MEDICAL EXPERT SYSTEM

A DISSERTATION

submitted in partial fulfilment of the requirement for the award of the Degree of

MASTER OF TECHNOLOGY

IN

COMPUTER SCIENCE AND TECHNOLOGY

SHAILESH KUMAR TIWARI

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY

JAWAHARLAL NEHRU UNIVERSITY

NEW DELHI -110067

CEERTIFICATE

It is certified that the contents of this dissertation which carries the title MEDICAL EXPERT SYSTEM has been submitted by SHAILESH KUMAR TIWARI , has not been previously submitted for any other degree of this or any other university.

Prof. N.P. MUKHERJEE

(DEAN)

SCSS/JNU

DR. P.C. SAXENA

(Supervisor)

SCSS/JNU

SHAILESH KUMAR TIWARI

(Student)

ACKNOWLEDGEMENT

I am highly indebted to my supervisor Dr. P.C. Saxena,Associate Professor School of Computer and System Sciences, Jwaharlal Nehru University,for his eminent guidance,constant supervision and overwhelming encouragement throughout my dissertation work.

I am grateful to Prof. N.P. Mukherjee dean SCSS J N U for allowing me to do the project.

I am also thankful to all the teaching and non-teaching staff in the SCSS J N U, Department of Computer Science & Engg. I.I.T. Delhi, Eye section of AIIMS, Library Staff of J N U, I.I.T.D. and AIIMS for providing me help.

PREFACE

According to a survey, the people in India have lowest medical attention in the world. A very large number of people are affected with eyesight problems. As a result use of glasses and contact lenses have been increasing rapidly. though contact lenses have not still become very popular, still people are preferring to use contact lenses.

For patient who are having problems with their lenses, there may be several causes and several approaches towards treatment. This expert system has basically four diagnostic outcomes. It may recommend total refit. This is the most expensive alternative and is recommended only when all other possibilities are exhausted. It may recommend change of lens material or use of different cosmetics if it is found that patient is allergic to these. A charge of lifestyle may also be recommended of workplace is causing problems. This expert system has the mechanism to explain why it has asked certain questions. it also explain how it has reached to certain conclusion. Since the distribution of medical experts in India is very uneven, and also there is shortage of medical experts, then system can be used on a IBM compatible PC at any place to provide the expert advice to the people.

CONTENTS

CHAPTER -1

ARTIFICIAL INTELLIGENCE

1. Artificial intelligence concerns with research and theory developments relating human thinking, reasoning and programs.

Expert systems, knowledge bases and knowledge processes are practical AI research results. The theories and programs are called artificial intelligence because they are intended to be electronic imitations of outcomes of human mental activity. Artificial intelligence is significant addition to analytical decision making and problem solving software. The application of AI concepts range from using computer to recognize human voice, creating programs to translate text from one language to other developing techniques that will allow robots to identify objects and reasons about the consequences of various actions and developing computer programs that will reason like human experts.

Accordingly we find that all AI applications fall into five different categories.

1. Natural language
2. Robotics

3. Improved human interfaces

4. Exploratory programming

5. Expert systems

1. Natural language

A natural language is any language that humans can speak. Some AI researchers are trying to develop computer hardware and software that will allow computers to interact with people in a natural language . At the moment commercial activity that involves concepts and techniques derived from natural language research interface is focused on developing natural language interfaces to data bases.

2. Robotics

Creating Robotic devices is hardly the exclusive concern of AI researchers. AI is concerned with only that subset of robotic devices guided by computer programs that allow the devices to analyse and solve the problems that they encounter. Such "intelligent robots" use AI technique to see and manipulate the object they interact with . In essence, the robot can act "intelligent" be cause it has a model of the world stored in a computer that allows it to identify things and analyse how those things will change in response to the various actions the

robot might initiate.

3. Improved Human Interfaces

A third area in which AI concepts and techniques are being actively employed involves the design and development of better interfaces. By applying psychological and programming techniques originally developed in AI labs, computers with interfaces like this found in machinfosh has been developed. The same hardware and software techniques that make macintosh easy to use are already on newer computers. Other improved interface techniques will become available as the underlying AI techniques becomes more widely understood.

4. Exploratory Programming

Exploratory programming refers to the application of AI concepts and techniques to developing large scale applications. The same techniques that allow AI programmers to quickly develop large scale applications, including new programming languages and programming environments, modularity and incremental development can all be used to increase conventional programmer's productivity.

Automatic programming uses AI techniques to all computer programmers to develop computer programs specifying the goals of a program and leaving it upto automatic programming system to generate most of t specific code for the application.

5. Expert Systems

Of all the commercial activities resulting fr AI research expert systems have received most attentio These problem solving systems were initially called expe systems to suggest that they functioned as effective as human experts at their highly specialized tasks.

An expert system is a program that manifests so combinations of concepts, procedures and techniqu derived from AI research. These techniques allow peop to design and develop computer systems that u knowledge and inferences techniques to solve problems.

CHAPTER -2

EXPERT SYSTEM

2.1 Introduction

The area of expert system inves and techniques for constructing man-mach specialized problem-solving expertis consists of knowledge about a particular and skill at solving these problems. Kn speciality is of two sorts: public and knowledge consists of the published defini theories of which textbook and references study are typically composed. But exp more than just this public knowledge. generally possess private knowledge. largely of rules of thumb that have com heuristics. Reasons for emphasising on than formal reasoning are many. Firs difficult and interesting problems do no algorithmic solutions. The second reason experts achieve outstanding performance knowledgeable. The third reason is tha scarce resource whose refinement and repr wealth.

Various definitions of expert systems can be found in literature. Some of them are

(i) Feigenbaum States : An expert system is an intelligent computer program that uses knowledge and inference - procedures to solve problems that are difficult enough and require significant human expertise for their solution. The knowledge necessary to perform at such a level, plus the inference procedures used can be thought of as a model of the expertise of the best practitioners of the field.

(ii) Buchnan defines expert system as a reasoning program that can be distinguished by other AI programs in its utility, performance and transparency.

(iii) The British Computer Society's Committee of the specialist Group in Expert Systems has defended expert systems as "The embodiment within a computer of a knowledge based component from an expert skill in such a way that machine can offer intelligent advice or take an intelligent characteristics, which many would regard as fundamental in the capability of the system and to justify its own line of reasoning in a manner

directly intelligible to inquirer. The style adopted to attain these characteristics is knowledge based programming.

2.2. Types of Expert Systems:

Most knowledge-engineering applications fall into a few distinct type.

Interpretation system such as speech understanding, image analysis etc. infer situation descriptions from observables.

Prediction system infer likely consequences from given situations. It employs parametric dynamic model. The category includes weather forecasting, crop estimates etc.

Diagnosis system infers malfunctions from observables. It relates observed behavioural irregularities with the underlying causes. The category includes medical, electronic, mechanical diagnosis etc.

Design systems develop configuration of objects that satisfy the constraints of the design problem. Such problems include circuit layouts, building design etc.

Planning systems design actions. These systems specialise in problems of design concerned with objects that perform functions. They include automatic programming, robot, project, route etc.

Monitoring systems compare observations of system behaviour to features that seem crucial to successful plan outcome. The crucial feature is vulnerability.

Debugging systems prescribe remedies for malfunction. These systems rely on planning , design and prediction capabilities to create specifications or recommendations for correcting a problem.

Repair system develop and execute plans to administer a remedy for some diagnosed problem. Such systems incorporate debugging, planning and execution capabilities.

Instruction systems diagnose and debug student behaviours. They incorporate diagnosis and debugging subsystems that specifically address the student as the system of interest.

A control expert system adaptively governs the behaviour of the system. To do this, the control system must repeatedly interpret the current situation, predict

the future , diagnose the causes of anticipated problems formulate a remedial plan and monitor its execution to ensure success. This category includes traffic control business management and mission control.

2.3 Components of Expert Systems

Figure 1 shows an idealised representation of an expert system. No existing expert system contains all the components shown, but one or more components occur in every expert system. Each component of this ideal system is described briefly in turn.

The ideal expert system contains a language processor for problem-oriented communications between the user and the expert system; a "blackboard" for recording intermediate results; a knowledge base comprising facts as well as heuristic planning and problem-solving rules; an interpreter that applies these rules; a scheduler to control the order of rule processing; a consistency enforcer that adjusts previous conclusions when new data (or knowledge) alter their bases of support; and a justifier that rationalizes and explains the system's behaviour.

