    '[LAMBDA (REQ)
        (MAPCAR '[LAMBDA (CR)
                    (CAR (SUBSET '[LAMBDA (CT)
                                     (COND
                                         [(EQUAL CR CT)
                                           CT]
                                         [T
                                           NIL])] TRANS))] REQ)])
```

```
(DEF 'VAPPEND           ~PREFIXES ARGUMENT WITH LETTER V
    '[LAMBDA (BOOL)
        (READLIST (CONS '"V"
                        (EXPLODE BOOL)))])

(DEF 'VOR       ~COMPUTES LOGICAL OPERATION "OR"
    '[LAMBDA (X Y)
        (COND
            [(NOT (NULL X))
              (SETQ F_CL
                  (APPEND F_CL
                      (LIST X)))
              (LIST X)]
            [(NOT (NULL Y))
              (SETQ F_CL
                  (APPEND F_CL
                      (LIST Y)))
              (LIST Y)]
            [T
              NIL])])

(DEF 'VAND      ~COMPUTES LOGICAL OPERATION "AND"
    '[LAMBDA (X Y)
        (COND
            [(AND (CAR X)
                  (CAR Y))
              (SETQ F_CL
                  (APPEND F_CL
                      (LIST X Y)))
              (APPEND (LIST X)
                  (LIST Y))]
            [T
              (COND
                  [(NOT (NULL X))
                    (SETQ F_CL
                        (APPEND F_CL
```

```
                                (LIST X)))]
                        [(NOT (NULL Y))
                          (SETQ F_CL
                              (APPEND F_CL
                                  (LIST Y)))])
                    NIL])])

(DEF 'X_RL      ~EXECUTES RULE LIST
    '[LAMBDA ()
         (MAPC '[NLAMBDA (CRL)
                     (SETQ F_CL
                         (R_R (EVAL CRL)))       ~INVOKES RULE RECOGNIZER
                     (COND
                         [(NULL (MEMB 'NONCONSUMABLE F_WL))
                           (RECORD_CRS_USD CRL)] ~RECORDS COURSES USED
                         [T
                           'NOTHING_TO_DO])] RULE_LIST)
         (SETQ SUMHRS 0)
         (SETQ HRSLEFT  ~FINDS NUMBER OF HOURS NOT YET USED AFTER ALL
             (SUM_HRS TRANS))])          ~RULES HAVE BEEN FIRED

(DEF 'F_KEYWDS          ~FINDS AND STORES KEYWORDS AND THEIR ASSOCIATED VALUES
    '[LAMBDA (L)
         (SUBSET '[LAMBDA (WORD)
                     (COND
                         [(MEMB WORD L)
                           (SETQ WRD WORD)
                           (SET WORD   ~RECORDS THE VALUE OF EACH
                               (CADR (MEMB WORD L)))   ~KEYWORD
                           WRD]
                         [T
                           NIL])]
             '(LEVEL DEPT CRS WHERE_HRS HRS DEPTS COURSES AREAS
              NONCONSUMABLE))])

(DEF 'R_R                       ~RULE RECOGNIZER
```

```
'[LAMBDA (RULE)
     (MAPC '[LAMBDA (LIST)  ~INITIALIZES LISTS TO NIL
                (SET LIST NIL)]
         '(F_WL D_L F_LL S_L RM_CL))
     (SETQ F_WL     ~STORES KEYWORDS AND THEIR ASSOCIATED VALUES
         (F_KEYWDS RULE))
     (COND
         [(MEMB 'DEPT F_WL)          ~DEPARTMENT HANDLER
           (COND
               [(SETQ D_L
                    (F_D DEPT))
                 (COND
                     [(MEMB 'HRS F_WL)       ~DEPARTMENT-HOURS HANDLER
                       (SETQ SUMHRS 0)
                       (SUM_HRS D_L)
                       (COND
                           [(LE (LENGTH F_WL) 2)    ~REMOVE COURSES IF
                             (COND          ~ONLY DEPT AND HRS CONSTRAINTS
                                 [(GE SUMHRS HRS)   ~ARE PRESENT
                                   (SETQ R_HRSCT 0)
                                   (R_HRS_CK D_L HRS)]
                                 [T
                                   (MODIFY '(NOT ENOUGH HOURS FOR)
                                   CRL)
                                   (SETQ R_HRSCT 0)
                                   (R_HRS_CK D_L SUMHRS)])]
                           [T
                             'DONT_DELETE_FROM_TRANS])]
                     [T
                       'NO_HRS])]
                [T
                  (MODIFY '(NO COURSES IN THE NEEDED DEPT FOR) CRL)]
           )]
         [T
           'NO_DEPT])
     (COND
```

```
    [(MEMB 'COURSES F_WL)       ~COURSES HANDLER
      (COND
          [(SETQ S_L            ~STORE SPECIFIC COURSES ON TRANS
               (EVAL_NO_NILS (EVAL COURSES)))  ~GIVEN IN RULE
            ]
          [T
            (MODIFY '(NO COURSES FOUND FOR) CRL)])]
    [T
      'NO_COURSES_KEYWORD])
(COND
    [(MEMB 'DEPTS F_WL)         ~DEPARTMENTS HANDLER
      (COND
          [(SETQ S_L            ~STORE COURSES FROM NEEDED DEPTS
               (F_R_ALL (F_R_ALL (F_SD (EVAL DEPTS)) NIL) NIL)
           )
            (COND
                [(NULL (MEMB 'AREAS F_WL))      ~REMOVE PARENTHESES
                  (SETQ LIST_HOLDER NIL)        ~SEPARATING DIFFERENT
                  (SETQ S_L                     ~DEPTS IF NOT NEEDED
                      (F_R_ALL (RM_P S_L) NIL))]       ~LATER
                [T
                  'DONT_CHANGE_S_L])]
          [T
            (MODIFY '(NO REQUIRED DEPTS ARE PRESENT FOR) CRL)]
      )]
    [T
      'NO_DEPTS_KEYWORD])
(COND
    [(MEMB 'CRS F_WL)                   ~COURSE HANDLER
      (COND
          [(GE (LENGTH S_L) CRS)        ~CHECK FOR CORRECT NUMBER
            (SETQ R_CRSCT 0)            ~OF COURSES
            (R_CRS_CK S_L CRS)]
          [T
            (COND
                [(NOT (EQUAL (CAR (LAST (CAR (LAST BLK_BD))))
```

