Artificial Intelligence(CS5329)

Expert System To Diagnose Mental Disorders

AUTHOR :

Akshatha Manohar Jain

netID : a_j289

StudentID : A04824992

Submission date : 03/02/2010

PROJECT TEAM MEMBERS:

1. AKSHATHA MANOHAR JAIN
2. VIDHYASHREE,NAGABUSHANA
3. SUNDARA RAJ SREENATH,SAHANA

1. INTRODUCTION

1.1 Artificial Intelligence (AI):

Artificial Intelligence is a branch of computer science, which focuses on the development of computer systems able to solve the problems which need human like expertise (like, Doctor).

1.2 The Problem Description:

A mental disorder, also called a mental illness or psychiatric disorder, is a behavioral or mental pattern that causes significant distress or impairment of personal functioning. Although the exact cause of most mental illnesses is not known, it is becoming clear through research that many of these conditions are caused by a combination of biological, psychological, and environmental factors. Some mental illnesses have been linked to abnormal functioning of nerve cell circuits or pathways that connect particular brain regions.
Biological factors that may be involved in the development of mental illness include:

- Genetics (heredity): Mental illnesses sometimes run in families, suggesting that people who have a family member with a mental illness may be somewhat more likely to develop one themselves. Susceptibility is passed on in families through genes.
- Brain defects or injury: Defects in or injury to certain areas of the brain have also been linked to some mental illnesses.
- Other factors: Poor nutrition and exposure to toxins, such as lead, may play a role in the development of mental illnesses.

Psychological factors that may contribute to mental illness include:

- Severe psychological trauma suffered as a child, such as emotional, physical, or sexual abuse
- An important early loss, such as the loss of a parent
- Neglect
- Poor ability to relate to others

Environmental Factors that may contribute to mental illness include:

- Death or divorce
- A dysfunctional family life
- Feelings of inadequacy, low self-esteem, anxiety, anger, or loneliness
- Changing jobs or schools
- Social or cultural expectations (For example, a society that associates beauty with thinness can be a factor in the development of eating disorders.)
- Substance abuse by the person or the person's parents.

Mental illness treatment can take place in a variety of settings and typically involves a multidisciplinary team of providers such as counselors, psychologists, psychiatrists, nurses, mental health aides, and peer support professionals. Mental health problems can vary greatly from person to person, even among those with the same mental health diagnosis.

- Psychiatric hospitalization.
- Inpatient or residential mental health treatment.
- Outpatient mental health treatment.
- Dual diagnosis treatment.
- Psychotherapy.
- Medication.

With modern inventions in Medical Science, there are methods to identify the mental illness and provide the treatment for complex diseases . Treatments and precautions need to be available in all the countries and regions.

1.3 Expert System as solution :

An Expert System is a computer system that emulates the decision making ability of a human expert. Thus, it acts like a human expert. It uses expert(human) knowledge to solve problems that would require human intelligence. There are a lot of applications in artificial intelligence domain that try to help human experts offering solutions for a problem. This project describes medical expert systems developed in order to make some predictions regarding various diseases.

Medical expert systems are designed to improve patient care by optimizing medical decision making. The distinguishing feature of medical expert systems is that they make recommendations based on input data; they are differentiated from decision support systems.

Solution for the Problem is developing an expert system which will be really helpful and effective to diagnose the mental illness / disorder, finding and recommending treatments to the respective disorders based on the diagnosis. Based on the given symptoms from the patient, patients can easily diagnose mental illness from Expert systems, whether they are suffering from any depression or mental illness then they can start medication based on the diagnosis. Some of the diagnosis needs clinical treatments which needs doctor assistance.

To do this we are using AI concepts i.e. by using data structures ,decision trees of Backward chaining and Forward chaining. We use Backward chaining first, by knowing the symptoms from the user we diagnose the mental disorder which is done by creating decision tree and forming IF and THEN rules which will be our Knowledge base.

According to the IF-THEN rules, we will ask the patient for the symptoms, based on the inputs from the patient the expert system will diagnose the treatment using forward chaining.

In short, we use Backward chaining to detect the mental disorder and forward chaining to provide the best possible treatment. We have collected the information for mental disorders and their treatments from doing the research on internet, for more accuracy expert doctors suggestions will be helpful.

2. TEAM CONTRIBUTIONS

2.1 Team Members Names:

1. Akshatha Manohar Jain

2. Vidhyashree Nagabhushana

3. Sahana Srinath

2.2 Contribution:

1. Akshatha Manohar Jain:

- Collected information from the internet for 10 mental disorders , treatments and their symptoms.
- Helped in giving base idea of how to approach decision tree.
- Created and finalized forward decision tree.
- Reviewed the IF – THEN rules for both Forward and Backward Chaining for changes/ suggestions.
- Done the testing of Forward chaining function.
- Done the testing of backward chaining function.

2. Vidhyashree Nagabhushana

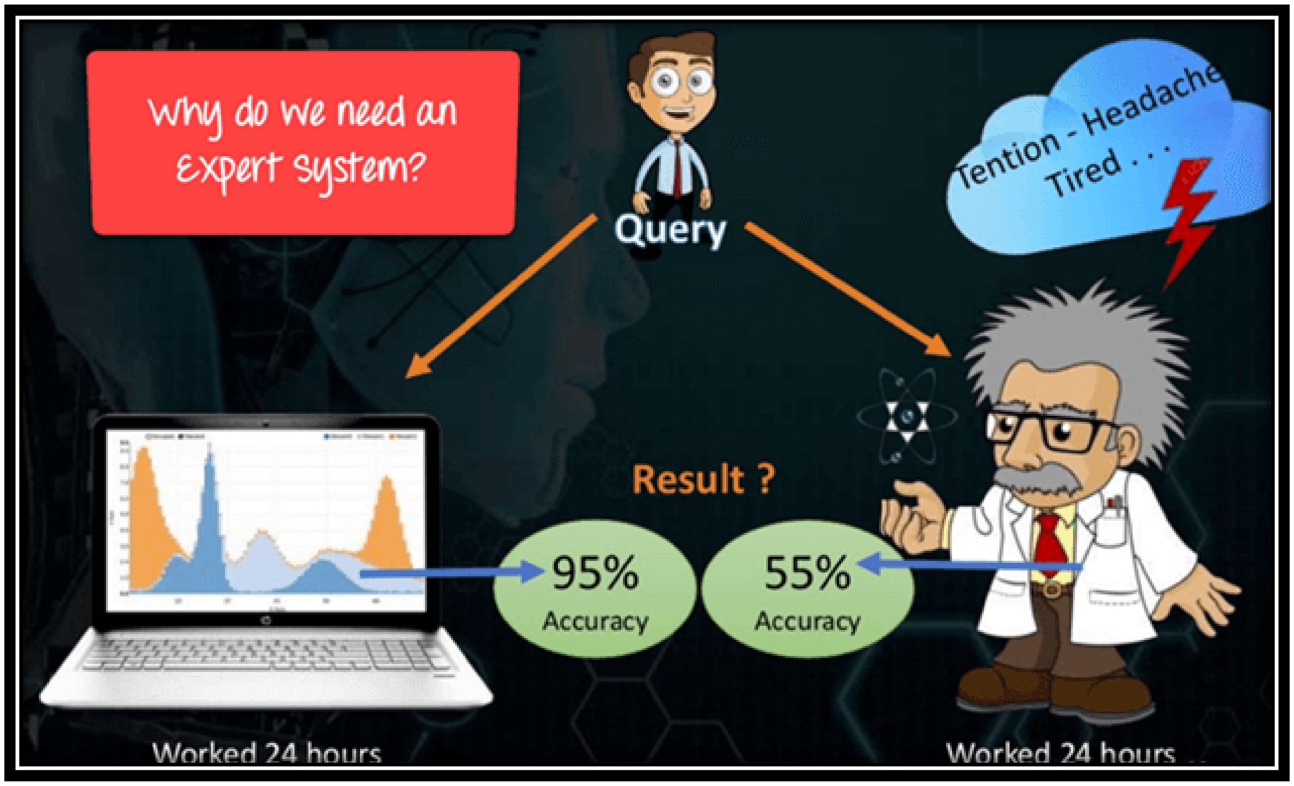
- she collected information from the internet for 10 mental disorders, treatments and their symptoms.
- Helped in giving base idea of how to approach decision tree.
- Created and finalized forward and backward decision tree.
- Draw decision tree of forward chaining and Backward Chaining using **Lucidchart.**
- Reviewed the IF – THEN rules for changes/ suggestions.
- Implemented Backward Chaining Function.
- Done the testing of Forward chaining function.
- Done the testing of backward chaining function.