The user interacts with the expert system in problem-oriented language, usually some restricted

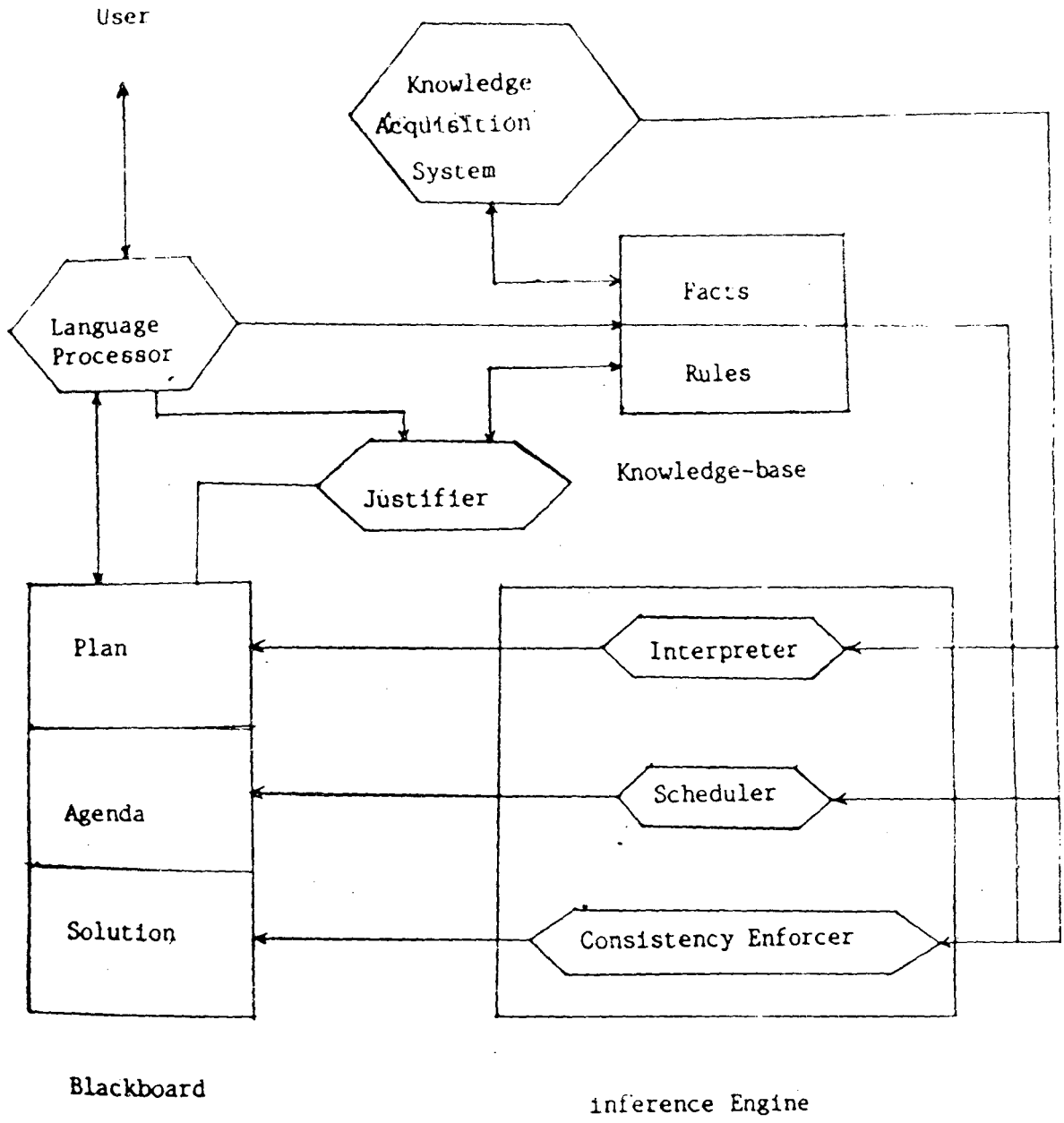


Fig. 1 Components of Expert Systems

variant of English and in some cases via means of a graphics or structure editor. The language processor mediates information exchanges between the expert system and the human user. Typically the language processor parses and interprets user questions, commands, and volunteers information. Conversely the language processor formats information generated by the system, including answers to questions, explanations and justifications for its behaviour, and requests for data. Existing expert systems generally employ natural language parsers written in INTERLISP (Teitelman and Masinter 1981) to interpret user inputs, and use less sophisticated techniques exploiting canned text to generate messages to the user.

The blackboard records intermediate hypotheses and decisions that the expert system manipulates. Every expert system uses some type of intermediate decision representation, but only a few explicitly employ a blackboard for the various types of decisions shown in Fig.1. The figure identifies three types of decisions recorded on the blackboard : plan, agenda, and solution elements. Plan elements describe the overall or general attack the system will pursue against the current problem, including current plans, goals, problem states,

and contexts. For example, a plan may recommend processing all low-level sensor data first, then formulating a small number of the most promising hypotheses, refining and elaborating each of these hypotheses until one best hypothesis emerges, and finally focusing exclusively on that hypothesis until the complete solution is found. This kind of plan has been incorporated in several expert systems. The agenda elements record the potential actions awaiting execution, which generally correspond to knowledge base rules that seem relevant to some decision placed on the blackboard previously. The solution elements represent the candidate's hypotheses and decisions the system has generated thus far, along with the dependencies that relate decisions to one another. Often these dependencies are called links.

The scheduler maintains control of the agenda and determines which pending action should be executed next. Schedulers may embody considerable knowledge, such as "Do the most profitable thing next" and "Avoid redundant effort". To apply such knowledge, the scheduler needs to give each agenda item priority according to its relationship to the plan and other extant solution elements. To do this, the scheduler generally needs to estimate the effects of applying the potential rule.

The interpreter executes the chosen agenda item by applying the corresponding knowledge base rule. Generally the interpreter validates the relevant conditions of the rule, binds variables in these conditions to particular solution blackboard elements, and then makes those changes to the blackboard that the rule prescribes. Interpreters of this sort are generally written in LISP because of its facility for manipulating and evaluating programs. Other languages are also suitable.

The consistency enforcer attempts to maintain a consistent representation of the emerging solution. This may take the form of likelihood revisions when the solution elements represent hypothetical diagnoses and some new data are introduced (Shortliffe 1976; Duda and Gaschnig 1981; Erman et al. 1980) Alternatively the enforcer might implement truth-maintenance procedures when the solution elements represent logical deductions and their truth-value relationship (McDermott and Doyle 1980). Most expert systems use some kind of numerical adjustment scheme to determine the degree of belief in each potential decision. This scheme attempts to ensure that plausible conclusions are reached and inconsistent ones are avoided.

The justifier explains the actions of the system to the user. In general, it answers questions about why some conclusion was reached or why some alternative was rejected. To do this, the justifier uses a few general types of question-answering plans. These typically require the justifier to trace backward along blackboard solution elements from the questioned conclusion to the intermediate hypotheses or data that support it. Each step backward corresponds to the inference of one knowledge based rule. The justifier collects these intermediate inferences and translates them into English for presentation to the user.

Finally the knowledge base records rules, facts, and information about the current problem that may be useful in formulating a solution. Whereas the rules of the knowledge base have procedural interpretations, the facts play only passive roles.

2.4 The Construction of Expert Systems

Workers in expert systems conduct principally empirical research to determine how to solve a problem requiring extensive knowledge and skill. To fashion a solution, they must build a working system by exploiting two types of assets, a methodology and a set of tools.

(i) A Methodology for Building Expert Systems

Because it takes experimentation to achieve high performance , an expert system evolves gradually. This evolutionary or incremental development technique has emerged as the dominant methodology in the expert systems area. The procedure of extracting knowledge from an expert and encoding it in program form is called knowledge acquisition. This transfer and transformation of problem-solving expertise from a knowledge source to a program is heart of the expert-system development process.

The burden of uncovering and formalizing the expert's knowledge falls on the shoulders of the knowledge engineer. Through an extended series of interactions, the knowledge engineering team (the knowledge engineer and the expert) defines the problem to be attacked, discovers the basic concepts involved, and develops rules that express the relationships existing between concepts. Although work is progressing on automating the expert system development process , at present knowledge engineers must rely on their own skill and insight to guide the knowledge acquisition activity. During identification, the knowledge engineer and expert work together to identify the problem area and define its

scope. They also identify the participants in the development process (additional experts), determine the resources needed (time, computing facilities), and decide upon the goals or objectives of building the expert system. A small but interesting subproblem may be identified and used to focus the knowledge-acquisition process.

During conceptualization, the expert and knowledge engineer explicate the key concepts, relations, and information-flow characteristics needed to describe the problem-solving process in the given domain. They also specify subtasks, strategies, and constraints related to the problem-solving activity.

Formalization involves mapping the key concepts and relations into a formal representation suggested by an expert system-building tool or language. The knowledge engineer must select the language and, with the help of the expert, represent the basic concepts and relations within the language framework.

During implementation, the knowledge engineer combines and recognizes the formalized knowledge to make it compatible with the information flow characteristics of the problem. The resulting set of rules and associated control structure define a prototype program capable of being

executed and tested.

Finally, testing involves evaluating the performance of the prototype program and revising it to conform to standards of excellence defined by experts in the problem domain. Typically the expert evaluates the program's performance and assists the knowledge engineer in the forthcoming revisions.

These stages of expert system development are not clear-cut, well-defined or never independent. At best they characterize roughly the complex process we call knowledge acquisition. For example, formalization and implementation are closely related, and failures to provide adequate rules and control during implementation may lead to immediate reformalizations. Also, during testing the knowledge engineering team may find that prototype correction requires partially revising the result of some earlier stage. This could involve reformulation rules and control processes, redesigning knowledge structure, discovering new concepts of abandoning old ones, and even redefining the problem's scope and goals.

(II) Tools for building Expert Systems.

Table 1 lists the primary tools presently available for knowledge engineering tasks. Except for OPS,

they operate in the INTERLISP environment. ROSIE is an excellent general purpose tool for building prototype expert systems. EMYCIN, KS 300 and KA<S are very useful general tools for diagnostic tasks. KAS provides facilities akin to those of EMYCIN but has not yet been used in the variety of ways EMYCIN has KS300 is an industrial system based on the EMYCIN methodology. OPS is the most powerful pure production-system interpreter available. AGE provides the best extant tool for developing a variety of different architectures. INTERLISP provides a LISP programming environment preferred by most workers in the field.

Despite the wealth of knowledge accumulated about constructing expert systems, choosing an appropriate tool for building a particular system remains a difficult yet crucial task. A tool that in some sense well suits a particular problem areas can facilitate the development process, shorten the development time, and lead to a finished product that performs with a high degree of efficiency.

Table 1

Programming systems for building exper systems

Tool	Developer	Features
ROSIE	Rand Corporation	Rule-base, Procedure-Oriented, General purpose
OPS5	Canegie Mellon University	Production-system formalism General purpose
EMYCIN	Stanford University	Rule-based, Diagnosis and explanation
KAS	Stanford Research	Rule-based, Diagnosis and explanation
AGE	Stanford University	LISP-based, Builds various PS architectures
INTERLISP	Xerox Corporation	LISP programming environment

Table 2 Summarizes guidelines for choosing a appropriate expert-system building tool. The predominan consideration involves matching the problem characteristic to necessary tool features and hence particular tools.