```
                    CRL))
                  (MODIFY '(NOT ENOUGH COURSES FOR) CRL)]
               [T
                 'ALREADY_NOTED])
            (COND
               [S_L
                 ((SETQ R_CRSCT 0) (R_CRS_CK S_L
                                     (LENGTH S_L)))]
               [T
                 (SET (VAPPEND CRL) NIL)])])]
    [T
      'NO_CRS_KEYWORD])
(COND
    [(MEMB 'AREAS F_WL)          ~AREAS HANDLER
      (SETQ AREASCT
          (LENGTH (SETQ UNIQ_DEPTS
                        (MAPCAR 'RETURN_UNIQ_DEPTS S_L))))
      (SETQ A_L          ~GET A COURSE FROM EACH UNIQUE AREA
          (MAPCAR '[LAMBDA (CSL)
                           (CAR CSL)] S_L))
      (SETQ LIST_HOLDER NIL)
      (SETQ S_L          ~REMOVE SEPARATING PARENTHESES
          (F_R_ALL (RM_P S_L) NIL))
      (MAPC '[LAMBDA (CAL)      ~REMOVE COURSES ON A_L FROM S_L
                (SETQ S_L
                    (F_RM CAL S_L))] A_L)
      (SETQ S_L           ~S_L CONSISTS OF A_L FOLLOWED BY S_L
          (SETQ A_L
              (APPEND A_L S_L)))
      (COND
          [(GE AREASCT AREAS)  ~TEST NUMBER OF AREAS LOCATED
            'AREASCT_OK]
          [T
            (MODIFY '(NOT ENOUGH AREAS FOR) CRL)])]
    [T
      'NO_AREAS_KEYWORD])
```

```
(COND
    [(MEMB 'LEVEL F_WL)          ~LEVEL HANDLER
      (SETQ LEVEL_HRS_CT 0)
      (SETQ F_LL                          ~FIND AND STORE COURSES
          (SUBSET '[LAMBDA (CDL)          ~OF THE NEEDED LEVEL ALSO
                       (COND              ~COUNT HOURS
                           [(GE (CADR CDL) LEVEL)
                             (SETQ LEVEL_HRS_CT
                                 (+ LEVEL_HRS_CT
                                     (CADDR CDL)))
                             CDL]
                           [T
                             NIL])] D_L))
      (COND
          [(MEMB 'WHERE_HRS F_WL)       ~WHERE_HRS HANDLER
            (COND
                [(GE LEVEL_HRS_CT WHERE_HRS)    ~TEST LEVEL HOURS
                  (SETQ R_HRSCT 0)              ~AMOUNT
                  (R_HRS_CK F_LL WHERE_HRS)
                  (MAPC '[LAMBDA (COURSE)
                             (SETQ D_L
                                 (F_RM COURSE D_L))] RM_CL)
                  (COND
                      [(GE (SETQ LEFT_HRS       ~TEST HOURS AMOUNT
                                 (- SUMHRS WHERE_HRS))
                           (SETQ NEED_HRS
                               (- HRS WHERE_HRS)))
                        (SETQ R_HRSCT 0)
                        (R_HRS_CK D_L NEED_HRS)]
                      [T
                        (MODIFY '(NOT ENOUGH HOURS FOR) CRL)
                        (SETQ R_HRSCT 0)
                        (R_HRS_CK D_L LEFT_HRS)])]
                [T
                  (MODIFY '(NOT ENOUGH WHERE_HRS FOR) CRL)
                  (COND
```

```
                    [(GE LEVEL_HRS_CT 0)
                      (SETQ R_HRSCT 0)
                      (R_HRS_CK F_LL LEVEL_HRS_CT)]
                    [T
                      'NO_WHERE_HRS_TO_REMOVE])])]
          [T
            'NO_WHERE_HRS_KEYWORD
            (COND
                [(GE LEVEL_HRS_CT HRS) ~TEST HOURS AMOUNT
                  (SETQ R_HRSCT 0)
                  (R_HRS_CK F_LL HRS)]
                [T
                  (COND
                      [(NOT (EQUAL (CAR (LAST (CAR (LAST
                                                    BLK_BD))))
                                   CRL))
                        (MODIFY '(NOT ENOUGH HOURS FOR) CRL)]
                      [T
                        'ALREADY_NOTED])
                  (SETQ R_HRSCT 0)
                  (R_HRS_CK F_LL LEVEL_HRS_CT)])])]
    [T
      'NO_LEVEL_KEYWORD])
(COND
    [(AND (MEMB 'HRS F_WL)      ~HOURS HANDLER IF NOT PREVIOUSLY
          (EQUAL D_L NIL))      ~DEALT WITH
      (SETQ SUMHRS 0)
      (SUM_HRS S_L)
      (COND
          [(GE SUMHRS HRS)
            (SETQ R_HRSCT 0)
            (R_HRS_CK S_L HRS)]
          [T
            (COND
                [(NOT (EQUAL (CAR (LAST (CAR (LAST BLK_BD))))
                             CRL))
```

```
                        (MODIFY '(NOT ENOUGH HOURS FOR) CRL)]
                    [T
                      'ALREADY_RECORDED])
                (SETQ R_HRSCT 0)
                (R_HRS_CK S_L SUMHRS)])]
            [T
              'HRS_ALREADY_REMOVED])])

(DEF 'F_D       ~FIND COURSE IN A GIVEN DEPARTMENT
    '[LAMBDA (D)
        (SUBSET '[LAMBDA (CT)
                    (COND
                        [(EQUAL D
                              (CAR CT))
                          CT]
                        [T
                          NIL])] TRANS)])