3. Sahana Srinath:

- She collected information from the internet for 10 mental disorders, treatments and their symptoms.
- Helped in giving base idea of how to approach decision tree.
- Created and finalized forward and backward decision tree.
- Reviewed the IF – THEN rules for changes/ suggestions.
- Implemented Backward Chaining Function
- Done the testing of Forward chaining function.
- Done the testing of backward chaining function.

3. DOMAIN

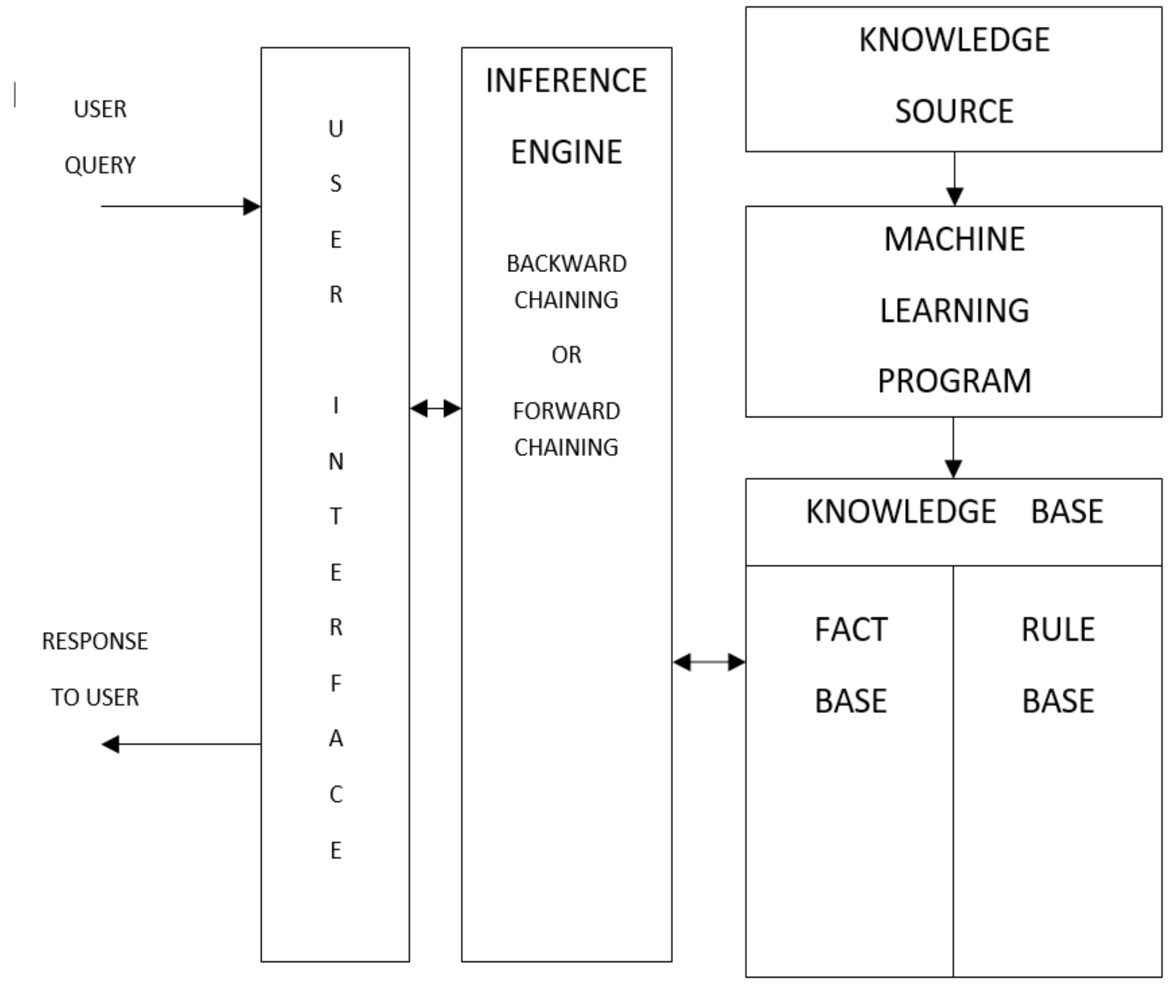
- The goal of the project is to develop an Intelligent Expert system for the staff of hospital to detect the mental disorder and to recommend treatment for the mental disorder based on the symptoms given by the patients.
- Generate meaningful Decision Tree and IF-THEN rules for both forward and backward chaining.
- From the rules, develop the program using data structures of Backward and Forward chaining principles.



4. EXPERT SYSTEM

An expert system is an AI software that uses knowledge stored in a knowledge base to solve problems that would usually require a human expert thus preserving a human expert's knowledge in its knowledge base.

4.1 Components of an expert system:



- **Knowledge Base** :
  Knowledge Base has two parts. In Fact, base, information is gathered from web or any experts in that field. In Rule base, the facts are transformed into IF-THEN rules.
- **Machine Learning** :
  We did not use Machine learning for this project.
- **Inference Engine**:
  The main functionality implies here. Backward Chaining to come to a decision and Forward chaining to provide the conclusion.
- **User Interface**:
  The interface to process user query and give response back to user.

4.2 Characteristics of an expert system:

- It helps to distribute the expertise of a human.
- One expert system may contain knowledge from more than one human expert thus making the solutions more efficient.
- It decreases the cost of consulting an expert for various domains such as medical diagnosis.
- They use a knowledge base and inference engine.
- Expert systems can solve complex problems by deducing new facts through existing facts of knowledge, represented mostly as if-then rules rather than through conventional procedural code.
- Expert systems were among the first truly successful forms of artificial intelligence (AI) software.
- Human experts are perishable, but an expert system is permanent.

4.3 Limitations of an expert system:

- Don't have human-like decision making power.
- Can't possess human capabilities.
- Can't produce correct result from less amount of knowledge.

4.4 Advantages of an expert system:

- Low accessibility cost.
- Fast response.
- Not affected by emotions unlike humans.
- Low error rate.

4.5 Disadvantages of an expert system:

- Expert system have no emotions.
- Common sense is the main issue of the expert system.
- It is developed for a specific domain.
- It needs to be updated manually. It does not learn itself.
- Not capable to explain the logic behind the decision.

5. DESIGN OF KNOWLEDGE BASE:

The knowledge base is a collection of rules or other information structures derived from the human expert. Rules are typically structured as If/Then statements of the form

IF <antecedent> THEN <consequent>

To represent knowledge effectively have been central to progress in various aspects of medical informatics. We collected information from the web to know about mental disorders, symptoms and their respective symptoms.

In our Project, Knowledge Base consists of:

- Symptoms
- Diseases

Here, few of the symptoms were common. One of the most common diagrams used in working out problems of this sort is the decision tree. So, we carefully created our decision tree First, by making sure that all the diseases will have at least one unique symptom. Then, we converted decision tree into IF-THEN rules.