Figure 2 outlines a basis for this selectio process. The problem characteristics suggest certai solution features, and these together with the desire expert system features suggest tool features that woul facilitate development. The desired tool features then for

the basis for choosing a particular tool.

Suppose, as a very simple example, that c problem characteristic is uncertain and errorful data a one desired system feature is self-modification (the syst will augment or change its own knowledge). The probl characteristic suggests using certainty combining and tru maintenance as solution features.

TABLE 2

Choosing an appropriate tool for building an expert system

Issues	Maxims
Generality:	Pick a tool with only the generality necessary to solve the problem.
Selection	Let the problem characteristics determine the tool selected.
Speed:	When time is critical, choose a tool with built-in explanation/interaction facilities.
Testing:	Test the tool early on by building a very small prototype system.

Picking a tool with certainty factors and a tru maintenance facility. The desired system feature of se modification suggests the tool should also contain rule control modification facilities.

CHAPTER 3

EXPERT SYSTEM DESIGN

3.1 Introduction

The development of the expert system should begin with thorough analysis of objective. First of all nature of the problem is studied. The problem must be clearly specified and the expected result from the system should also be stated.

Once the objective is defined, the types of problems that need to be solved and the manner in which the program should approach them become readily apparent. In this phase also the need for specificity is great. Although the knowledge should be exhaustive in details it should also be limited to the facts and rules required to achieve the stated objectives. For existence, a heart patient's age and weight are more important in designing there specific for heart disease, but the patient's eye colour and hearing activities are not.

The second phase involves, requiring, structuring and translating the body of expertise required to solve the problems. Both domain experts and knowledge engineers are usually associated with this process. The complexity of the process often determines the type and number of experts.

The planner should find an expert with right type of knowledge for the stipulated objectives and problems. He should focus on expertise acquired from indepth experience and repeated observations. For most expert system, the experienced practitioners are more valuable than academicians in building a large knowledge based system with numerous pre-defined subproblems. It is often advisable to gather information at several levels of the organization, from rank-and file to top management. In building a small knowledge based system with a limited problem scope, a single source is preferable.

Third phase of the design is concerned with the transfer of knowledge that involves the user input from end-user is required early in the design process to ensure that the system will be operationally practical and provide maximum relevance to environment.

3.2 Language and tools

Programming language plays a very important role in expert system. The characteristics of good programming language are that it should be easily implementable and should be user friendly.

The earliest programming languages were imperative or procedural language. Their drawback for using as a AI language was that it was not possible to express logics efficiently in these languages . Accordingly declarative type of languages were developed. In this programming language the statement have a declarative interpretation and can be read as a formal description of the problem without recource of the machine. Functional and logic programmings are the major programming paradigms of declarative languages.

3.3 PROLOG

Prolog-an acronym for programming in logic is essentially rule base programming language. The original objective of its creation was to integrate a development in mathematical logic-the resolution principle proposed by J.A. Robinson into a programming language. Robinson.s resolution principle suggested the application of only one powerful rule of inference to mechanical theorem proving instead of the multiple rules proposed by logicians. This facilitated the design of a programming language that would enable a programmer to make a computer simulate the thinking process by making detection from information given in logical formulas.

not require the programmer to provide the computer with a sequence of instructions to be executed. Instead of telling the computer, what has to be done, the programmer describes the object that must be computed. prolog should be viewed as a formalism for defining knowledge independent of the method of computation.

In order to learn how a machine computes the answer to a question about the knowledge described by a Prolog program, one must realize the two essential properties of the machine.

1. It must be nondeterministic in that it should be capable of exploring several choices at a given time.
2. At each instant, it should be able to reduce a constraint (i.e. to solve a system of equations and inequations).

The recursive version of this nondeterministic m/e which searches the entire search space (search can be directed using predicated fail and cut), is shown in the figure.

The solve procedure used a number of functions-unity, copy, head, tall, append. Functions all the above are clear except the function unity and it will be described

below. Given two trees (trees represent knowledge), the unify procedure tries to make these look alike by some substitution (called bindings) to the variables. For a given path in a search tree (which is traversed in a depth first fashion), the system keeps track of all the bindings the instantiations of all the variables are kept on a stack named trail (called env here). So, the unity function tries to unify 2 trees in a given environment. The substitutions made (if any) are added to the env to give newenvironment (newenv). If unify reaches a point in search space in which a particular substitution conflicts with the previously made substitution (present in the trail), then system back tracks.

Unification:

Terms 1 , 2	(constant) C2	(Variable) X2	(Composite)
(constant)	Succeed if C1 - C2	succeed with	fail
(Variable) X1	Succeed with X1 : -C2	Succeed with X1 :-X2	succeed with X1 : -T2
(Composite)	fail	Succeed with X2:-T1	succeed if with (1)T1 and T2 have same functor and arity (2)The matching of corresponding children succeeds

3.4 Rule base exaust system

A rule based system has two primary components the knowledge base and inference engine.

(1) Knowledge base

The knowledge base is the data or knowledge used to make decisions. It consists of a set of clauses where each clause is either a fact about the given information or a rule about how the solution may relate to or be inferred from the given facts.

Some examples of facts in Prolog are

English	Prolog
Ritu is female	female (Ritu)

In Prolog, a rule consists of a head and a body. The head represents the conclusion and body is antecedent. The body consists of one or more premises. The premises in the antecedent form a conjunction of goals and must be satisfied for the goal to be true. If all the premises are true then the conclusion is true, if any premise fails, the conclusion fails.

As an example:

If condition 1 and condition 2 are true then take action.

This can be represented as

ACTION:- CONDITION 1, CONDITION 2,

(ii) **Inference Engine**

The inference engine has two functions - inference and control. The inference involves matching and unification. the control function determines, the order in which the rules are tested and what happens when a rule succeeds or fails. The inference engine takes the facts that what it knows are true from the data and rule-base and uses them to test the rules in database through the process of unification. When a rule succeeds, the conclusion is added to the data base. When more than one rules may unify with the known facts then control function must determine, through conflict resolution, which rule to fire.

The matching can be of two types. the simplest type is the match to an exact pattern of literals. The use of variables is another source of matching. Matching operation is called unification which has been dealt in Prolog.

Backtracking is the mechanism that instructs the Prolog where to go to Look for the solution to the program. This process gives to prolog the ability to search for all

known facts and rules for a solution. the solution of the compound goals proceeds from left to right. If any condition in chain fails, Prolog backtracks to the previous condition, tries to prove it again with an ordered variable binding, and when moves forward again to see if the failed condition will succeed with new binding. Prolog moves relentlessly forward through. The conditions trying away available binding in an attempt to get the goals to succeed in as many ways as possible.

The expert system process symbolic representation of reality by means of heuristic rules, usually by the technique of backward chaining, the engine begins with a terminal conclusion, the object goal. The program searches through the rule test for my rule that has the object goal as its conclusion. The engine then tests each part of the rules premises and the process begins a new.

Forward chaining progresses from a given information to goal. Starting from what is initially known, the current state of the knowledge is used to make a chain of inferences until either a goal is reached or a solution is said to be unattainable after exceeding some cut off in use of resources.

3.5 Frame based architecture of expert systems

A frame is a data structure for representing a stereotyped situation, like being in certain kind of living room or going to a child's birthday party. Attached to each frame are general kind of information. Some is about what one can expect to happen next, some is about what to do if those expections are not confirmed.

We can think of a frame as a network of nodes and relations. The "top level" of a frame is fixed and represent things that are always true about the supposed situation. The lower levels have many terminals - "roots" that must be filled by specific instances of data.

Collection of related frames are linked together into frame systems. The effects of importants actions are mirrored by transformation between the frames of a system.

CHAPTER - 4

4. System implementation

4.1 Introduction

Early attempts in the field of expert systems were made on generalized problem solving, but it was found that these systems did not offer sufficient power for high performance. Later knowledge based system were developed which emphasised both the accumulation of large amount of knowledge in single domain and development of domain specific techniques in order to develop a high level of expertise. But the development of such large expert systems requires a team of people trained in knowledge engineering and specific field of application. Moreover this type of system requires huge amount of memory and can be implemented only on mainframe computers. From commercial point of view, these systems are not cost effective for organization and individual.

Keeping in view the above facts, small expert systems have gained wide popularity among the researchers of this discipline in recent years. Small expert systems are designed to run on personal computers. They are designed to help individuals deal with small but nasty problems. In some cases, these problems require knowledge engineering techniques because these are not easily amenable to

traditional solutions. In other cases the problems are simply ones that users understand and want to solve. Users may not understand the traditional programming techniques that would be required to solve these problems, but they will be able to turn to knowledge based approaches simply because of the user friendly interface and rapid prototyping features that are available.

Probably the most promising strategy for an individual exploring applications is to purchase and experiment with one of the small tools that are now in the market place. There are many reasonably small problems for which knowledge based systems offer cost effective solutions. And in the end there is no substitute for the experiences gained from building a prototype system. With a small investment in software and training and a properly focussed project, one can develop a useful prototype. This approach will allow one to investigate the field and develop interest within his company without making a commitment.

The expert system developed here is small one for diagnosing the causes of problem with patients who wear contact lenses and to make recommendation for their treatment. program has been written in Turbo Prolog. It uses both frame and rule methods for representing the knowledge.

4.2 Problem formulation

Contact lenses are an increasingly popular replacement for conventional spectacles. Technology has produced many alternatives in lens materials, so that many people can improve their appearance, vision, and enjoyment of physical activities such as dance and athletics. For patients who are having problems with their lenses, there are many possible causes and several approaches to treatment short of reverting to spectacles.