(DEF 'F_R_CNT_HRS       ~FIND AND REMOVE A COURSE FROM TRANS WHILE COUNTING
    '[LAMBDA (COURSE TRANS)     ~HOURS AS THEY ARE REMOVED
        (COND
            [(EQUAL (CAR TRANS) COURSE)
              (SETQ R_HRSCT
                  (+ R_HRSCT
                      (CAR (LAST COURSE))))
              (CDR TRANS)]
            [T
              (CONS (CAR TRANS)
                  (F_R_CNT_HRS COURSE
                      (CDR TRANS)))])])

(DEF 'F_RM              ~FIND AND REMOVE ANY EXISTING COURSE FROM
    '[LAMBDA (COURSE TRANS)             ~ANY LIST
        (COND
            [(EQUAL (CAR TRANS) COURSE)
              (CDR TRANS)]
```

```
            [T
              (CONS (CAR TRANS)
                  (F_RM COURSE
                      (CDR TRANS)))])])

(DEF 'F_SD                ~FIND COURSES FROM A SET OF DEPARTMENTS
    '[LAMBDA (DL)
        (MAPCAR '[LAMBDA (CDL)
                    (SUBSET '[LAMBDA (CT)
                                (COND
                                    [(EQUAL CDL
                                        (CAR CT))
                                      CT]
                                    [T
                                      NIL])] TRANS)] DL)])

(DEF 'F_R_ALL             ~FIND AND REMOVE ALL OCCURRENCES OF ANY WORD
    '[LAMBDA (LIST WD)           ~FROM ANY LIST
        (SUBSET '[LAMBDA (CL)
                    (COND
                        [(EQUAL CL WD)
                          NIL]
                        [T
                          CL])] LIST)])

(DEF 'RM_P                ~REMOVE EXTRA PARENTHESES FROM A COURSE LIST
    '[LAMBDA (COURSE_LIST)
        (COND
            [(ATOM (CAR COURSE_LIST))
              (SETQ LIST_HOLDER
                  (APPEND LIST_HOLDER
                      (LIST COURSE_LIST)))]
            [T
              (RM_P (CAR COURSE_LIST))
              (RM_P (CDR COURSE_LIST))])])
```

```
(DEF 'MODIFY            ~ATTACH MESSAGES TO THE BLACK BOARD
    '[LAMBDA (MSG IDENT)
        (SETQ BLK_BD
            (APPEND BLK_BD
                (LIST (APPEND MSG
                              (LIST IDENT)))))])

(DEF 'RM_C              ~REMOVE A COURSE FROM THE TRANS
    '[LAMBDA (COURSE)
        (SETQ TRANS
            (F_RM COURSE TRANS))])

(DEF 'R_HRS_CK          ~REMOVE COURSE ON 1ST ARGUMENT LIST FROM TRANS AS
    '[LAMBDA (LIST NUMHRS)      ~DICTATED BY THE 2ND ARGUMENT
        (EVERY '[LAMBDA (CL)
                    (COND
                        [(LT R_HRSCT NUMHRS)
                          (SETQ TRANS
                              (F_R_CNT_HRS CL TRANS))
                          (SETQ RM_CL
                              (APPEND RM_CL
                                  (LIST CL)))]
                        [T
                          'NIL])] LIST)])

(DEF 'R_CRS_CK          ~REMOVE COURSE ON ANY LIST FROM TRANS AS
    '[LAMBDA (LIST NUMCRS)      ~DICTATED BY THE NUMBER OF COURSES
        (EVERY '[LAMBDA (CL)            ~ARGUMENT
                    (COND
                        [(NULL (MEMB 'NONCONSUMABLE F_WL))
                          (COND
                              [(LT R_CRSCT NUMCRS)
                                (SETQ TRANS
                                    (F_RM CL TRANS))
                                (SETQ R_CRSCT
                                    (ADD1 R_CRSCT))]
```

```
                                    [T
                                      NIL])]
                                [T
                                  (SET (VAPPEND CRL) LIST)
                                  NIL])] LIST)])

(DEF 'RETURN_UNIQ_DEPTS          ~RETURNS NAMES OF UNIQUE DEPARTMENTS
    '[LAMBDA (CSL)
        (CAAR CSL)])

(DEF 'RM_DC              ~REMOVES DUPLICATE COURSES ON FOUND COURSE LIST
    '[LAMBDA (CL)
        (SETQ DUPCTR 0)
        (SUBSET '[LAMBDA (CT)
                    (COND
                        [(EQUAL CL CT)
                          (SETQ DUPCTR
                              (ADD1 DUPCTR))
                          (COND
                              [(GT DUPCTR 1)
                                (SETQ F_CL
                                    (F_R_CNT_HRS CL F_CL))]
                              [T
                                NIL])]
                        [T
                          NIL])] F_CL)])

(DEF 'R_XHRS             ~REMOVES EXTRA HOURS FROM ANY LIST
    '[LAMBDA (CDL)
        (COND


            [(LE HRS
                 (- SUMHRS
                     (CAR (LAST CDL))))
              (SET LIST_NAM
```

```
                    (F_RM CDL
                        (EVAL LIST_NAM)))
                  (SETQ SUMHRS
                      (- SUMHRS
                          (CAR (LAST CDL))))]
              [T
                'NOTHING_TO_REMOVE])])

(DEF 'X_LOGIC_RULES     ~EXECUTES LOGIC RULES AND RECORDS RESULTS
    '[NLAMBDA (CLL)
         (SETQ LIST_HOLDER NIL)
         (SETQ F_CL NIL)
         (COND
             [(SETQ C_L
                  (F_R_ALL (RM_P (EVAL_LR (EVAL CLL))) NIL))
               (MAPC 'RM_C C_L)
               (MODIFY '(ALL REQUIREMENTS MET FOR) CLL)]
             [T
               (MODIFY '(REQUIREMENTS NOT MET FOR) CLL)
               (SETQ F_CL
                   (F_R_ALL (RM_P F_CL) NIL))
               (MAPC 'RM_C F_CL)])
         (RECORD_CRS_USD CLL)])