In order to accomplish feats of apparent intelligence, an expert system relies on two components: a knowledge base and an inference engine. A knowledge base is an organized collection of facts about the system's domain. An inference engine interprets and evaluates the facts in the knowledge base in order to provide an answer. Typical tasks for expert systems involve classification, diagnosis, monitoring, design, scheduling, and planning for specialized endeavours.

Facts for a knowledge base must be acquired from human experts through interviews and observations. This knowledge is then usually represented in the form of "if-then" rules (production rules): "If some condition is true, then the following inference can be made (or some action taken)." The knowledge base of a major expert system includes thousands of rules.

5.1. Decision Tree

It is a very useful and effective type of diagram because it enables us to visualize all the factors that must be considered in reaching a decision and to see how one consideration leads to others, which then lead to still others, and so on.

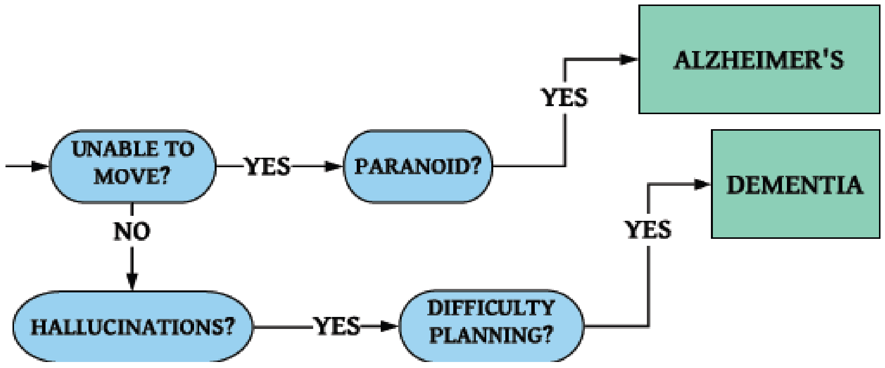
The decision tree is so named because it branches off just like a tree, and at the very end of each branch or system of branches is a conclusion, we use the decision tree to illustrate the problem clearly.

In decision analysis a decision tree and the closely related influence diagram are used as a visual and analytical decision support tool, where the expected values (or expected utility) of competing alternatives are calculated.

A decision tree consists of 3 types of nodes:

- Decision nodes - commonly represented by squares.
- Chance nodes - represented by circles.
- End nodes - represented by triangles.

Decision trees, influence diagrams, utility functions, and other decision analysis tools and methods are taught to undergraduate students in schools of business, health economics, and public health, and are examples of operations research or management science methods.



5.1.1 Explaining one of the scenario of the project:

In our case rectangles are the disorders which are the terminal nodes.
Here, First, expert system will as "Are you paranoid" k the question as, "Are you Unable to move? And if the user replies "YES" System will ask next question as disease is Alzheimer's . If user answer is "NO", again system will ask the question as, "Are you suffering from Hallucinations? If the User answer is "YES", Again system will ask the question as, "Do you find difficulty in planning? If the user answer is yes then the disease is dementia, if user answer is no it will ask for another symptom and diagnosis the diseases for all symptoms which are mentioned in the decision tree.

The diagram consists of circles and rectangles called "nodes". The arrow lines that connect these nodes are known as "arcs" or "branches." The circles which contain questions are "decision nodes." The rectangular shapes contain the goals of the diagram, and they signify conclusions. The arrow lines designate the direction of the diagram. Many of the nodes have branches leaving them, providing pathways to other nodes.

Each circular node contains a variable, and the paths are conditions placed upon the values of that variable. When we establish rules for this domain, these conditions will become clauses of the IF portion of an IF-THEN rule. The rectangles are conclusions or sub conclusions (Unable to move and Hallucinations). But it could also be a part of a path leading to another conclusion since it has a branch leaving its node. In that case, since the branch is not a condition, that is, it does not depend on anything and we can only emerge from that rectangle along that one branch , it is called a "sub conclusion" of another goal. A sub conclusion is also a clause in an IF statement.

Now we have carefully and thoroughly diagrammed the problem into decision tree. In the same way we collected data for 27 diseases along with their symptoms and treatments then we designed the decision tree for the project by keeping one symptom unique to each disease.

As computer does not understand decision trees, we need to convert the decision tree into set of rules that the computer can understand.

5.2 Conversion of decision tree to IF-THEN Rules :

IF-THEN rules are made up of two parts.
1. The IF part is comprised of conditions called clauses connected to one another with words known as logical operators such as AND, OR, and NOT.
2. The THEN part is evaluated only if the IF part is true.
The combination of linked decision nodes (circles) and a conclusion node (rectangle) represents an IF-THEN rule. The IF part contains all the decision nodes in the path leading to a conclusion node; each decision node in the path leading to the conclusion contributes one clause to the IF portion. The conclusion itself forms the THEN portion.

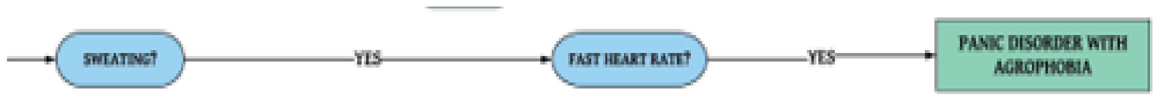


Fig: Conversion of Tree to Rule

As stated, before decision tree is a predictive model which maps observations about an item to conclusions about the item's target value. So, decision tree may have multiple paths, but all paths will be separated based on the values of nodes in terms of YES or NO. Meaning, if value node 1 = yes then go to path 1 else go to path2.
The decision tree can be linearized into decision rules, where the outcome is the contents of the leaf node, and the conditions along the path form a conjunction in the if clause.
In general, the rules have the form:

if *condition1* and *condition2* and *condition3* then outcome.

So, we can create the rule from above decision tree as follows:

IF SWEATING = YES && FAST_HEART_RATE = YES THEN DIAGNOSIS = PANIC DISORDER WITH AGROPHOBIA.

Then, we generated rules for each possible conclusion which is done according to the following technique.
1. Choose a conclusion (rectangular) node of the decision tree. Record it.
2. Choose a decision node with a connecting branch to the *left* of the node you chose in step 1. Record that node.
3. Continue doing step 2 until either there are no more nodes to the left or until a conclusion node is reached. If a conclusion (rectangular) node is reached, record it and stop. If there are no more nodes, stop.
4. Each decision node in the path is the variable part of an IF clause. The value associated with the branch is the condition. All connected clauses use the logical AND.
5. The conclusion becomes the THEN part.
The decision tree comes in handy-it provides a simple and expedient method for establishing the rules that comprise the knowledge base. Without the knowledge base we cannot go forward.