The expert system begins by interviewing the patient. As the patient responds to each question, the system dynamically builds an abstract picture of the patients general health condition, the contact lens history, the patient's goals in seeking advice. Once the patient interview is over, the system applies a set of heuristics, or "rules of thumb", to the patient profile.

There are basically four diagnostic outcomes:

1. A total refit may be recommended, which means that patients will start from the beginning and have their eye measure and their vision checked, and the whole process of monitoring their progress will take place. This is the most expensive alternative, and it must

be thoroughly justified.

2. A change of lens material may be recommended for certain patients, Contact lenses can be made of many different types of plastics and other materials, and if it can be shown that the patient is allergic to the present material, then simply changing the type of lens material may be successful.

3. A pair of new lenses made of the same material is usually suggested for patients where it is believed that the present lenses are too old or that the lense have become damaged in some way. In addition, if the patient's vision has changed, a new prescription can be given without a refit.

4. An investigation of the patient's lifestyle may be recommended if the patient's interview does not suggest any of the other three alternatives. patients who wear cosmetics, take various medications, or work in a place where they are exposed to dust, fumes, or chemicals may be having adverse reactions to these factors rather than to their lenses. In this case, any changes to the lenses cannot be recommended until it is proven that these other factors are not causing the specific problem.

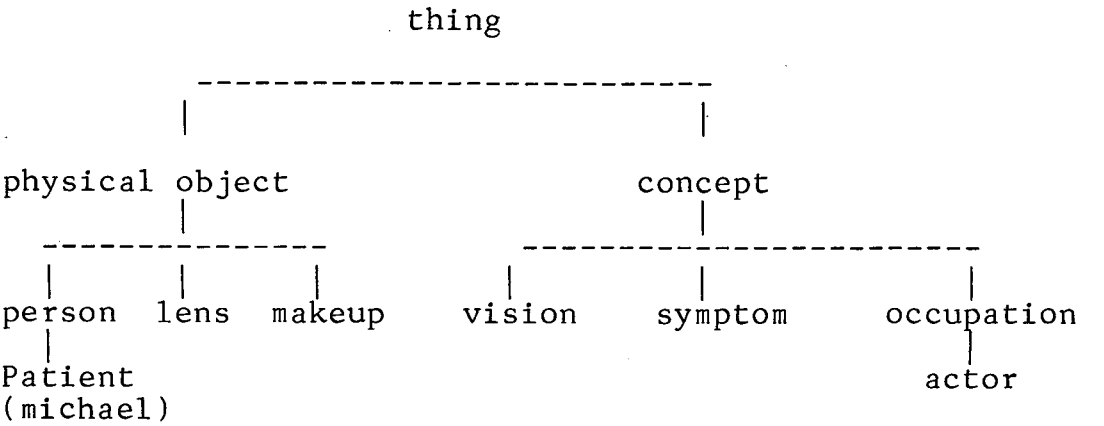
4.3 Architectures of the system

(i) Frame based architecture

The frame system in the expert system is a complete semantic network. The network contains information relevant to the patient.

The frames are constructed and filled in as the patients answer questions. At the top level, there is an empty frame called "thing". Everything is a "thing". Things are either physical objects or abstract concepts. Thus, the PROLOG program contains three "seed" frames.

```
Thing.
Physobj (thing, end).
Concept (thing, end).
```



The two arguments in the physobj and concept frames indicate that physobj and concept are "a kind of' thing, and the "end" aim simply indicates that there is no instance of a physical object or a concept when the program begins. The concept of a frame always relating to the next higher level, in that it is "a kind of" something else, is very important. In other frame systems, "a kind of" may be abbreviated as AKO,. The AKO is the connection between frames that helps to build the semantic network. Each more specific frame inherits the general characteristics of the frame that it is a kind of. for example, one could say that a characteristic of a physical object is that it has weight. In creating an instance of a physical object, a person for example, the person frame inherits the characteristic of having weight.

In this frame network, frames with slots will be generated that apply specifically to the problem domain of diagnosing contact lens difficulties. The first two frames to be created will be the person frame and a frame for the specific individual, which will use the person,s first name as the predicate name. Experts first patient will be named Michael. The only information needed in the person frame is that a person is an instance of a thing and that Michael is an instance of a person. The finished person frame will look like this:

Person (thing, Michael)

Specific facts about Michael are needed understand his condition, and these facts will form slots in the michael frame. Some of the facts that syst needs to know about Michael will allow that system to m certain inference about other facts.

Here is a list of facts the system needs to k about Michael as a person, and why they are needed:

1. The system needs to know if Michael is male female. because if Michael is a female, it appropriate to ask about various cosmetics that have a bearing on the problem. (If Michael is actor, if can be inferred that he will also use s stage cosmetics)

2. The system needs to know Michael's age because it then be inferred that he is a student if he is unc 18 and retired if he is over 65.

3. The system needs Michael's occupation because if can't infer that he is a student or retired, then h occupation will allow us to infer where he works a the types of workplace hazards that could affect h contact lens problem.

4. The system needs to know Michael's general health condition and whether he has any allergies, for allergy medications could affect contact lens comfort.

Four slots will be created as the information comes in through the interviewing process. All of the interview questions are handed through a set of similar predicates. The predicate for question#2, which asks the patient's first name, is given below

```
Question2:- Prprompt (2, [H,2]),                    (1)
            explain_demon (H,2), question2;         (2)
            make_person_frame (H)).                 (3)
```

The prprompt predicate displays the right question according to the number supplied by the first argument. It also collects patient's answer as a list of atoms.

The explain_demon predicate is another important concept in expert system design. A demon is a procedure that is always active in a program, waiting for a particular event to happen. The presence of the demon is not obvious to the user. In this case, the purpose of the demon is to keep checking every response the user makes to see if the user has requested an explanation, that is, the user has typed in 'why'. If the variable H is instantiated to the atom 'why', then the demon becomes active and produces the

appropriate message according to the number supplied to it.

The predicate for creating the person and michael frames is called make_person_frame. It requires an argument that is the patient's first name.

```
make_ Person_frame (H) :-
        (retract(physobj(thing,end)); true),                    (1)
        assertz (physobj (thing, person)),                      (2)
        assertz (person (physobj,H)),                           (3)
        Clause  =..  [H, person, end],                          (4)
        assertz (Clause),                                       (5)
        assertz (slot(H, firstname, value,H)),                  (6)
        (name_check (H,L), assertz (slot (H, gender,
        value,L));                                              (7)
        assertz (slot (H, gender, if_needed,
        ask_gender)))                                           (8)
```

In (1) and (2), the database is updated to reflect that now the physobj frame has an instance, person. In (3), a person frame is created with an instance, michael, Michael is a kind of person, person is a kind of physical object and physobj is a kind of thing. In order to create the michael frame, the "univ" (=..) predicate is used to create a clause with michael as the functor, and person and end as the arguments. The atom and means that there is no instance of michael.

All slots will have the same general format for every slot in every frame.

For example:

Slot
(frame_name,slot_name,how_filled,specific_attribute).
The four arguments in any slot clause will represent:

1. frame name: The frame name of the frame the slot belongs to.
2. slot name: Which attribute of the frame the slot is representing.
3. how filled: How the slot is to be filled, for example, value for an atom, integer for an integer.
4. specific attribute: The specific atom, integer, or list that represents the attribute.

The first slot created in (6) is for the first name attribute. It may seem redundant because the frame is named michael.

The subgoals in (7) and (8) show an additional aspect of frame-based systems-the procedural attachment. Instead of a specific value as an attribute, the nature of some slots requires that something else must happen when the slot is created. This example deals with the simple but

critical issues of Michael's gender. The program is equipped with a list of common first names and the assumed gender. The predicate name - check has a first name and the atoms 'male' or 'female' for its arguments. A real production program running on a large enough system should have hundreds of names. For the purposes of illustration, our system does not contain any information about the name Michael. If it did, the subgoal name_check would succeed and the following slot would be created:

```
slot(michael, gender, value, male).
```

Because Michael is missing from the database, the gender cannot be inferred: therefore, Michael must be asked about this. A slot for gender is created with a procedural attachment:

```
slot (michael, gender, if needed, ask_gender).
```

The "if-needed" indicates that the slot has no specific attribute value, and something must be done to properly fill it. The "ask_gender" is the name of the procedure to invoke to resolve the gender question. There are three basic types of procedural attachments. The "if needed" type is used whenever more information must be collected to fill a slot. An "if added" procedural attachment indicates that a procedure must be performed if a particular frame or slot is added. An "if removed"

indicates that something must be done if a slot or fram removed.

The system will recognize the "if_needed" in gender slot and invoke the ask_gender goal. Be changing the contents of the gender slot, the system also add a clause to the check_name predicate for the Michael as a male name. This is one way an expert sy can become a little more intelligent. Now, the next somebody named Michael used the system, the system infer that Michael is a male and will not ask.

Another situation for an "if needed" proce attachment is created when the system obtains the pati age. If the patient's age is under18, the system creat special occupation slot with the atom default as the ' filled" argument, and student as the attribute value. the age is 65 or older, then a default slot is created the attribute value retired. If the patient is between ages of 18 and 65, then it will be necessary to ask a occupation. A slot with a procedural attachment will created. These are the possible occupation slots could be created after the patient's age is collected.

slot (michael, age, integer, 35).

or

Slot (michael, age, if_needed,ask_age).

or

slot (michael, occupation, default, student).

or

slot (michael, occupation,1 default, retired).

or

slot (michael, occupation, if_needed, ask_job).

Because Michael has given his age as 35, the occupation slot with the procedural attachment will be created.