(DEF 'EVAL_LR           ~EVALUATES LOGIC RULES
    '[LAMBDA (L_RULE)
         (COND
             [(ATOM (CAR L_RULE))
               (L_R L_RULE)]
             [(AND (ATOM (CAAR L_RULE))
                   (NULL (CDR L_RULE)))
               (L_R L_RULE)]
             [(MEMB (CADR L_RULE)
                   '(OR AND))
               (APPLY (VAPPEND (CADR L_RULE))
                   (LIST (EVAL_LR (CADDR L_RULE))
```

```
                    (EVAL_LR (CAR L_RULE))))]
              [T
                (LIST (APPLY (VAPPEND (CADAR L_RULE))
                             (LIST (CAR (EVAL_LR (CDDAR L_RULE)))
                                   (CAR (EVAL_LR (LIST (CAAR L_RULE))))))))]]
)

(DEF 'L_R       ~LOGIC RULE RECOGNIZER
    '[LAMBDA (LIST)
         (SETQ D_L NIL)
         (SETQ F_WL
             (F_KEYWDS LIST))
         (COND
             [(MEMB 'DEPT F_WL)
               (COND
                   [(SETQ D_L
                        (F_D DEPT))
                     (COND
                         [(MEMB 'HRS F_WL)
                           (SETQ SUMHRS 0)
                           (SUM_HRS D_L)
                           (COND
                               [(GT SUMHRS HRS)
                                 (SETQ LIST_NAM
                                     'D_L)
                                 (SUBSET 'R_XHRS D_L)
                                 D_L]
                               [(EQUAL SUMHRS HRS)
                                 D_L]
                               [T
                                 NIL])]
                         [T
                           'NO_HRS])]
                   [T
                     NIL])]
              [T
```

```
                (COND
                    [(MEMB 'COURSES F_WL)
                      (COND
                          [(SETQ S_L
                                (EVAL_NO_NILS (EVAL COURSES)))
                            S_L]
                          [T
                            NIL])]
                    [T
                      'NO_COURSES_KEYWORD])
                (COND
                    [(MEMB 'HRS F_WL)
                      (SETQ SUMHRS 0)
                      (SUM_HRS S_L)
                      (COND
                          [(GT SUMHRS HRS)
                            (SETQ LIST_NAM
                                'S_L)
                            (SUBSET 'R_XHRS S_L)
                            S_L]
                          [(EQUAL SUMHRS HRS)
                            S_L]
                          [T
                            NIL])]
                    [T
                      'NO_HRS_KEYWORD])])])

(DEF 'X_FE      ~EXECUTE FREE ELECTIVE CHECK
    '[LAMBDA ()
         (SETQ THRSRM
             (- TCNTHRS HRSLEFT))
         (EVERY '[LAMBDA (CL)
                    (COND
                        [(NUMBERP CL)
                          (SETQ TOT_HRS_REQUIRED CL)
                          NIL]
```

```
                    [T
                      T])] FREE_ELECTIVE_REQUIREMENT)
          (COND
              [(LT TCNTHRS TOT_HRS_REQUIRED)     ~RECORD RESULTS
                (SETQ BLK_BD
                    (APPEND BLK_BD
                        (LIST (APPEND (LIST 'NOT
                                            'ENOUGH
                                            'HOURS
                                            'FOR
                                            'FREE_ELECTIVE_REQUIREMENT
                                            'SINCE)
                                      (LIST (- TOT_HRS_REQUIRED TCNTHRS)
                                          'MORE
                                          'HOURS
                                          'ARE
                                          'NEEDED)))))
                (COND
                    [TRANS
                      (SETQ R_HRSCT 0)
                      (R_HRS_CK TRANS HRSLEFT)]
                    [T
                      NIL])]
               [T
                 (SETQ R_HRSCT 0)
                 (R_HRS_CK TRANS
                     (- TOT_HRS_REQUIRED THRSRM))
                 (MODIFY '(THESE WERE EXTRA COURSES) TRANS)])
          (RECORD_CRS_USD 'FREE_ELECTIVE_REQUIREMENT)
          (CREATE_HISTORY)])

(DEF 'WHAT_FILLED       ~FINDS COURSES USED TO FULFILL REQUIREMENT
    '[NLAMBDA (RULE_NAME)
        (EVAL (VAPPEND RULE_NAME))])

(DEF 'WHATS_NOT_FILLED  ~DISPLAYS UNFULFILLED REQUIREMENTS
```

```
    '[LAMBDA ()
         (MAPC 'PRINT
             (SUBSET '[LAMBDA (CBB)
                          (SETQ REPORT
                              (COND
                                  [(SOME '[LAMBDA (CL)
                                               (COND
                                                   [(MEMB CL CBB)
                                                     CBB]
                                                   [T
                                                     NIL])]
                                      '(NO NOT MISSING))
                                    CBB]
                                  [T
                                    NIL])))] BLK_BD))])

(DEF 'CREATE_HISTORY    ~CREATES HISTORY LIST
    '[LAMBDA ()
         (SETQ HISTORY
             (LIST (LIST 'NON_COUNTABLE VNON_COUNTABLE)))
         (MAPC 'BLD_HIST PRE_SEL)
         (MAPC 'BLD_HIST LOGIC_LIST)
         (MAPC 'BLD_HIST
             '(LR_LIT_SPEECH))
         (MAPC 'BLD_HIST RULE_LIST)
         (MAPC 'BLD_HIST
             '(FREE_ELECTIVE_REQUIREMENT))])

(DEF 'BLD_HIST          ~BUILDS HISTORY LIST
    '[NLAMBDA (CL)
         (SETQ HISTORY
             (APPEND HISTORY
                 (LIST (LIST CL
                             (EVAL (VAPPEND CL))))))])

(DEF 'WHY       ~ALLOWS QUESTIONING OF THE SYSTEM AS TO WHY A
```

```
    '[NLAMBDA (RULE)    ~PARTICULAR REQUIREMENT WAS NOT FULFILLED
        (MAPC 'WRITER
            (EVAL_WHY RULE))])

(DEF 'WRITER    ~FORMATS SYSTEM REPORT BY WRITING SPACES
    '[LAMBDA (CL)       ~TO SEPARATE OUTPUT ITEMS
        (APPEND (LIST (SPACES 1))
            (PRINC CL))])