5.3 Backward Chaining Decision Tree:

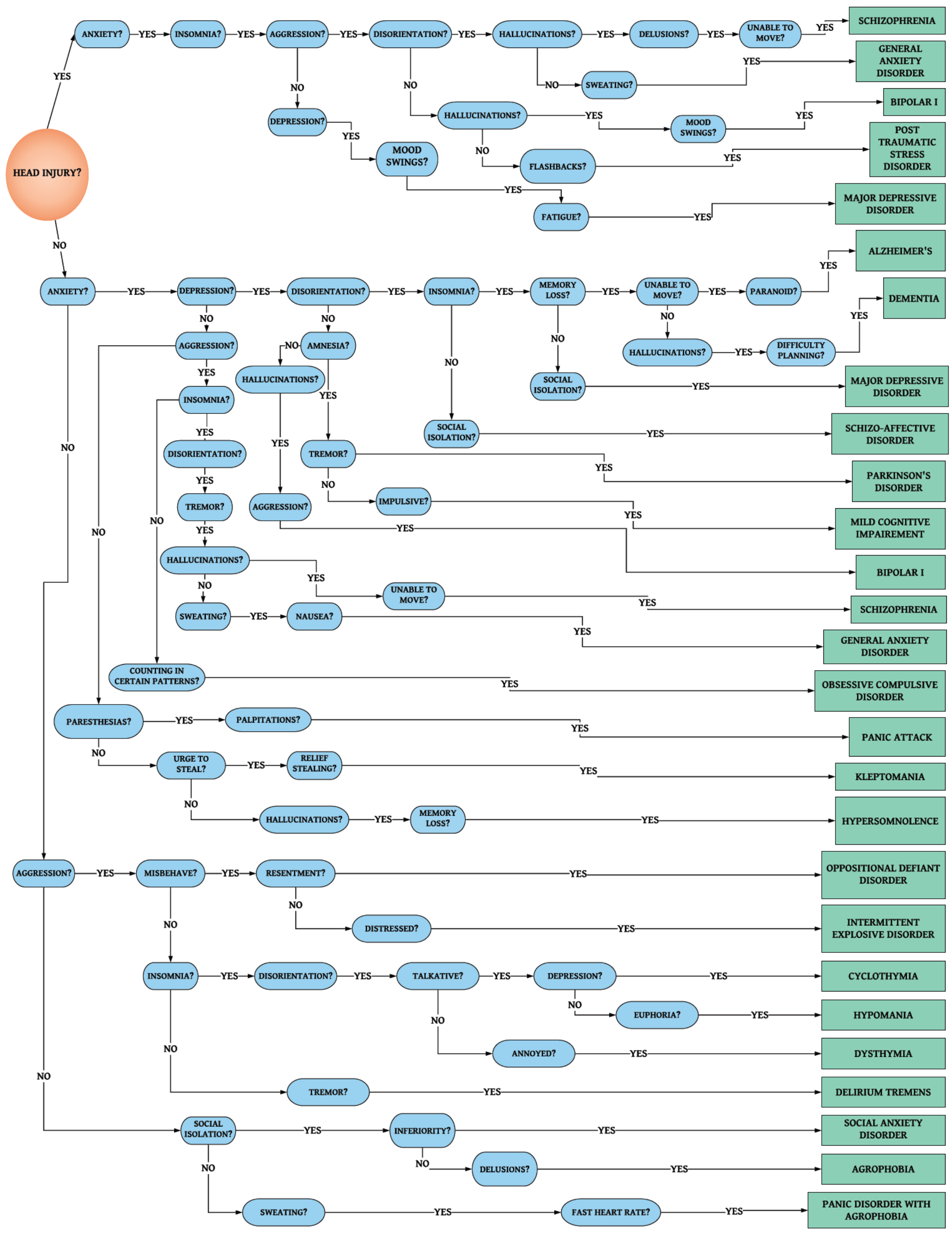


Figure : Backward Chaining Decision Tree

5.4 Knowledge Base Backward Chaining:

10	IF HEAD_INJURY = YES && ANXIETY = YES && INSOMNIA = YES && AGGRESSION = YES && DISORIENTATION = YES && HALLUCINATIONS = YES && DELUSIONS = YES && UNABLE_TO_MOVE = YES THEN DIAGNOSIS = Schizophrenia
20	IF HEAD_INJURY = YES && ANXIETY = YES && INSOMNIA = YES && AGGRESSION = YES && DISORIENTATION = YES && HALLUCINATIONS = NO && SWEATING = YES THEN DIAGNOSIS = Generalized Anxiety Disorder
30	IF HEAD_INJURY = YES && ANXIETY = YES && INSOMNIA = YES && AGGRESSION = YES && DISORIENTATION = NO && HALLUCINATIONS = YES && MOOD_SWINGS = YES THEN DIAGNOSIS = Bipolar 1
40	IF HEAD_INJURY = YES && ANXIETY = YES && INSOMNIA = YES && AGGRESSION = YES && DISORIENTATION = NO && HALLUCINATIONS = NO && FLASHBACKS = YES THEN DIAGNOSIS = Post Traumatic Stress Disorder
50	IF HEAD_INJURY = YES && ANXIETY = YES && INSOMNIA = YES && AGGRESSION = NO && DEPRESSION = YES && MOOD_SWINGS = YES && FATIGUE = YES THEN DIAGNOSIS = Major Depressive Disorder
60	IF HEAD_INJURY = NO && ANXIETY = YES && DEPRESSION = YES && DISORIENTATION = YES && INSOMNIA = YES && MEMORY_LOSS = YES && UNABLE_TO_MOVE = YES && PARANOID = YES THEN DIAGNOSIS = Alzheimers
70	IF HEAD_INJURY = NO && ANXIETY = YES && DEPRESSION = YES && DISORIENTATION = YES && INSOMNIA = YES && MEMORY_LOSS = YES && UNABLE_TO_MOVE = NO && HALLUCINATIONS = YES && DIFFICULTY_PLANNING = YES THEN DIAGNOSIS = Dementia
80	IF HEAD_INJURY = NO && ANXIETY = YES && DEPRESSION = YES && DISORIENTATION = YES && INSOMNIA = YES && MEMORY_LOSS = NO && SOCIAL_ISOLATION = YES THEN DIAGNOSIS = Major Depressive Disorder
90	IF HEAD_INJURY = NO && ANXIETY = YES && DEPRESSION = YES && DISORIENTATION = YES && INSOMNIA = NO && SOCIAL_ISOLATION = YES THEN DIAGNOSIS = Schizoaffective Disorder
100	IF HEAD_INJURY = NO && ANXIETY = YES && DEPRESSION = YES && DISORIENTATION = NO && AMNESIA = YES && TREMOR = YES THEN DIAGNOSIS = Parkinson's Disorder
110	IF HEAD_INJURY = NO && ANXIETY = YES && DEPRESSION = YES && DISORIENTATION = NO && AMNESIA = NO && HALLUCINATIONS = YES && AGGRESSION = YES THEN DIAGNOSIS = Bipolar 1"
120	IF HEAD_INJURY = NO && ANXIETY = YES && DEPRESSION = YES && DISORIENTATION = NO && AMNESIA = YES && TREMOR = NO && IMPULSIVE = YES THEN DIAGNOSIS = Mild Cognitive Impairment
130	IF HEAD_INJURY = NO && ANXIETY = YES && DEPRESSION = NO && AGGRESSION = YES && INSOMNIA = YES && DISORIENTATION = YES && TREMOR = YES && HALLUCINATIONS = YES && UNABLE_TO_MOVE = YES THEN DIAGNOSIS = Schizophrenia