The next step involves the creation of two more frames, p-occupation, for patient occupation, and a specific frame for the occupation. Michael is an actor, and the system contains knowledge about where an actor works and the various risk factors in the worksplace that could have an adverse effect on contact lens wear. The frames and slot will be constructed as follows:

concept (thing, p_occupation).

p_occupation (concept,actor).

actor (p_occupation,end).

slot (actor, workplace, value, stage).

slot (actor, risk_factors, value, [makeup, dust, stress]).

The predicate that creates these slots is called make_occupation-frame. The next group of questions

collect information that will cause the lens frame to be created and filled in. the system contains several small databases that make the following items recognizable.

1. The type of contact lens worn. There are three basic types: conventional firm lens ("firm"), "gas_permeable" (a firm lens), and soft ("soft") lens. For these types, the system also has a list of acceptable synonyms.

2. A list of cut-rate contact lens providers. "These are usually retail outlet setups that offer contact lenses for very low fees. The low fee structure frequently precludes the careful followup needed to give a patient a good fit with the right type of lens. If the patient's doctor or contact lens center is not in the database, the system cannot assume that the doctor did not do the proper followup.

3. A fictional list of popular lens care products. It's important for the doctor to know how the patient has been cleaning his or her lenses.

Besides this information the system must also collect information about when the patient was last fit with lense and how long the patient has been wearing the present pair of lenses. At the end of the grip of questions about

the contact lenses, the following frames and slots will be created.

```
Physobj (thing, lens).
lens (physobj, end).
slot (lens, type, value, soft).
slot (lens, when_fit, integer, 80).
slot (lens, time_num,integer,1).
slot(lens, time_term,value, [year]).
slot (lens, who fit, value,smith).
```

The next group of questions addresses the specifics of the patient's vision problems. The patient may type his or her symptoms in a natural free format. The system will simply look for key words and match them upto a list of synonyms for a particular eye problem. The system builds a symptom frame, with slots carrying information about the duration of the symptoms and the type of symptoms.

ii) Rule-Based Architectures

Rule-based architectures are best suited for diagnostic tasks. For this system, rules will take the general form: If certain condition(s) exist(s), then draw a conclusion.

The diagnostic part of the system is primarily a set of rules that make inferences about the patient's

problem by investigating the contents of the various frames and slots that were built during the interview phase. As the system checks for existing conditions, it determines which of the four possible diagnoses are indicated: a new fit, a new pair of the same type of lenses, a new pair of lenses of a different type, or a possible change in cosmetics or environment. The conditions are given different weighting factors, and the diagnosis that has the highest score in weighting factors is offered as the system's conclusion. The weighting factors are not probabilistic in this case: they reflect the relative importance of the condition for arriving at a specific conclusion.

Several of the more complex rules will be discussed to clarify the system design. The first rule considers the type of lens and the age of the patient's pair of lenses.

Rule: If the patient is wearing a firm lens and has been wearing the same lenses for three years or longer, or the patient is wearing a soft lens and has been wearing the same lenses for 18 months or longer, then a new pair of the same type of lens is suggested because the patient's lenses have exceeded their average life span.

Here is this rule in PROLOG:

```
Check lens age :-  !.

     Slot(lens, type, value, soft),
     Slot(lens, time-num, integer, X),
     (Slot(lens, time-term, value, months)
     X >= 18, assertz (factor (1,8, new lenses));
     Slot (lens, time-term, value, years),
     X > 1, assertz (factor(1,8, new lenses)),!.

Check lens age :-

     Slot (lens, time-term, value, years)
     X >= 3, assertz (factor (1,8, new-lenses)).

Check-lens-age : - !.
```

The first clause is for the situation of a patient with a soft lens, and the second clause is for any other type of lens (gas permeable or firm). The last clause is an unconditional "catch all" in case the information supplied by the patient is inconclusive. The system could easily be modified to reinterview the patient for any critical information that is missing. As each rule which is tested is found to apply to the patient. that is, the conditions are true, a factor clause is added to the database. The factor predicate has the following general format:

Factor (Factor-number, Factor-weight, Conclusions)

The factor number is an integer that will be used to access the right corresponding message when the system delivers its diagnosis and recommendations. The factor weights are integers between 0 and 10, and are "inexact" reflections of the relative importance of the condition of the conclusion. The factor weights are selected to make the system's opinions come out the same as the doctor's opinion.

The second rule presented in detail concerns the case in which a patient has always had poor vision:

Rule: If the patient is wearing firm lenses and has always had poor vision, then assume a refit is necessary with a toric(shaped) lens or a lens that would center better than the present lens, or if the patients is wearing soft lenses and has always had poor vision, then a refit may be necessary with a newer lathe-cut material or a toric (shaped) hydrogel lens. If the patient's poor vision is relatively recent, then assume that the lenses have aged or a new prescription is needed.

```
Check-problem-period :-
        slot (lens, type, value, X), not (X = soft),
        slot (vision, quality, value, poor),
        slot (vision, time, value, always),
        assertz (factor (3,9, refit)), !.
```

```
Check-problem-period :-
        slot (vision, quality, value, poor),
        slot (vision, time, value, always),
        assertz (factor (4,9, refit)).
Check-problem-period :-
        slot (vision, time-term, value, List),
        not (List = [year]),not(List = [years]),
        assertz (factor (5,9, new-lenses)).
```

The first clause is for the condition where a firm or gas permeable soft lens patient has always had poor vision. The second clause is for the soft lens patient who has always had poor vision. The last clause is for any patient whose poor vision has developed within the last 12 months.

In this particular expert system, the semantic network created by the frames is relatively simple and is very much controlled by the part of the program that performs the patient's interview. There is little danger of "unattached" slots or frames - slots that belong to a nonexistent frame or frames that are "a kind of" nonexistent frame. Unattached slots or frames mean that potentially valuable knowledge is lost. A frame-based semantic network uses the relationships provided by the network as a means of producing a certain level of cognitive understanding of a situation. Many systems that implement natural language

understanding are frame-based. For example, the relationship between the specific patient (e.g., Michael) and the actor frames is used to make a connection between the fact that Michael is an actor and therefore may wear makeup that causes problems with his contact lenses.

What about systems that may have frames and slots created by many different processes, and systems that have a complex and rapidly changing network? There is a danger that frames and slots may become accidentally detached from the network. There is a danger that frames and slots may become accidentally detached from the network, especially during program development and testing. it's helpful to have a process that checks for broken links in the network before the program attempts to form some theory of understanding using the network as a basis.

After the interviewing process is complete, the system engages a goal called check_semantic_network. This goal makes certain that every slot has a frame and every frame is "a kind of" some more general frame and that, ultimately, every slot and frame can be related to thing. If there are nay missing links, the information is displayed at the user console. Of course, this is an undesirable even, and one hopes it happens only during program testing, if at all.

CHAPTER 5

CONCLUSION

This expert system is a small expert system developed to diagnose the causes of the problems with patients who wear contact lenses and to make recommendations for their treatment . This expert system has been implemented in Turbo Prolog. All the knowledge used in the system has been implemented using frames and rules. The system has been implemented on IBM-PC under MS-DOS operating system.

The system has the capability to explain why certain questions have been asked by it. The expert system begins by interviewing the patient. It builds up data base from the answers. The diagnostic system then performs the diagnosis. It can recommend any of the four actions such as total refit, change of lens material, a new pair of lenses of same material or change of life style depending upon the inference which it has drawn from the interview.