(DEF 'EVAL_WHY          ~ASSISTS IN THE EXPLANATION FACILITY
    '[LAMBDA (RULE)
        (MAPC 'WRITER
            (APPEND (APPEND '(BECAUSE)
                        (APPEND (LIST RULE
                                    'REQUIRES)))))
        (MAPC 'WRITER                  ~OUTPUTS THE FORMULATED
            (BLD_EXPL (EVAL RULE)))        -EXPLANATION
        (TERPRI)])

(DEF 'BLD_EXPL          ~BUILDS EXPLANATION OF WHY A SPECIFIC REQUIREMENT
    '[LAMBDA (RULE)             ~WAS OR WAS NOT MET
        (COND
            [(NOT (ATOM (CAR RULE)))
              (COND
                  [(OR (MEMB 'AND RULE)
                       (MEMB 'OR RULE))
                    (COND
                        [(OR (NOT (ATOM (CAAR RULE)))
                             (NOT (NULL (F_KEYWDS (CAAR RULE)))))
                          (COND
                              [(SETQ F_WL      ~THE REQUIREMENT IS A LOGIC
                                   (F_KEYWDS (CAAR RULE)))     ~RULE
                                (SETQ EXPL NIL)
                                (WHY_LR RULE)]
                              [T                ~THE REQUIREMENT IS OF THE
                                (MAPC 'PRINT RULE)])]  ~LOGIC CLASSIFICATION
```

```
                        [T
                          (SETQ LIST_HLDR NIL)
                          (MAPC '[LAMBDA (CR)
                                    (SETQ LIST_HLDR
                                        (APPEND LIST_HLDR
                                            (LIST CR)))] RULE)
                          LIST_HLDR])]
                    [T                          ~THE REQUIREMENT WAS OF THE
                      (MAPC 'WRITER RULE)])]  ~PRE-SELECTED CLASSIFICATION
               [T
                 (WHY_RULES RULE)])])     ~THE REQUIREMENT WAS OF THE RULE
                                           ~CLASSIFICATION

(DEF 'WHY_LR    ~DETERMINE EXPLANATION FOR A LOGIC
    '[LAMBDA (CR)            ~RULE REQUIREMENT
        (COND
            [(OR (EQUAL CR
                        'OR)
                 (EQUAL CR
                        'AND))
              (SETQ EXPL
                  (APPEND EXPL
                      (LIST CR)))
              (WHY_LR (CDR CR))]
            [(MEMB (CAR CR)
                   '(OR AND))
              (SETQ EXPL
                  (APPEND EXPL
                      (LIST (CAR CR))))
              (WHY_LR (CDR CR))]
            [(ATOM (CAR CR))
              (SETQ EXPL
                  (APPEND EXPL
                      (LIST (WHY_RULES CR))))]
            [T
              (WHY_LR (CAR CR))
```

```
            (WHY_LR (CDR CR))])
        (SETQ EXPL
            (EDIT_EXPR EXPL))])

(DEF 'EDIT_EXPR          ~REMOVE ALL NILS FROM THE EXPLANATION
    '[LAMBDA (RULE)
        (COND
            [(MEMB 'NIL RULE)
              (SETQ RULE
                  (F_R_ALL RULE NIL))]
            [T
              'NOTHING_TO_DO])
        RULE])

(DEF 'WHY_RULES          ~CREATES EXPLANATION FOR RULE REQUIREMENTS
    '[LAMBDA (RULE)
        (SETQ LIST_HLDR NIL)
        (SETQ F_WL
            (F_KEYWDS RULE))
        (MAPC '[LAMBDA (CL)
                  (COND
                      [(AND (MEMB CL F_WL)
                            (NOT (MEMB CL
                                       '(LEVEL NONCONSUMABLE))))
                        (SETQ F_WRD_LST
                            (MEMB CL RULE))
                        (COND
                            [(EQUAL CL
                                  'WHERE_HRS)
                              (SETQ LIST_HLDR
                                  (APPEND LIST_HLDR
                                      (LIST 'WHERE
                                          (CADR F_WRD_LST)
                                          'HRS)))]
                            [T
                              (SETQ LIST_HLDR
```

```
                                    (APPEND LIST_HLDR
                                        (REVERSE (LIST (CAR F_WRD_LST)
                                                       (CADR F_WRD_LST)))
                                    ))])]
                            [T
                              (COND
                                  [(NOT (MEMB CL LIST_HLDR))
                                    (SETQ LIST_HLDR
                                        (APPEND LIST_HLDR
                                            (LIST CL)))]
                                  [T
                                    'DONT_ADD_TO_LIST])])] RULE)
        LIST_HLDR])

(DEF 'SHOW_NAMES        ~PRODUCES SYSTEM NAMES OF ALL THE REQUIREMENTS
    '[LAMBDA ()
         (MAPC 'PP_WRITER PRE_SEL)
         (TERPRI)
         (MAPC 'PP_WRITER LOGIC_LIST)
         (TERPRI)
         (MAPC 'PP_WRITER
             '(LR_LIT_SPEECH))
         (TERPRI)
         (MAPC 'PP_WRITER RULE_LIST)
         (TERPRI)
         (MAPC 'PP_WRITER
             '(FREE_ELECTIVE_REQUIREMENT))
         (TERPRI)])

(DEF 'PP_WRITER         ~PRINTS EACH REQUIREMENT NAME ON A SEPARATE LINE
    '[LAMBDA (CL)
         (APPEND (LIST (SPACES 1))
             (PRINTC CL))])

(DEF 'BLD_TRANS         ~BUILDS OR CREATES A NEW TRANSCRIPT
    '[LAMBDA ()
```

```
        (SETQ TRANS
            (LIST (READ)))
        (COND
            [(SETQ VAR
                 (READLINE))
              (SETQ TRANS
                  (APPEND TRANS VAR))]
            [T
              'THATISALL])])