140	IF HEAD_INJURY = NO && ANXIETY = YES && DEPRESSION = NO && AGGRESSION = YES && INSOMNIA = YES && DISORIENTATION = YES && TREMOR = YES && HALLUCINATIONS = NO && SWEATING = YES && NAUSEA = YES THEN DIAGNOSIS = Generalized Anxiety Disorder
150	IF HEAD_INJURY = NO && ANXIETY = YES && DEPRESSION = NO && AGGRESSION = YES && INSOMNIA = NO && COUNTING_IN_PATTERN = YES THEN DIAGNOSIS = Obsessive Compulsive Disorder
160	IF HEAD_INJURY = NO && ANXIETY = YES && DEPRESSION = NO && AGGRESSION = NO && PARESTHESIAS = YES && PALPITATIONS = YES THEN DIAGNOSIS = Panic Attack
170	IF HEAD_INJURY = NO && ANXIETY = YES && DEPRESSION = NO && AGGRESSION = NO && PARESTHESIAS = NO && URGE_TO_STEAL = YES && RELIEF_STEALING = YES THEN DIAGNOSIS = Kleptomania
180	IF HEAD_INJURY = NO && ANXIETY = YES && DEPRESSION = NO && AGGRESSION = NO && PARESTHESIAS = NO && URGE_TO_STEAL = NO && HALLUCINATIONS = YES && MEMORY_LOSS = YES THEN DIAGNOSIS = Hypersomnolence
190	IF HEAD_INJURY = NO && ANXIETY = NO && AGGRESSION = YES && MISBEHAVE = YES && RESENTMENT = YES THEN DIAGNOSIS = Oppositional Defiant Disorder
200	IF HEAD_INJURY = NO && ANXIETY = NO && AGGRESSION = YES && MISBEHAVE = YES && RESENTMENT = NO && DISTRESSED = YES THEN DIAGNOSIS = Intermittent Explosive Disorder
210	IF HEAD_INJURY = NO && ANXIETY = NO && AGGRESSION = YES && MISBEHAVE = NO && INSOMNIA = YES && DISORIENTATION = YES && TALKATIVE = YES && DEPRESSION = YES THEN DIAGNOSIS = Cyclothymia
220	IF HEAD_INJURY = NO && ANXIETY = NO && AGGRESSION = YES && MISBEHAVE = NO && INSOMNIA = YES && DISORIENTATION = YES && TALKATIVE = YES && DEPRESSION = NO && EUPHORIA = YES THEN DIAGNOSIS = Hypomania
230	IF HEAD_INJURY = NO && ANXIETY = NO && AGGRESSION = YES && MISBEHAVE = NO && INSOMNIA = YES && DISORIENTATION = YES && TALKATIVE = NO && ANNOYED = YES THEN DIAGNOSIS = Dysthymia
240	IF HEAD_INJURY = NO && ANXIETY = NO && AGGRESSION = YES && MISBEHAVE = NO && INSOMNIA = NO && TREMOR = YES THEN DIAGNOSIS = Delirium Tremens
250	IF HEAD_INJURY = NO && ANXIETY = NO && AGGRESSION = NO && SOCIAL_ISOLATION = YES && INFERIORITY = YES THEN DIAGNOSIS = Social Anxiety Disorder
260	IF HEAD_INJURY = NO && ANXIETY = NO && AGGRESSION = NO && SOCIAL_ISOLATION = YES && INFERIORITY = NO && DELUSIONS = YES THEN DIAGNOSIS = Agrophobia
270	IF HEAD_INJURY = NO && ANXIETY = NO && AGGRESSION = NO && SOCIAL_ISOLATION = NO && FAST_HEART_RATE = YES && SWEATING = YES THEN DIAGNOSIS = Panic Disorder with Agrophobia

5.5 Forward Chaining Decision Tree:

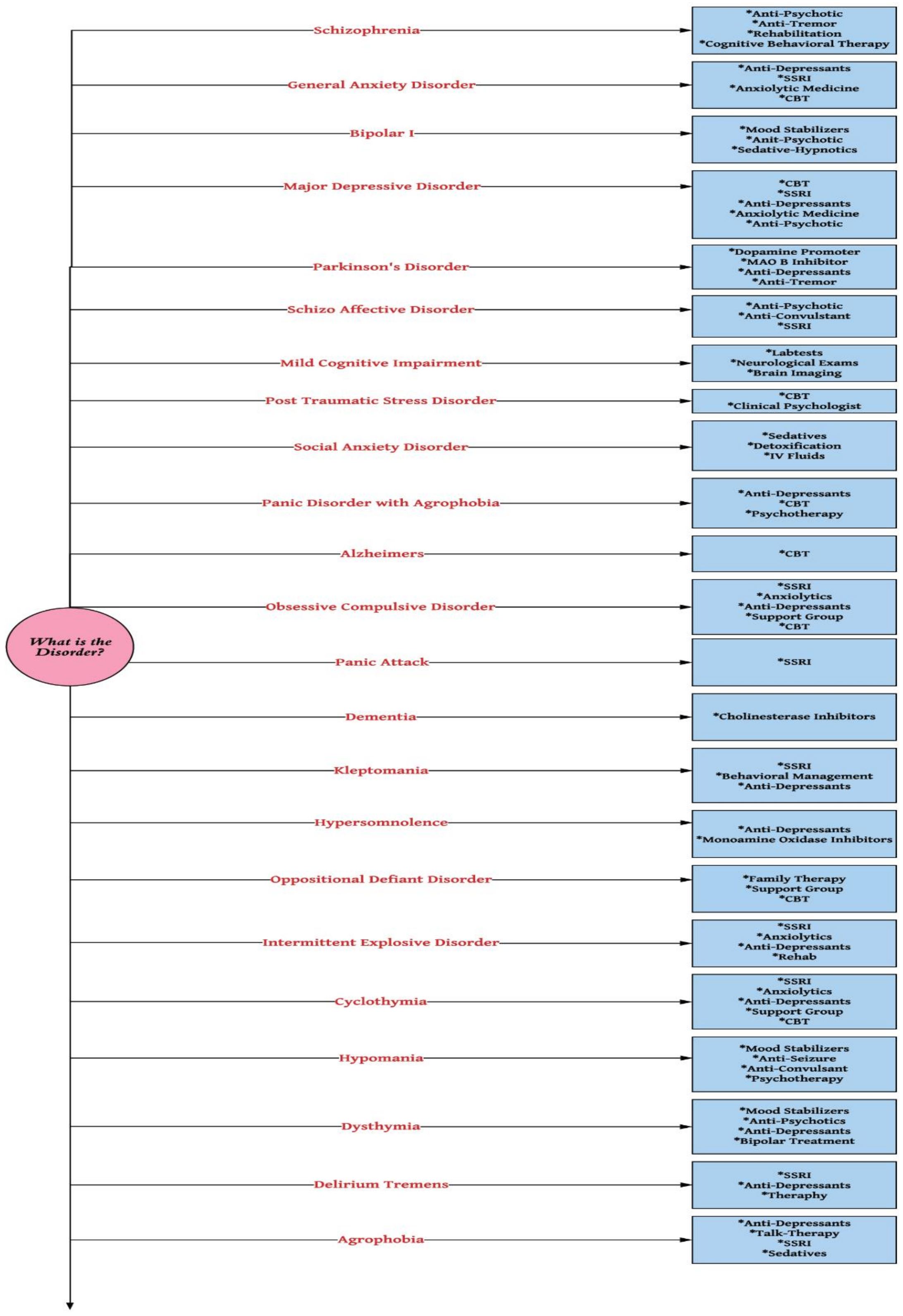

What is the Disorder?
Schizophrenia
*Anti-Psychotic *Anti-Tremor *Rehabilitation *Cognitive Behavioral Therapy
General Anxiety Disorder
*Anti-Depressants *SSRI *Anxiolytic Medicine *CBT
Bipolar I
*Mood Stabilizers *Anit-Psychotic *Sedative-Hypnotics
Major Depressive Disorder
*CBT *SSRI *Anti-Depressants *Anxiolytic Medicine *Anti-Psychotic
Parkinson's Disorder
*Dopamine Promoter *MAO B Inhibitor *Anti-Depressants *Anti-Tremor
Schizo Affective Disorder
*Anti-Psychotic *Anti-Convulstant *SSRI
Mild Cognitive Impairment
*Labtests *Neurological Exams *Brain Imaging
Post Traumatic Stress Disorder
*CBT *Clinical Psychologist
Social Anxiety Disorder
*Sedatives *Detoxification *IV Fluids
Panic Disorder with Agrophobia
*Anti-Depressants *CBT *Psychotherapy
Alzheimers
*CBT
Obsessive Compulsive Disorder
*SSRI *Anxiolytics *Anti-Depressants *Support Group *CBT
Panic Attack
*SSRI
Dementia
*Cholinesterase Inhibitors
Kleptomania
*SSRI *Behavioral Management *Anti-Depressants
Hypersomnolence
*Anti-Depressants *Monoamine Oxidase Inhibitors
Oppositional Defiant Disorder
*Family Therapy *Support Group *CBT
Intermittent Explosive Disorder
*SSRI *Anxiolytics *Anti-Depressants *Rehab
Cyclothymia
*SSRI *Anxiolytics *Anti-Depressants *Support Group *CBT
Hypomania
*Mood Stabilizers *Anti-Seizure *Anti-Convulsant *Psychotherapy
Dysthymia
*Mood Stabilizers *Anti-Psychotics *Anti-Depressants *Bipolar Treatment
Delirium Tremens
*SSRI *Anti-Depressants *Theraphy
Agrophobia
*Anti-Depressants *Talk-Therapy *SSRI *Sedatives