A: PROGRAM.

```
/*-------------------------------------------------------*/
/* medical expert  system  for  contact lens wearers     */
/*                                                       */
/*-------------------------------------------------------*/
give_advice:-  perform_interview,check_semantic_network,
               perform_expert_diagnosis.
give_advice:- nl,printr('Thank you.').
     perform_inteview:-give_info,
     question1,questin2      ,question3,question4(Age),
     question5(Age), question6, question7, question8,
     question9, question10(Ans),(Ans=no;
     question11,question12, question13, question14,
     question15,     question16,question17,question18),
     question19.
check_semantic_network:-
                clause(slot(Frame,X,Y,Z,Body),Body=true,
                Frameclause=..[Frame,Higher,end],
                clause(Frameclause,Fbody),Fbody=true,
                Frameclause=..[Frame,Higher,Lower],
                get_to_thing([Frame,Higher,Lower]),
                fail.
check_semantic_network:- !.
get_to_thing([A,thing,C]) :-thing,!.
get_to_thing([A,B,C]):-
                Next_up = ..[B,Higher,A],
                (clause(Next_up,Body),Body=true;
                printr('Missing link ...'),write('from'),
                write(B),nl),!,
                Next_up=..[B,Higher,A],
                get_to_thing([B,Higher,A]).

givo_info:- nl,tab(20),printr('Exeprt advice for
                 contact lens users.').
                 tab(20),printr('**********************'),
                 nl,nl,
                tab(15),printr('Please answer  every
                question ,and your answers' ),
                tab(15),printr('with a carriage return.
                Anytime you want to know'),
                tab (15),printr ('Why I am asking you a
                particular question,please'),
                tab(15),printr( 'type why , and I will
                explain it to you.'),nl,nl,
                tab(30),printr('I hope I can help you
                today.'),nl,nl.

prprompt(X,[H|T]) :-
                nl,nl,write('Question : #'), printr (X),
                message(X),
                getdata([H|T]).
```

```
printr(String) :- write(String),nl.

question1 :- prprompt(1,[H|T]),!,
                (explain_demon(H,1),question1;
                (H= yes; H=no,sorry; question1)).
 sorry :-  nl, printr('I am sorry, but this expert
                system can help only people.'),
                printr ('with contact lenses.'),
                give_advice,nl.
 question2 :- prprompt ( 2, [H, T]),!,
                (explain_demon(H,2),question2;
                make_person_frame(H)).
 question3 :- slot (Person,gender,if_needed,Goal),
                call(Goal).
 question3:- !.
 ask_gender :- prprompt(3, [H|T]),Sex= H,!,
                 ((explain_demon( Sex,3),ask_gender);
                 (check_gender (Sex),ask_gender)),
                 fix_gender_slot(Sex).
 question4(Age) :- prprompt(4,[H|T]),!,
                 Age = H, !,
                 (explain_demon(H,4),question4(Howold);
                 add_age_slot(Age)).

 question5(Age) :- slot(Person,occupation,
                 if_needed,Goal), call(Goal).
 question5 (Age):- !.
 ask_job :- prprompt(5,[H|T]),oc = H, !,
                ((explain_demon(oc,5), ask_job);
                add_job_slot(oc)).
 question6 :- prprompt(6,[H|T]),!,
                (explain_demon(H,6),question6;
                create_lens_frame,
                lens_lookup(H, Lens),
                add_lens(Lens).
 question7 :- prprompt(7, [H|T]),!,
                (explain_demon(H,7),question7;
                process_year_info(H)).
 question8 :- prprompt (8, [H|T]),!,
                (explain_demon(H,8),question8;
                process_time(lens, [H|T])).
 question9 :- prprompt(9, [H|T])),!,
                (explain_demon(H,9),question9;
                check_doctors(H)).
 question10(Ans) :- prprompt(10,[H|T]),!,
                (explain_demon(H,10),question10(A);
                make_vision_frame([H|T],Ans)).
 question11 :- prprompt ( 11, [ H|T]),!,
                (explain_demon(H, 11),question11;
                (H=yes,assertz(slot(vision,time,
```

```
                    value,always));
                    process_time(vision,[H|T]))),
  question12 :- prprompt(12,[H|T]),
                    (explain_demon(H,12),question12;
                    create_symptom_frame([H|T])).
  question13 :- prprompt (13, [H|T]),
                    (explain_demon(H,13),question13;
                    create_care_slot([H|T])).
  question14 :-(p_occupation(Y,actor);slot(Person,
                    gender,value, female)),
                    prprompt (14,[H|T]),
                    (explain_demon(H,14),question14;
                    create_makeup_frame([H|T])).
  question14:- true.
  question15 :- prprompt (15,[H|T]),
                    (explain_demon(H,15),question15;
                    create_health_slot([H|T])).
  question16 :- prprompt(16,[H|T]),
                    (explain_demon(H,16),question16;
                    create_allergy_slot([H|T])).
  question17 :- prprompt (17, [H|T]),
                    ( explain_demon(H, 17), question17;
                    (H=yes,!;update_workplace([H|T]))).
  question18:-prprompt(18,[H|T]),
                    (explain_demon(H,18),question18;
                    (H=yes,assertz(slot(symptem,time ,
                    value, always));
                    process_time(symptom,[H|T]))).
   question19:-prprompt(19,[H|T]),
                    (explain_demon(H,19),question19;
                    add_goals_slot(H).
   thing.
   physobj(thing,end).
   concept(thing,end).
   printer(String) :-Write (String),nl.

   make_person_frame(H):-
        (retract(physobj(thing,end);true),
        assertz(physobj(thing,person)),
          assertz(person(physobj,H),
          Clause=..[H,person,end],
          assertz(Clause),
          assertz(slot(H,firstname,value,H)),
          (name_check(H,L),assertz(slot(H,gender,value,L);
          assertz(slot(H,gender,if_needed,ask_gender))),!.
add_age_slot(Age):- key_age(Age).
key_age(Age):- integer(Age),age<18,
                person(physobj,Person),
      assertz(slot(Person,age,integer,Age)),
     assertz(slot(Person,,occupation,default,student)),!.
```

```
 key_age(Age):-
          integer(Age),Age>18,Age<65,
          person(physobj,Person),
       assertz(slot(Person,age,integer,Age)),
       assertz(slot(Person,occupation,if_needed,ask_job)),!.
key_age(Age):- integer(Age),Age>65,person(physobj,Person),
          assertz(slot(person.age,integer,Age)),
        assertz(slot(Person,occupation,default,retired)),!.
key_age(Age):-not(integer(Age)),person(physobj,Person),
       assertz(slot(Person,age,if_needed,ask_job)),
       assertz(slot(Person,occupation,if_needed,sk_job)),!.
add_job_slot(H):-person(physobj,Person),
               occupation(H,Workplace),
               retract_old_job,
               assertz(slot(Person,occupation,value,H)),
               make_occupation_frame(H).
add_job_slot(H):- person(physobj,Person),
                retract_old_job,
          assertz(slot(Person,occupation,value,unknown)).
make_occupation_frame(H):-
                (retract(concept(thing,end));true),
                assertz(concept(thing,p_occupation)),
                occupation(H,W),workplace(W,R),
                assertz((p_occupation(concept,H)),
                Clause=..(H,p_occupation,end),
                assertz(Clause),
                assertz(slot(H,workplace,value,W)),
                assertz(slot(H,risk_factor,value,R)).
retract_old_job:-person(physobj,Person),
      retract(slot(Person,occupation,if_needed,ask_job)).
fix_gender_slot(H):-retract(slot(Person,gender,X,Y)),
                 add_to_names(Person,H).
add_tonames(Person,Sex):-assertz(name_check(Person,Sex)).
add_lens(Lens):-(retract(physobj(thing,end));true),
                assertz(physobj(thing,lens)),
                assertz(lens(physobj,end)),
                assertz(slot(lens.type,value,Lens)).
create_lens_frame:- assertz(lens(physobj,end)).
process_year_info(Year):- Year=<71,
                ((slot(lens,type,value,unknown),
                retract_unk_lens,
                assertz(slot(lens,type,value,firm)));true),
                assertz(slot(lens,when_fit,integer,Year)).
process_year_info(Year):-Year>71,
               assertz(slot(lens,when_fit,integer,Year)).
retract_unk_lens:-  retract(slot(lens,type,value,unknown)).
process_time(SlotType,[H|T]):-
           member(Int,[H|T]),integer(Int),
           time_terms(List),
           intersection(List,[H|T],Term),
```

```
            assertz(slot(SlotType,time_num,integer,Int)),
            assertz(slot(SlotType,time_term,value,Term)).
make_vision_frame([H|T],Ans):-
            (retract(concept(thing,end));true),
             assertz(concept(thing,vision)),
      (((H=excellent;H=good;H=fair),Ans=no;
      ((H=poor,Ans=yes),Vision=H);Vision=unknown,Ans=no),!,
             assertz(vision(concept,end)),
             assertz(slot(vision,quality,value,Vision)).
create_symptom_frame([H|T]):-
             (retract(concept(thing,end));true),
             assertz(concept(thing,symptom)),
             assertz(symptom(concept,end)),
             fill_sysmptom_slots([H|T]).
fill_symptom_slots([]):-slot(symptom,type,value,X)).
             assertz(slot(symptom,type,value,unknown)).
fill_symptom_slots([H|T]):-
             (symptom(Type,Wordlist),member(H,Wordlist),
             assertz(slot(symptom,type,value,Type));true),
             fill_symptom_slots(T).
create_care_slot([H|T]):-lens_cara(Brand,Type),
             [H|T]=Brand ,
             assertz(slot(lens,care_product,value,Brand)).
create_care_slot([H|T]):-
             assertz(slot(lens,care_product,value,unknown)).
create_make_frame([H|T):-
             (retract(physobj(thing,end));true),
             assertz(physobj(thing,makeup)),
             assertz(makeup(physobj,end)),
             fill_makeup_slot([H|T]).
fill_makeup_slots([]):- slot(makeup,X,Y,Z),
             assertz(slot(makeup,data,value,unknown)).
fill_makeup_slots([H|T]):-
       (cosmetic(H),assertz(slot(makeup,data,value,H)));
             cosm_brand(H,A),
             assertz(slot(makeup,data,value,H));true)
             ,fill_makeup_slots(T).
create_health_slot([H|T]):-person(physobj,Person),
             ((H=excellent;H=good;H=fair;H=poor),
             assertz(slot(Person,health,value,H)));
             assertz(slot(Person,health,value,unknown))).
create_allergy_slot([H|T]):-person(physobj,Person),
             ((H=yes;H=no),
             assertz(slot(Person,allergy,value,H));
             assertz(slot(Person,allergy,value,unknown)).
update_workplace([H|T]):-person(physobj,Person),
             slot(Person,occupation,value,Oc),
             p_occupation(concept,Oc),
             slot(Oc,risk_factor,value,RiskList),
             intersection([H|T],RiskList,[]),
```

```
            assertz(slot(Oc,add_risks,value,[H|T]),
            add_risks(Oc,[H|T]).
update_workplace([H|T]):-!.
add_risks(Oc,[H|T]):-occupation(Oc,Wp),workplace(Wp,Risks),
            append(Risks,[H|T],Morerisks),
              retract(workplace(Wp,Risks),
            assertz(workplace(Wp,MOreRisks)).
 add_goals_slot(H):- person(physobj,Person),
                     (H=a,Goaltype= longer_wear;
                      H=b,Goaltype=extended_wear;
                      H=c,Goaltype= comfort_appearance;
                      H=d,Goaltype=better_vision),
                     assertz(slot(Person,goal,value,Goaltype)).
  add_goals_slot(H):-
                     person(physobj,Person),
                     assertz(slot(Person,goal,value,unknown)).
  check_doctors(Dr):- doctors(Dr),
                   assertz(slot(lens,who_fit,value,cut_rate)),!.
 check_doctors(Dr):-
        assertz(slot(lens,who_fit,value,not_on_file)).
        last(X,[X]).
        last(X,[_|Y]):-last(X,Y).
        remove(X,[X|L],L).
        remove(X,[H|L],[H|L]):-not(X=H),remove(X,L,L1).
        member(X,[X,_]).
        member(X,[_|T]):-member(X,T).
        append([],L,L).
        append([X|L1],L2,[X|L3]):-append(L1,L2,L3).
        intersection([],X,[]).
        intersection([[X|R],Y,[X|Z]):-
        member(X,Y),!,intersection(R,Y,Z).
        intersection([X|R],Y,Z):-intersection(R,Y,Z).
        prefix([],_).
        prefix([X|L],[X|M]):-prefix(L,M).
        appendlist([],L,L):-!.
        appendlist([X|L1],,L2,[X,L3]):-
        appendlist(L1,L2,L3).
        build_list(X,G,_):-assertz(b1(0)),
                  call(G),assettz(b1(X)),fail.
        build_list(_,_,L):- gather_all([],M),!,L=M.
        gather_all(S,L):-
        getnext((X),!,gather_all([X|S],L).
        gather_all(L,L).
        getnext(X):-retract(b1(X)),!,X\==0.
        sum_up([],0).
        sum_up([X|Y],Z):- sum_up(Y,X1),Z is X1+X.
        maximum(Value,Goals,_):-
        make_scrpad(Value,Goals),fail.
        maximum(Value,Goals,Max):-find_max(0,Max),!.
        find_max(V1,V2):-
```

```
        gtnxt(Value),Value>V1,find_max(Value,V2);
                  find_max(V1,V2).
        find_max(V2,V3).
        make_scrpad(Value,Goals):-
        assertz(found(mark)),call(Goals),
                  asserta(found(Value)).
        getnxt(Value):-retract(found(Value)),Value\==mark.
        getdata(List):- keep_reading(List).
        keep_reading([H|T]):-
                  getstring([HL|TL]),
                  not([HL|TL])=[13|_]),
                  (HL>65,HL<91,HX is HL+32;HX is HL ),
                  form_atom_list([HX|TL],[H|T]).
        form_atom_list([],[]) :-!.
        form_atom_list([H|T],[Headword|Tailword]):-
                  prefix(String,[H|T]),
                  (last(32,String),remove(32,Stringword);
                   last(13,String),remove(13,Stringword)),
                  appendlist((String,Rest,[H|T]),
                  name(Atomic,Word),
                  Headword=Atomic,
                  form-atom-list(Rest,Tailwords).
        getstring([H|T]):- get0(H),
                   (not(H=13),getstring(T);!).
time_terms([month,months,year,years,week,weeksday,days]).
        name_check(shailesh,male):-!.
        name_check(shalini,female):-!.
        check_gender(male):-!.
        check_gender(female):-!.
        occupation(executive,office).
        occupation(teacher,school).
        occupation(athelete,sports_arena).
        occupation(actor,stage).
        occupation(waiter,restaurant).
        occupation(factory,factory).
        occupation(secretary,office).
        occupation(computer,office).
        occupation(construction.construction).