(DEF 'ADD_TO_TRANS      ~ADDS ANY NUMBER OF COURSES
    '[LAMBDA ()                  ~TO THE TRANSCRIPT
        (COND
            [(NOT (ATOM (SETQ VAR
                              (READ))))
              (SETQ VAR
                  (APPEND (LIST VAR)
                      (READLINE)))
              (CK_DUP_ADDS VAR)]
            [T
              (MAPC 'PRINC
                  '(INVALID\ COURSE\ ENTERED))
              (TERPRI)])])

(DEF 'CK_DUP_ADDS       ~CHECKS FOR AN ATTEMPT TO ADD A
    '[LAMBDA (VAR)               ~DUPLICATE COURSE TO THE
        (MAPC '[LAMBDA (CV)            ~TRANSCRIPT
                  (SETQ DUPCTR 0)
                  (SUBSET '[LAMBDA (CT)
                              (COND
                                  [(EQUAL CV CT)
                                    (SETQ DUPCTR
                                        (ADD1 DUPCTR))
                                    (MAPC 'PRINC
                                        (APPEND (LIST CV)
                                            '(
```

```
                                                    \ IS\ ALREADY\ ON\ TRANS~
CRIPT)))
                                              (TERPRI)
                                              (COND
                                                  [(GT DUPCTR 0)
                                                    (SETQ VAR
                                                        (F_RM CV VAR))]
                                                  [T
                                                    NIL])]
                                             [T
                                               NIL])] TRANS)] VAR)
          (SETQ TRANS
              (APPEND TRANS VAR))])

(DEF 'DELETE_FROM_TRANS          ~DELETES COURSES FROM THE TRANSCRIPT
    '[LAMBDA ()
          (SETQ DEL_LIST
              (LIST (READ)))
          (COND
              [(SETQ DEL_LIST
                   (APPEND DEL_LIST
                       (READLINE)))
                ]
              [T
                'THATSALL])
          (COND
              [(SETQ F_CL
                   (EVAL_NO_NILS DEL_LIST))
                (MAPC 'RM_C DEL_LIST)]
              [T
                (MAPC 'PRINC
                    '(COURSE\ IS\ NOT\ PRESENT\ ON\ TRANSCRIPT))])
          (TERPRI)
          TRANS])

(SETQ TR1
```

```
    '((CMPSC 1 1) (MA&ST 4 3) (MA&ST 6 2) (CHEM 1 4) (ENGTC 10 3) (HIST
176 3) (MA&ST 8 5) (CMPSC 260 3) (MTENG 1 1) (CHEM 3 3) (CMPSC 73 2) (EN~
GMGT 130 3) (MA&ST 21 5) (MILSC 10 1) (PHYSCS 23 4) (ENGLSH 1 3) (HIST 1~
12 3) (CMPSC 74 3) (ECONOM 110 3) (MILSC 30 1) (PSYCH 50 3) (PHYED 103 1
) (PHYSCS 25 4) (PHYSCS 26 1) (CMPSC 83 3) (CMPSC 163 3) (ENGLSH 106 3)
(MA&ST 204 3) (MILSC 20 1) (MILSC 40 1) (PHIL 15 3) (CMPSC 183 3) (CMPSC
 218 3) (ELENG 61 3) (SP&MS 85 3) (CMPSC 168 3) (CMPSC 349 3) (CMPSC 361
 3) (MA&ST 343 3) (SP&MS 283 3) (CMPSC 268 3) (CMPSC 293 3)))

(SETQ TR2
    '((PHYSCS 22 1) (CHEM 1 4) (CHEM 2 1) (ENGTC 10 3) (HIST 176 3) (MA&~
ST 8 5) (MTENG 1 1) (CHEM 3 3) (CMPSC 73 2) (MA&ST 21 5) (PHYSCS 21 4) (
ENGLSH 1 3) (HIST 112 3) (CMPSC 74 3) (ECONOM 110 3) (MA&ST 22 4) (PHYED
 103 1) (PHYSCS 25 4) (PHYSCS 26 1) (CMPSC 83 3) (CMPSC 163 3) (ENGLSH 1~
06 3) (MA&ST 204 3) (PHIL 15 3) (CMPSC 183 3) (CMPSC 218 3) (CMPSC 253 3
) (ELENG 61 3) (SP&MS 85 3) (ENGLSH 60 3) (MA&ST 208 3) (CMPSC 168 3) (C~
MPSC 349 3) (ELENG 211 3) (MA&ST 215 3) (SP&MS 283 3)))

(SETQ TR3
    '((CHEM 1 4) (ENGLSH 110 3) (CHEM 2 1) (ENGTC 10 3) (ECONOM 111 3) (
MA&ST 8 5) (MTENG 1 1) (CHEM 3 3) (CMPSC 73 2) (MA&ST 21 5) (ECONOM 215
3) (PHYSCS 23 4) (ENGLSH 1 3) (CMPSC 74 3) (ECONOM 110 3) (MA&ST 22 4) (
PHYED 103 1) (PHYSCS 24 4) (CMPSC 83 3) (CMPSC 163 3) (ENGLSH 106 3) (CM~
PSC 183 3) (CMPSC 218 3) (CMPSC 253 3) (ELENG 61 3) (SP&MS 85 3) (CMPSC
101 3) (ENGLSH 60 3) (MA&ST 208 3) (CMPSC 168 3) (CMPSC 349 3) (CMPSC 36~
1 3) (ELENG 211 3) (MA&ST 215 3) (SP&MS 283 3) (CMPSC 264 3) (CMPSC 268
3)))

(SETQ TR4
    '((CMPSC 1 1) (CHEM 1 4) (ENGLSH 110 3) (CHEM 2 1) (ENGTC 10 3) (CMP~
SC 260 3) (ECONOM 111 3) (MA&ST 8 5) (PSYCH 50 3) (POLYSCI 90 3) (CHEM 3
 3) (CMPSC 73 2) (MA&ST 21 5) (PHIL 10 3) (ECONOM 215 3) (PHYSCS 23 4) (
ENGLSH 1 3) (CMPSC 74 3) (ECONOM 110 3) (MA&ST 209 3) (MA&ST 22 4) (PHYE~
D 103 1) (PHYSCS 24 4) (MA&ST 115 3) (CMPSC 83 3) (CMPSC 163 3) (ENGLSH
106 3) (CMPSC 183 3) (CMPSC 218 3) (CMPSC 253 3) (ELENG 61 3) (SP&MS 85
3) (CMPSC 101 3) (ENGLSH 60 3) (MA&ST 208 3) (CMPSC 168 3) (CMPSC 349 3)
```