Figure : Forward Chaining Decision Tree

5.6 Knowledge Base Forward Chaining

10	IF DIAGNOSIS=Schizophrenia THEN TREATMENT= Anit-Psychotic, Anti-Tremor, Rehabilitation and Cognitive Behavioral Therapy but medications can also be prescribed which Improves mental function, lowers blood pressure, and may balance mood. The medications are RazadyneÆ (galantamine), ExelonÆ (rivastigmine), and AriceptÆ (donepezil).
20	IF DIAGNOSIS=General_Anxiety_Disorder THEN TREATMENT=Treatment consists of Anti-Depressants, SSRI, Anxiolytic Medicine and CBT. Medications to increase dopamine, Medications can help control the symptoms of Parkinson's.
30	IF DIAGNOSIS=Bipolar_I THEN TREATMENT= Mood Stabilizers, Anit-Psychotic and Sedative-Hypnotics.Your doctor may prescribe paroxetine (Paxil) or sertraline (Zoloft).
40	IF DIAGNOSIS=Post_Traumatic_Stress Disorder THEN TREATMENT= Treatment is usually lifelong and often involves a combination of medications, psychotherapy with a Clinical Psychologist and CBT. Medications like Chlorpromazine, Haloperidol, Fluphenazine and benztropine will also help.
50	IF DIAGNOSIS=Major_Depressive_Disorder THEN TREATMENT= Treatment includes antipsychotic drug paliperidone (Invega). CBT,Reuptake Inhibitor (SSRI)To ease depressed mood and anxiety.Can take Chlorpromazine,Haloperidol,Fluphenazine along with anticonvulsant medication like carbamazepine,Valproic Acid.
60	IF DIAGNOSIS=Alzheimers THEN TREATMENT=The mainstay of treatment is usually CBT,talk therapy, or a combination of the two. Increasingly, research suggests these treatments may normalize brain changes associated with depression.
70	IF DIAGNOSIS=Dementia THEN TREATMENT= Major treatment is Cholinesterase Inhibitors. Medications may include:lithium (Lithobid), valproic acid (Depakene), divalproex sodium (Depakote), carbamazepine (Tegretol, Equetro, others) and lamotrigine (Lamictal).
80	IF DIAGNOSIS=Schizo_Affective_Disorder THEN TREATMENT= Anti-Convulsant and Antipsychotic drugs, such as aripiprazole (Abilify), cariprazine (Vraylar), quetiapine (Seroquel), asenapine (Saphris)
90	IF DIAGNOSIS=Parkinson's Disorder THEN TREATMENT= Dopamine Promoters, MAO B Inhibitors, antipsychotic medications, or antidepressants like Chlorpromazine, Haloperidol, (Depakote), carbamazepine. Such medications usually need to be taken daily and regularly to be effective.
100	IF DIAGNOSIS=Mild_Cognitive_Impairment THEN TREATMENT= Treatment involves Lab tests, Neurological Exams, Brain Imaging for further analysis by Consulting Doctor. Life style changes, coping and support from family will help.

110	IF DIAGNOSIS = Obsessive Compulsive Disorder THEN TREATMENT = Anxiolytics, antidepressants, in particular selective serotonin reuptake inhibitors (SSRIs)† mood† stabilizers, including lithium, valproic acid, and carbamazepine. Support Group and CBT treatment will also help
120	IF DIAGNOSIS=Panic_Attack THEN TREATMENT= Certain antidepressants called selective serotonin reuptake inhibitors (SSRIs), such as fluoxetine (Prozac) and sertraline (Zoloft), are used for the treatment
130	IF DIAGNOSIS=Kleptomania THEN TREATMENT=Medications like SSRI,Anti-depressants like Sertraline,Citalopram,Fluoxetine, Paroxetine, Diazepam, Buspirone, Alprazolam,Lorazepam,Clonazepam will help. Behavioral Management is needed
140	IF DIAGNOSIS=Hypersomnolence THEN TREATMENT= Treatments include Anti-Depressants, Monoamine Oxidase Inhibitor,SSRI, Antidepressant,Sertraline,Citalopram,Fluoxetine, BupropionVenlafaxineTrazodone medications can be prescribed
150	IF DIAGNOSIS=Oppositional Defiant Disorder THEN TREATMENT= Treatment may include counseling/Family Therapy, Support Group and CBT
160	IF DIAGNOSIS=Intermittent_Explosive_Disorder THEN TREATMENT= SSRI, Mood Stabilizers, Anti-Depressants and Rehab will help.
170	IF DIAGNOSIS=Cyclothymia THEN TREATMENT= SSRI, Anxiolytics, Anti-Depressants, Support Group, CBT Treatment is normally provided to treat this disorder.
180	IF DIAGNOSIS=Hypomania THEN TREATMENT= Mood Stabilizers, Anti-Seizure, Anti-Convulsant and Psychotherapy can be given to the patient
190	IF DIAGNOSIS=Dysthymia THEN TREATMENT= Mood Stabilizers, Anti-Psychotics, Anti-Depressants and May also include similar treatment as Bipolar Disorder
200	IF DIAGNOSIS=Delirium_Tremens THEN TREATMENT= SSRI treatment, Anti-depressant Drugs and therapy will help.Medications like lithium, valproic acid, and carbamazepine can be consumed by Patient
210	IF DIAGNOSIS=Social_Anxiety_Disorder THEN TREATMENT= Sedatives, Detoxification and IV Fluids can help reduce anxiety and withdrawal symptoms
220	IF DIAGNOSIS=Agrophobia THEN TREATMENT= Anti-Depressants, Talk-therapy, SSRI and Sedatives. Medications like Sertraline,Citalopram,Fluoxetine is normally prescribed
230	IF DIAGNOSIS=Panic_Disorder_with_Agrophobia THEN TREATMENT= Patient may be given CBT, Psychotherapy AND Anti- Depressants include divalproex sodium (Depakote), lamotrigine (Lamictal), and valproic acid (Depakene) Treatment usually involves counseling and therapy. In rare cases, medications may be used

6. INFERENCE ENGINE

The brain of an expert system is the inference engine that provides a methodology for reasoning about information in the knowledge base.
The inference engine is the main processing element of the expert system. The inference engine chooses rules from the agenda to fire. If there are no rules on the agenda, the inference engine must obtain information from the user in order to add more rules to the agenda. It makes use of knowledge base, in order to draw conclusions for situations. It is responsible for gathering the information from the user, by asking various questions and applying it wherever necessary. It seeks information and relationships from the knowledge base and to provide answers, predictions and suggestions the way a human expert would.
An inference engine interprets and evaluates the facts in the knowledge base in order to provide an answer. Typical tasks for expert systems involve classification, diagnosis, monitoring and design.
The inference engine may also include capabilities for explanation, so that it can explain to a user the chain of reasoning used to arrive at a specific conclusion by tracing back over the firing of rules that resulted in assertion.