        workplace(office,['cigerette smoke',stress] ).
        workplace(school,[stress]).
        workplace(sports_arena,[dirt,stress]).
        workplace(stage,[makeup,dust,stress]).
        workplace(restaurant,['cigrette
        smoke',fumes,stress]).
        workplace(factory,[dust,fumes,chemicals,stress]).
        workplace(construction,[dust,fumes,
        chemicals,stress]).
        lens-lookup(firm,firm).
        lens_lookup(hard,firm).
        lens_lookup(gas,gas_permeable).
```

```
    lens_lookup(soft,soft).
    lens_lookup(hydrogel,soft).
    lens_lookup(hydrophilic,soft).
    lens_lookup( ,unknown).
    doctors(lensomat).
    doctors(contactville).
doctors(lensmark).
symptem(blinking,[blinking]).
symptom(red_eyes,[red,pink]).
symptom(dry_eye,[eye]).
symptom(itch,[itching,itchy,itch,itches]).
symptom(tearing,[tearing,tears,watery,water,watering]).
  symptom(clouding,[clouding,fogging,
  blurring,blur,cloudy,clouding)).
  symptom(squinting,[squinting,squint]).
  symptom(pain,[pain,hurt]).
  symptom(centering,[slip,fall,lose,loose]).
  symptom(foreign_body_sensation,[something,speck]).
    cosmetic(remover).
    cosmetic(mascara).
    cosmetic(eyeshadow).
    cosmetic(eyelinder).
    cosm_brand(suzyq,hypo).
    cosm_brand(luckygirl,hypo).
    cosm_brand(superglitz,no).
    cosm_brand(curtainup,hypo).
    cosm_brand(breakaleg,no).
    lens_care([super,lens],soft).
    lens_care([ultra,lens],gas_permeable).
    lens_care([mazic,lens],firm).
    factor(0,0,drift_type).
    factor(0,0,refit).
    factor(0,0,change).
    factor(0,0,new_lenses).
    factor(0,0,other).
    perform_expert_diagnosis:-
              abolish(question1/1),
              abolish(question2/1),
              abolish(question3/1),
              abolish(question4/1),
              abolish(uestion5/1),
              abolish(question6/1),
              abolish(question7/1),
              abolish(question8/1),
              abolish(question9/1),
              abolish(question10/1),
              abolish(question11/1),
              abolish(question12/1),
              abolish(question13/1),
              abolish(question14/1),
```

```
            abolish(question15/1),
            abolish(question16/1),
            abolish(question17/1),
            abolish(question18/1),
            abolish(question19/1),
            check_lens_age,
            check_previous_doctor,
            check_problem_period,
            check_makeup_used,
            check_allergies,
            check_workplace,
            check_symptom_time,
            check_wearing_time,
            check_finances,
            summarize.
check_lens_age:-
              slot(lens,type,value,soft),
              slot(lens,time_num,integer,X),
      (slot(lens,time_term,value,months),X>=18,
              assertz(factor(1,8,new_lenses));
              slot(lens,time_term,value,years),
              X>1,assertz(1,8,new_lenses))),).
check_lens_age:-
              slot(lens,time_term,integer,X),
         X>=3,assertz(factor(1,8,new_lenses)).
check_lens_age:-!.
check_previous_doctor:-
              slot(lens,who_fit,value,cut_rate),
              assertz(factor(2,9,refit)).
check_previous_doctor:-!.
check_problem_period:-
              slot(lens,type,value,X),
              not(X=soft),
              slot(vision,quality,value,poor),
              slot(vision,time,value,always),
              assertz(factor(3,9,refit)),!.
check_problem_period:-
              slot(vision,quality,value,poor),
              slot(vision,time,value,always),
              assertz(factor(4,9,refit)).
check_problem_period:-
              slot(vision,time_term,value,List),
              not(List=[year]),not(List=[years]),
              assertz(factor(5,9,new_lenses)).
check_problem_period:-!.
check_makeup_used:-makeup(physobj,X),
              assertz(factor(6,8,change)).
check_makeup_used:-!.
check_allergies:-slot(Person,allergy,value,yes),
              assertz(factor(7,9,change).
```

```
check_allergies:-!.
check_workplace:-slot(Person,occupation,value,X),
        occupation(X,Y),workplace(Y,RiskList),
              (member(dust,RiskList);
               member(fumes,RiskList);
               member(chemicals,RiskList)),
               assertz(factor(8,10,change)).
check_workplace:-!.
check_symptom_time:-prprompt(20,[H|T]),
    (explain_demon(H,20),check_symptom_time;
   (H=sudden,assertz(factor(9,6,new_lenses)));
   (H=constant,assertz(factor(8,10,diff_type)));
   (H=gradual,assertz(factor(11,6,new_lenses)));
            check_symptom_time).
check_wearing_time:-prprompt(21,[H|T]),
      (explain_demon(H,21),check_wearing_time;
            (integer(H),check_wearing_time)),
            check_lens_type(H).
check_lens_type(H):-H<12,
        slot(Person,goal,value,extended_wear),
            slot(lens,type,value,soft),
            assertz(factor(12,10,other)).
check_lens_type(H):-!.
check_finances:-prprompt(22,[H|T]),
            (explain_demon(H,22),check_finances;
  H=yes,assertz(slot(Person,finances,value,yes));
  H=no,assertz(slot(Person,finances,value,no));
  check_finances),!.
summarize:-
     build_list(X,factor(_,X,new_lenses),List1),
          build_list(Y,factor(_,Yrefit),List2),
          build_list(Z,factor(_,Z,change),List3),
       build_list(A,factor(_,A,diff_type),List4),
          build_list(B,factor(_,B,other),List5),
  sum_up(List1,Sum1),assertz(total(new_lenses,
          Sum1)),sum_up(List2,Sum2),
          assertz(total(refit,Sum2)),
   sum_up(List3,Sum3),assertz(total(change,Sum3)),
sum_up(List4,Sum4),assertz(total(diff_type,Sum4)),
sum_up(List5,Sum5),assertz(total(other,Sum5)),
          maximum(Value,total(X,Value),Max),
          total(Diagnosis,Max),
          output_diagnosis(Diagnosis,Max).
output_diagnosis(Diagnosis,Max):-
          nl,nl,printr('My conclusion is that  you
  should consider  following:'),
          nl,d_lookup(Diagnosis),nl,
write('because  there is  afactor  weight
          of'),
write(Max),write('.'),nl,nl,printr('explation'),
```

```
                nl,print_factor(Diagnosis).
        ask_if_done:-!.
        d_lookup(new_lenses):-printr('a  new pair of  same
        typeof lenses that you are wearing').
        d_lookup(refit):-printr('anew  refit  for  contact
        lenses').
        d_lookup(change):-  printr('an evaluation of  your
        workplace'),
        printr('cosmetic  products  or  medicine  you  are
        using').
        d_lookup(diff_type):- printr('a different type  of
        lens').
        d_lookup(other):-printr('a    goal    other    than
        extended  wear at this point').
        print_factors(Diagnosis):-factor(X,Y,Diagnosis),
        factor_message(X),fail.
        print_factors(Diagnosis):-!.
    factor_message(1):-nl,printr(The  average life span  of
    a soft lens is 18months.'),
     printr  ('  The average life span of firm  lens  is  3
     years.'),
  printr('because you have been wearing your lenses for a
 period'), printr('which  exceeds the life span ,a new  pair
of lenses is
 recommended').
 factor_message(2):-nl,printr('because  your practioner  was
 one
     that operated with a'),
 printr('cut  rate or feestructure he probably did not  have
   the'),
 printr('resaurces to do the follow up you were'),
 printr(' never fitted probably in the first place').
 factor_message(3):-nl,printr('since  you are  wearing  firm
lens , and have always had poor vision'),
 printr('a  refit  with  a toric  lens is  recommended  for
better vision,').
 factor_message(5):-nl,printr('Since  yourpoor  vision  is
arecent problem,your lenses may'),
 printr('just  be  too  old  or  you  need  a  prescription
change').
 factor_message(6):-nl,printr('Because  you  use  cosmetics
regularly,
  an investigation of'),
 printr('the  specific  products  that  you  are  using  is
recommended ),
 printr('before taking any other action.').
 factor_message(7):-nl,printr('Because  you  said  that  you
have allrgies ,you may be taking '),
 printr('some  medications  that  may  be  causeng  dry  eye
feeling or
```

```
 other.'),
 printr('discomfort.A    careful    investigation    of    any
medication
 should'),
 printr('be performed before anything else is done.').
 factor_message(8):-nl,  printr('Because you are exposed  to
dust, fume,vhemcals at your workplace')'
 printr('it is recomended that environmentbe checked out.'),
 printr('before taking any other action.').
 factor_message(9):-nl,printr('Because  your  symptoms  are
sudden , the problem may be due to a'),
 printr('scratched lens, a fractured lens or due to aforeign
body in the eye.').
 factor_message(10):-nl,printr('Because  your symptoms  have
been constant over time,there'),
 printr(,is  probably some problem with the material of  the
lens.').
 factor_message(11):-nl,printr('Because  your symptoms  have
been gradually increasing,'),
 printr('it  is likely that material of your lenses is  made
of'),
 printr('is   breaking   down   and   causing   even   adverse
chemical.'),
 printr('reactions').
 factor_message(12):- nl,printr('Because you report that you
 can wear your lenses only for a')'
 printr('few hours at a time,it is not realistic to be  able
to use
 estended wear lens at this time.').
 explain_demon(why,1):-nl,printr('unable to help because you
are not aregular contact lens wearer.').
 explain_demon(why,2):-nl,printr('I  want to know your  name
so that
 ican have conversation.').
  explain_demon(why,3):-nl,printr('Iwant to know about  your
sex so that I can infer about cometics.')'
 explain_demon(why,4):-nl,printr('To  make some  assumptions
about life style age is required.').
 explain_demon(why,5):-nl,printr('Job  will  help  to  make
inferences about workplace environment.').
 explain_demon(why,6):-nl,printr('Type  of lens is  required
to help you.').
 explain_demon(why,7):-nl,printr('if  Iknow  what  year  you
were  fit with the glasses then  I   can be more certain  of
type of lens.').
 explain_demon(why,8):-nl.printr('I  must know for how  long
you are using contacd lens to')',printr(' determine  whether
you need complete refit or
  some new lense .').
 explain_demon(why,9):-nl,printr('Your  practitioner may  be
```

```
one who delivered services for very low'),
 printr('  fee  and  therefore did not have  time  to  do  a
careful followup on your fitting.').
 explain_demon(why,10):-nl,printr('General  state of  vision
is required so that questions can be continued.').
 explain_demon(why,11):-nl,printr('Poor   vision   was   from
onset of wearing or started later.').
 explain_demon(why,12):-nl,printr('symptoms  are  important
for diagnosis.').
 explain_demon(why,13):-nl,printr('type  of lens being  used
may be cause of the problem.').
 explain_demon(why,14):-nl,printr('Changing  the  brand  of
cosmetic may help.').
 explain(why,15):-   nl,printr('General   health   is   the
indicator of how well you can wear lenses.').
 explain_demon(why,16):- nl,printr('Allergies may contribute
to the symptom.').
 explain_demon(why,17):-nl,printr(,Job  will  help  to  know
about
   some of environmental conditions.').
 explain(why,18):-nl,printr('If   symptoms  have   developed
later then lens may not be problem.').
 explain_demon(why,19):-nl,printr('To  assess whether  goals
are realistic based on your history.').
 explain_demon(why,20):-nl,printr('It        will        help
diagnosis.').
 explain_demon(why,21):-nl,printr(To  qualify as  acandidate
for  etended  wear  lens you mustwear  for  fullday  without
discomfort.').
 explain_demon(why,22):- nl,printr(Concern over the cost may
be a factor in staying with firm lenses.').
 message(1):-printr('Are   you   wearing   contact   lenses
regularly.').
 message(2):-printr('What is your first name please?').
 message(3):-printr('Are you a male or female?').
 message(4)  :- printer ('How old are you? Please give  your
name in  whole years.').
 message(5) :- printer ('What is job or occupation?').
 message(6) :- printer ('What type of contact lenses are you
wearing:  '),printer  ('Firm,  gas  permeable,  or  do   not
know?').
 message(7) :- printer ('Approxmately in what year were  you
fit  with your present '),
                printer  (' lenses? just give the  last  two
digits  of the year.').
 message(8)   :-   printer   ('  For   how   long   have   you
consistently  been wearing your'),printer ('present  lenses?
Answer in years,  month, weeks or days. ').
 message(9)  :- printer ('Who was your former  contact  lens
practitioner? I will not '),printer ('not contact him or her
```

```
unless you agree to it .').
 message(10) :- printer ('How well are you seeing with your
contact lenses? Is your'),printer ('vision with the lenses:
excellent, dood, fair or poor?').
 message(11) :- printer ('Has your vision always been poor?
If the answer is not yes,'),printer (' then for how long in
years, months, weeks or days/').
 message(12) :- printer (' How do the contact lenses feel?
Describe symptoms.').
 message(13) :- printer ('What type of lens care system have
you been using?'),printer ('Which products').
 message(14) :- printer ('What sort of eye make-up, make-up
remover, or other eye'),printer ('cosmetics have you been
using? Which products ').
 message(15) :- printer ('How is your general health?
excellent, good, fair or poor?').
 message(16) :- printer ('Do you have any allergies?').
 message(17) :- slot(Person, Occupation, Value, Occup),
                occupation(Occup, Wrkpl),
                workplace(Wrkpl,Risks ),
     write('Because your occupatio is that of '),
write(Occup),nl,
     write('I assume that ytour workplace ia a/an
'),write(Wrkpl),nl,write('Where you may be exposed to any of
the following: '),nl,write(Risks),nl,printer('Is this true?
If you have any factors to add,'), printer ('list them
here. ').
 message(18) :- printer (' Have you always had these
symptoms? If not, then for how'),printer ('many years,
months, weeks or days?').
 message(19):- printer ('What are your goals for your
contact lenses? Choose the letter of the),printer ('phrase
which comes closest to your goal. '),nl,
 printer ('a. longer wear'),printer ('b. overnight wear or
extended wear'),printer ('c. better comfort or cosmetic
appearance'),printer('d. better vision').
 message(20) :- printer ("Was the development of your
present symptoms: '),printer ('sudden or gradual or were
they constant?').
 message(21) :- printer('How many hours per day do you wear
your present lenses '),printer('comfortably? Just give the
number of hours. ').
 message(22) :- printer (' Is the cost of the lenses and
lens care of concern '),printer ('to you? Answer yes or no.
').
```