```
 (CMPSC 361 3) (ELENG 211 3) (MA&ST 215 3) (SP&MS 283 3) (CMPSC 264 3) (
CMPSC 268 3)))

(SETQ TR5
    '((SOCIOL 81 3) (CMPSC 1 1) (CHEM 1 4) (ENGLSH 110 3) (CHEM 2 1) (EN~
GTC 10 3) (CMPSC 260 3) (ECONOM 111 3) (MA&ST 8 5) (PSYCH 52 3) (PSYCH 5~
0 3) (POLYSCI 90 3) (CHEM 3 3) (CMPSC 73 2) (MA&ST 21 5) (PHIL 10 3) (EC~
ONOM 215 3) (PHYSCS 23 4) (ENGLSH 1 3) (CMPSC 74 3) (ECONOM 110 3) (MA&S~
T 209 3) (MA&ST 22 4) (PHYED 103 1) (PHYSCS 24 4) (MA&ST 115 3) (CMPSC 8~
3 3) (CMPSC 163 3) (ENGLSH 106 3) (CMPSC 183 3) (CMPSC 218 3) (CMPSC 253
 3) (ELENG 61 3) (SP&MS 85 3) (CMPSC 101 3) (ENGLSH 60 3) (MA&ST 208 3)
(CMPSC 168 3) (CMPSC 349 3) (CMPSC 361 3) (ELENG 211 3) (MA&ST 215 3) (S~
P&MS 283 3) (CMPSC 264 3) (CMPSC 268 3)))

(SETQ ROLLA_TRANS
    '((MA&ST 4 3) (MA&ST 6 2) (CHEM 1 4) (CHEM 2 1) (ENGTC 10 3) (HIST 1~
76 3) (MA&ST 8 5) (MTENG 1 1) (CHEM 3 3) (CMPSC 73 2) (MA&ST 21 5) (MILS~
C 10 1) (PHYSCS 21 4) (ENGLSH 1 3) (HIST 112 3) (CMPSC 74 3) (ECONOM 110
 3) (MA&ST 22 4) (MILSC 30 1) (PHYED 103 1) (PHYSCS 25 4) (PHYSCS 26 1)
(CMPSC 83 3) (CMPSC 163 3) (ENGLSH 106 3) (MA&ST 204 3) (MILSC 20 1) (MI~
LSC 40 1) (PHIL 15 3) (CMPSC 183 3) (CMPSC 218 3) (CMPSC 253 3) (ELENG 6~
1 3) (SP&MS 85 3) (CMPSC 0 0) (ENGLSH 60 3) (MA&ST 208 3) (CMPSC 168 3)
(CMPSC 349 3) (CMPSC 361 3) (ELENG 211 3) (MA&ST 215 3) (SP&MS 283 3) (C~
MPSC 0 0) (CMPSC 423 4)))

(SETQ NON_COUNTABLE
    '((MA&ST 1 1) (MA&ST 2 5) (MA&ST 4 3) (MA&ST 6 2) (MILSC 10 1) (MILS~
C 20 1) (MILSC 30 1) (MILSC 40 1)))

(SETQ PRE_SEL
    '(CSOR SCI_PROG JCL MCHLANG BLK_STR_LANG COBOL ASSMBLR NUM_ANALYSIS
DATA_STRUC OPER_RES EE MATH ENGL))

(SETQ CSOR
    '((CMPSC 1 1)))
```

```
(SETQ SCI_PROG
    '((CMPSC 73 2)))

(SETQ JCL
    '((CMPSC 74 3)))

(SETQ MCHLANG
    '((CMPSC 83 3)))

(SETQ BLK_STR_LANG
    '((CMPSC 163 3)))

(SETQ COBOL
    '((CMPSC 168 3)))

(SETQ ASSMBLR
    '((CMPSC 183 3)))

(SETQ NUM_ANALYSIS
    '((CMPSC 218 3)))

(SETQ DATA_STRUC
    '((CMPSC 253 3)))

(SETQ OPER_RES
    '((CMPSC 260 3)))

(SETQ EE
    '((ELENG 61 3) (ELENG 211 3)))

(SETQ MATH
    '((MA&ST 8 5) (MA&ST 21 5) (MA&ST 22 4)))

(SETQ ENGL
    '((ENGLSH 1 3) (ENGLSH 60 3)))
```

```
(SETQ LOGIC_LIST
    '(L_STAT L_ALG L_LABSCI L_PHYS))

(SETQ L_STAT
    '((MA&ST 215 3) OR (MA&ST 343 3)))

(SETQ L_ALG
    '((MA&ST 203 3) OR (MA&ST 208 3)))

(SETQ L_LABSCI
    '(((CHEM 1 4) AND (CHEM 2 1)) OR ((LIFSCI 1 3) AND (LIFSCI 2 2))))

(SETQ L_PHYS
    '((((PHYSCS 21 4) AND (PHYSCS 22 1)) OR (PHYSCS 23 4)) AND (((PHYSCS
 25 4) AND (PHYSCS 26 1)) OR (PHYSCS 24 4))))

(SETQ LR_LIT_SPEECH
    '(((HRS 3 FROM COURSES LIT) AND (HRS 3 FROM DEPT SP&MS)) OR (HRS 6 F~
ROM COURSES LIT)))