There are primarily two modes for an inference engine:

- Backward chaining.
- Forward chaining

7. METHODOLOGIES

7.1 Backward Chaining :

The backward-chaining algorithm, as its name suggests, works backward from the query. If the query q is known to be true, then no work is needed. Otherwise, the algorithm finds those implications in the knowledge base whose conclusion is q. If all the premises of one of those implications can be proved true (by backward chaining), then q is true. As with forward chaining, an efficient implementation runs in linear time. GOAL-DIRECTED Backward chaining is a form of goal-directed reasoning.

7.1.1 Describing the algorithm:

1. Identify the conclusion.
2. Search the conclusion list for the first instance of the conclusion's name. If found, place the rule on the conclusion stack using the rule number and a (1) to represent the clause number.

    Formula : CLAUSE NUMBER = 11* (RULE NUMBER / 10 - 1) + 1;

    If not found, notify the user that an answer cannot be found.

3. Instantiate the IF clause (i.e., each condition variable) of the statement.
4. If one of the IF clause variables is not instantiated, as indicated by the variable list, and is not a conclusion variable, that is, not on the conclusion list, ask the user to enter a value.
5. If one of the clauses is a conclusion variable, place the conclusion variable's rule number on the top of the stack and go back to step 3.
6. If the statement on top of the stack cannot be instantiated using the present IF-THEN statement, remove the unit from the top of the stack and search the conclusion list for another instance of that conclusion variable's name.
7. If such a statement is found, go back to step 3.
8. If there are no more conclusions left on the conclusion stack with that name, the rule for the previous conclusion is false. If there is no previous conclusion, then notify the user that an answer cannot be found. If there is a previous conclusion, go back to step 6.
9. If the rule on top of the stack can be instantiated, remove it from the stack. If another conclusion variable is underneath, increment the clause number, and for the remaining clauses go back to step 3. If no other conclusion variable is underneath, we have found the disorder for user entered symptoms. The user can start the treatment.

7.1.2 Advantages of backward chaining :

1) The system will stop processing once the variable has its value. It's a “floor system”.
2) The system that uses backward chaining tries to set goals in order which they arrive in the knowledge base.
3) The search in backward chaining is directed.
4) While searching, the backward chaining considers those parts of the knowledge base which are directly related to the considered problem or backward chaining never performs unnecessary inferences.
5) Backward chaining is an excellent tool for specific types of problems such as diagnosing and debugging.
6) Compare to forward chaining, few data are asked, but many rules are searched.

7.1.3 Disadvantages of backward chaining:
1) The goal must be known to perform the backward chaining process.
2) The implementation process of backward chaining is difficult.

7.2 Forward Chaining :

Forward chaining starts from the set of facts (the available data) and then checks if the given rules are satisfied. If so (there is a matching) then the rule is executed (fires). This process continues until the goal is found (assuming that just one answer is required), or there are no new facts to be added.

The rules are expressed in a close form to human language but in reality the rule based system must be represented in a way that the limited machine-processes can understand. There is no standard syntax rule. LHS of the rule consists of object name followed by the objects attributes and values.

7.2.2 Describing the algorithm:

1. The condition is identified.
2. The condition variable is placed on the conclusion variable queue and its value is marked on the variable list.
3. The clause variable list is searched for the variable whose name is the same as the one in the front of the queue. If found, the rule number and a 1 are placed into the clause variable pointer. If not found, go to step 6.

Formula : Rule_no= (((Clause_no / 3) +1) * 10).

4. Each variable in the IF clause of the rule that is not already instantiated is now instantiated. The variables are in the clause variable list. If all the clauses are true, the THEN part is invoked.
5. The instantiated THEN part of the variable is placed in the *back* of the conclusion variable queue.
6. When there are no more IF statements containing the variable that is at the *front* of the conclusion variable queue, that variable is removed.
7. If there are no more variables on the conclusion variable queue, end the session. If there are more variables, go to step 3.

7.2.3 Advantages of forward chaining :

1) Runs great when a problem naturally begins by collecting data and searching for information that can be collected from it to be used in future steps.
2) Forward chaining has the capability of providing a lot of data from the available few initial data or facts.
3) Forward chaining is a very popular technique for implementation to expert systems, and systems using production rules in the knowledge base. For the expert system that needs interruption, control, monitoring, and planning, the forward chaining is the best choice.
4) When there are few facts and initial states, the forward chaining is very useful to be applied.

7.2.4. Disadvantages of a forward chaining:

1) New information will be generated by the inference engine without any knowledge about which information will be used for reaching the goal.
2) The user might be asked to enter a lot of inputs without knowing which input is relevant to the conclusion.
3) Several rules may fire that have nothing to reach the goal.
4) It might produce different conclusions which are the causes of a high cost of the chaining process.

8. PROGRAM IMPLEMENTATION

A Backward chain object is then is instantiated by loading the data into the data structures used in the backward chaining. The Backward chain object executes and returns the Disease. A Backward chain Object is instantiated by loading the data into the forward chaining data structures. The Forward Chain object then executes by passing data to it that was retuned by the Backward Chain start method call

8.1 Data structures needed for backward chaining:

8.1.1 Variable List :

We implemented variable list using Structure called variable which contains varname(variable name), varstatus(variable status), varvalue(variable value).

```
struct variable
{
    string varname;
    string varstatus;
    string varvalue;
};
```

Varname: is the name of each variable contained in the IF part of the knowledge base rules.

Varstatus: initially varstatus in the INSTANTIATION column is always set to NI(Not Instantiated). It will be changed to instantiated(I) when each variable is set to a value.
A variable appears atmost once in the list no matter how many times it appears in the clauses.

Varvalue : is the string value which consists of either “YES” or “NO” which is choosed by the user.

We used following variables in the project:

HEAD_INJURY	**NI**
ANXIETY	**NI**
INSOMNIA	**NI**
AGGRESSION	**NI**
DISORIENTATION	**NI**
HALLUCINATIONS	**NI**
DELUSIONS	**NI**
UNABLE_TO_MOVE	**NI**
SWEATING	**NI**
MOOD_SWINGS	**NI**
FLASHBACKS	**NI**
DEPRESSION	**NI**

FATIGUE	NI
MEMORY_LOSS	NI
PARANOID	NI
DIFFICULTY_PLANNING	NI
SOCIAL_ISOLATION	NI
AMNESIA	NI
TREMOR	NI
IMPULSIVE	NI
COUNTING_IN_PATTERN	NI
PARESTHESIAS	NI
PALPITATIONS	NI
URGE_TO_STEAL	NI
RELIEF_STEALING	NI
MISBEHAVE	NI
RESENTMENT	NI
DISTRESSED	NI
TALKATIVE	NI
EUPHORIA	NI
ANNOYED	NI
INFERIORITY	NI
FAST_HEART_RATE	NI

This variable list will be stored in the text file and is read from array of structures of type variable.

```
variable variableList[33];
```

8.1.2 Clause variable List :

In our project, Clause variables List stores all the variables which are used in the IF part of the rules. Clause variables are stored in an array with 11 slots allocated to each slot as we have 10 maximum symptoms or IF clause variables to diagnosis each disease, one slot is left intentionally to avoid confusion. If the clause variables are less than 10 and 1 extra blank space is written as "b" instead of blank space.

When all the IF part clauses in the rule are connected by "AND" logical operator, the clause variables corresponding to the respective rule will be instantiated before executing the THEN part.

Example:

Consider Rule_no is 10, then the clause_no for the rule number is:
11 * ((10/10)-1) + 1 = 1.