B - REFERENCES:

1. Andrew Basden "On the application of expert system" (published in developments in expert system)

2. Avion B and Feigenbaum, E.A.(1982). The Handbook Of Artificial Intelligence . 3 Vols.(Lsaltos, CA: William Kaufman),1:3.

3. Bruce I.Blum and Ralph E.Watcher. "Expert system application software in software engineering." Published in telcommunication and informatics, vols. 3 of no. 4.

4. Bruce , G.Buchnan and Edward H. Shortliffe, "Rule Based Expert Systems", (Reading, M.A. :Addison -Wesley).

5. Brattle research corporation , Artificial Intelligence and Fiffth Generation Computer Technologies (Boston)

6. Elaine Rich (1983), " Artificial Intelligence", (New York: Mcgraw Hill).

7. Fagan, L. Shortliffe, E. and Buchnan, B.(1980), Comput based medical decision making from Mycin to VM. Autom dica, Vol.3.

8. Farkas P. Kassalow Tw : A guide to refitting the unsuccessful contact lens patients. J Am Optom Assoc 1965 Mar.

9. Hayes-Roth, F. ,Waterman, D.A. and Lenat , D.B. (1983), " Building Expert Systems" (Addison -Wesley)

10. Harmon , P. and King D. (1985), "Expert-Systems", (New Year :Wiley Press).

11. Hector , J.L. (1986) : "Making believers out of computers". Artificial Intelligence 30.

12. Holman ,J.G. and Cookson , M.J., (1987) " Expert Systen for Medical Applications", Journal of Medical Engg. and Technology, Vol.1. no.4.

13. Jackson , P. (1986). " Introduction to expert systems.' , Addison Wesley.

14. Jorgens, J. (1988). " Expert Systems in Clinical Engineering", J. of Clinical Engineering May/June.

15. Nilsson, N. J., "Principles of AI ".(Springer- Verlae Heidelberg).

16. Rich, E. (1983). AI, New York: Mcgraw Hill.

17. Townsend, C.(1988)," Introduction to Turbo Prolog." BPB Publications.

18. Turbo Prolog Reference Guide.Vrsion 2.0,IBM Borland.

19. Turbo Prolog Users' Guide, version 2.0,IBM, Borland.