(SETQ LIT
    '((ENGLSH 75 3) (ENGLSH 80 3) (ENGLSH 102 3) (ENGLSH 105 3) (ENGLSH
106 3) (ENGLSH 110 3) (ENGLSH 130 3) (ENGLSH 133 3) (ENGLSH 144 3) (ENGL~
SH 225 3) (ENGLSH 235 3) (ENGLSH 312 3) (ENGLSH 315 3) (ENGLSH 330) (ENG~
LSH 331 3) (ENGLSH 335 3) (ENGLSH 336 3) (ENGLSH 345 3) (ENGLSH 353 3) (
ENGLSH 355 3) (ENGLSH 356 3) (ENGLSH 361 3) (ENGLSH 362 3) (ENGLSH 370 3
) (ENGLSH 371 3) (ENGLSH 372 3) (ENGLSH 375 3) (ENGLSH 376 3) (ENGLSH 37~
8 3) (ENGLSH 380 3) (FRENCH 80 4) (FRENCH 90 3) (FRENCH 170 3) (FRENCH 3~
70 3) (FRENCH 375) (GERMAN 70 2) (GERMAN 90 3) (GERMAN 170 3) (GERMAN 37~
0 3) (GERMAN 375 3) (GERMAN 385 3) (RUSSIAN 90 3) (RUSSIAN 170 3) (RUSSI~
AN 370 3) (RUSSIAN 375 3) (SPANISH 170 3) (SPANISH 277 3) (SPANISH 370 3
) (SPANISH 371 3) (SPANISH 377 3) (SPANISH 378 3) (SPANISH 379 3)))

(SETQ RULE_LIST
    '(R_CONST R_PHIL R_MATH R_SOCSCI R_SCIENG R_CSC))
```

```
(SETQ R_PHIL
    '(HRS 3 FROM THE DEPT PHIL))

(SETQ R_CONST
    '(CRS 1 FROM THE COURSES CONST A NONCONSUMABLE REQUIREMENT))

(SETQ CONST
    '((HIST 112 3) (HIST 175 3) (HIST 176 3) (POLYSCI 90 3)))

(SETQ R_MATH
    '(HRS 3 OF LEVEL 23 OR GREATER FROM THE DEPT MA&ST))

(SETQ R_SOCSCI
    '(HRS 9 FROM AREAS 2 OF THE DEPTS SOC_SCI))

(SETQ R_SCIENG
    '(HRS 9 FROM THE DEPTS BS_NOT_CS))

(SETQ BS_NOT_CS
    '(AROENG CIVENG CHEM CHEMENG CERENG ECONOM ELENG ENGMGT ENGMCH GEOLE~
NG GEOPHYSCS LIFSCI MA&ST MECHENG MTENG MINENG PETENG PHYSCS PSYCH))

(SETQ R_CSC
    '(HRS 12 WHERE_HRS 6 ARE OF LEVEL 300 OR GREATER FROM THE DEPT CMPSC
))

(SETQ SOC_SCI
    '(ECONOM HIST POLYSCI PSYCH SOCIOL))

(SETQ FREE_ELECTIVE_REQUIREMENT
    '(AT LEAST 130 COUNTABLE HOURS))

(SETQ TRANS
    '((SP&MS 283 3) (CMPSC 268 3)))

(SETQ BLK_BD
```

```
    '((ALL REQUIREMENTS MET FOR CSOR) (ALL REQUIREMENTS MET FOR SCI_PROG
) (ALL REQUIREMENTS MET FOR JCL) (ALL REQUIREMENTS MET FOR MCHLANG) (ALL
 REQUIREMENTS MET FOR BLK_STR_LANG) (ALL REQUIREMENTS MET FOR COBOL) (AL~
L REQUIREMENTS MET FOR ASSMBLR) (ALL REQUIREMENTS MET FOR NUM_ANALYSIS)
(ALL REQUIREMENTS MET FOR DATA_STRUC) (ALL REQUIREMENTS MET FOR OPER_RES
) (ALL REQUIREMENTS MET FOR EE) (ALL REQUIREMENTS MET FOR MATH) (ALL REQ~
UIREMENTS MET FOR ENGL) (ALL REQUIREMENTS MET FOR L_STAT) (ALL REQUIREME~
NTS MET FOR L_ALG) (ALL REQUIREMENTS MET FOR L_LABSCI) (ALL REQUIREMENTS
 MET FOR L_PHYS) (ALL REQUIREMENTS MET FOR LR_LIT_SPEECH) (THESE WERE EX~
TRA COURSES ((SP&MS 283 3) (CMPSC 268 3)))))

(SETQ HISTORY
    '((NON_COUNTABLE NIL) (CSOR ((CMPSC 1 1))) (SCI_PROG ((CMPSC 73 2)))
 (JCL ((CMPSC 74 3))) (MCHLANG ((CMPSC 83 3))) (BLK_STR_LANG ((CMPSC 163
 3))) (COBOL ((CMPSC 168 3))) (ASSMBLR ((CMPSC 183 3))) (NUM_ANALYSIS ((
CMPSC 218 3))) (DATA_STRUC ((CMPSC 253 3))) (OPER_RES ((CMPSC 260 3))) (
EE ((ELENG 61 3) (ELENG 211 3))) (MATH ((MA&ST 8 5) (MA&ST 21 5) (MA&ST
22 4))) (ENGL ((ENGLSH 1 3) (ENGLSH 60 3))) (L_STAT ((MA&ST 215 3))) (L_~
ALG ((MA&ST 208 3))) (L_LABSCI ((CHEM 1 4) (CHEM 2 1))) (L_PHYS ((PHYSCS
 23 4) (PHYSCS 24 4))) (LR_LIT_SPEECH ((ENGLSH 110 3) (ENGLSH 106 3))) (
R_CONST ((POLYSCI 90 3))) (R_PHIL ((PHIL 10 3))) (R_MATH ((MA&ST 209 3))
) (R_SOCSCI ((ECONOM 111 3) (PSYCH 52 3) (POLYSCI 90 3))) (R_SCIENG ((CH~
EM 3 3) (ECONOM 215 3) (ECONOM 110 3))) (R_CSC ((CMPSC 101 3) (CMPSC 349
 3) (CMPSC 361 3) (CMPSC 264 3))) (FREE_ELECTIVE_REQUIREMENT ((SOCIOL 81
 3) (ENGTC 10 3) (PSYCH 50 3) (PHYED 103 1) (MA&ST 115 3) (SP&MS 85 3)))
))
```