We used following clause variables in the project:

1	HEAD_INJURY
2	ANXIETY
3	INSOMNIA
4	AGGRESSION
5	DISORIENTATION

6	HALLUCINATIONS
7	DELUSIONS
8	UNABLE_TO_MOVE
9	b
10	b
11	b
12	HEAD_INJURY
13	ANXIETY
14	INSOMNIA
15	AGGRESSION
16	DISORIENTATION
17	HALLUCINATIONS
18	SWEATING
19	b
20	b
21	b
22	b
23	HEAD_INJURY
24	ANXIETY
25	INSOMNIA
26	AGGRESSION
27	DISORIENTATION
28	HALLUCINATIONS
29	MOOD_SWINGS
30	b
31	b
32	b
33	b
34	HEAD_INJURY
35	ANXIETY
36	INSOMNIA
37	AGGRESSION
38	DISORIENTATION
39	HALLUCINATIONS
40	FLASHBACKS
41	b
42	b
43	b
44	b
45	HEAD_INJURY
46	ANXIETY
47	INSOMNIA
48	AGGRESSION
49	DEPRESSION
50	MOOD_SWINGS
51	FATIGUE
52	b
53	b
54	b
55	b
56	HEAD_INJURY
57	ANXIETY
58	DEPRESSION
59	DISORIENTATION
60	INSOMNIA
61	MEMORY_LOSS
62	UNABLE_TO_MOVE
63	PARANOID
64	b

65	**b**
66	**b**
67	**HEAD_INJURY**
68	**ANXIETY**
69	**DEPRESSION**
70	**DISORIENTATION**
71	**INSOMNIA**
72	**MEMORY_LOSS**
73	**UNABLE_TO_MOVE**
74	**HALLUCINATIONS**
75	**DIFFICULTY_PLANNING**
76	**b**
77	**b**
78	**HEAD_INJURY**
79	**ANXIETY**
80	**DEPRESSION**
81	**DISORIENTATION**
82	**INSOMNIA**
83	**MEMORY_LOSS**
84	**SOCIAL_ISOLATION**
85	**b**
86	**b**
87	**b**
88	**b**
89	**HEAD_INJURY**
90	**ANXIETY**
91	**DEPRESSION**
92	**DISORIENTATION**
93	**INSOMNIA**
94	**SOCIAL_ISOLATION**
95	**b**
96	**b**
97	**b**
98	**b**
99	**b**
100	**HEAD_INJURY**
101	**ANXIETY**
102	**DEPRESSION**
103	**DISORIENTATION**
104	**AMNESIA**
105	**TREMOR**
106	**b**
107	**b**
108	**b**
109	**b**
110	**b**
111	**HEAD_INJURY**
112	**ANXIETY**
113	**DEPRESSION**
114	**DISORIENTATION**
115	**AMNESIA**
116	**HALLUCINATIONS**
117	**AGGRESSION**
118	**b**
119	**b**
120	**b**
121	**b**

122	HEAD_INJURY
123	ANXIETY
124	DEPRESSION
125	DISORIENTATION
126	AMNESIA
127	TREMOR
128	IMPULSIVE
129	b
130	b
131	b
132	b
133	HEAD_INJURY
134	ANXIETY
135	DEPRESSION
136	AGGRESSION
137	INSOMNIA
138	DISORIENTATION
139	TREMOR
140	HALLUCINATIONS
141	UNABLE_TO_MOVE
142	b
143	b
144	HEAD_INJURY
145	ANXIETY
146	DEPRESSION
147	AGGRESSION
148	INSOMNIA
149	DISORIENTATION
150	TREMOR
151	HALLUCINATIONS
152	SWEATING
153	b
154	b
155	HEAD_INJURY
156	ANXIETY
157	DEPRESSION
158	AGGRESSION
159	INSOMNIA
160	COUNTING_IN_PATTERN
161	b
162	b
163	b
164	b
165	b
166	HEAD_INJURY
167	ANXIETY
168	DEPRESSION
169	AGGRESSION
170	PARESTHESIAS
171	PALPITATIONS
172	b
173	b
174	b
175	b
176	b
177	HEAD_INJURY
178	ANXIETY

179	DEPRESSION
180	AGGRESSION
181	PARESTHESIAS
182	URGE_TO_STEAL
183	RELIEF_STEALING
184	b
185	b
186	b
187	b
188	HEAD_INJURY
189	ANXIETY
190	DEPRESSION
191	AGGRESSION
192	PARESTHESIAS
193	URGE_TO_STEAL
194	HALLUCINATIONS
195	MEMORY_LOSS
196	b
197	b
198	b
199	HEAD_INJURY
200	ANXIETY
201	AGGRESSION
202	MISBEHAVE
203	RESENTMENT
204	b
205	b
206	b
207	b
208	b
209	b
210	HEAD_INJURY
211	ANXIETY
212	AGGRESSION
213	MISBEHAVE
214	RESENTMENT
215	DISTRESSED
216	b
217	b
218	b
219	b

220	b
221	HEAD_INJURY
222	ANXIETY
223	AGGRESSION
224	MISBEHAVE
225	INSOMNIA
226	DISORIENTATION
227	TALKATIVE
228	DEPRESSION
229	b
230	b
231	b
232	HEAD_INJURY
233	ANXIETY
234	AGGRESSION
235	MISBEHAVE
236	INSOMNIA
237	DISORIENTATION
238	TALKATIVE
239	DEPRESSION
240	EUPHORIA
241	b
242	b
243	HEAD_INJURY
244	ANXIETY
245	AGGRESSION
246	MISBEHAVE
247	INSOMNIA
248	DISORIENTATION
249	TALKATIVE
250	ANNOYED
251	b
252	b
253	b
254	HEAD_INJURY
255	ANXIETY
256	AGGRESSION
257	MISBEHAVE
258	INSOMNIA
259	TREMOR
260	b

261	b
262	b
263	b
264	b
265	HEAD_INJURY
266	ANXIETY
267	AGGRESSION
268	SOCIAL_ISOLATION
269	INFERIORITY
270	b
271	b
272	b
273	b
274	b
275	b
276	HEAD_INJURY
277	ANXIETY
278	AGGRESSION
279	SOCIAL_ISOLATION
280	INFERIORITY
281	DELUSIONS
282	b
283	b
284	b
285	b
286	b
287	HEAD_INJURY
288	ANXIETY
289	AGGRESSION
290	SOCIAL_ISOLATION
291	FAST_HEART_RATE
292	SWEATING
293	b
294	b
295	b
296	b
297	b

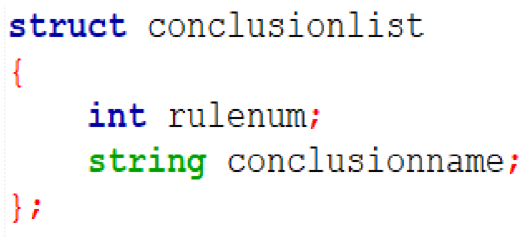
This Clause variable list will be stored in the text file and is read from array of strings "clauseVariableList".

```
string clauseVarList[300];
```

8.1.3 Conclusion List:

Conclusion list is a data structure which lists all the possible conclusions in sequential order.

In our Project, we created structure of conclusion list which consists of rulenum (rule number) and conclusionname(conclusion name).



```
struct conclusionlist
{
    int rulenum;
    string conclusionname;
};
```

All the consequents of the THEN part will be stored along with respective rule numbers.

Conclusion list in the project:

10	**DIAGNOSIS**
20	**DIAGNOSIS**
30	**DIAGNOSIS**
40	**DIAGNOSIS**
50	**DIAGNOSIS**
60	**DIAGNOSIS**
70	**DIAGNOSIS**
80	**DIAGNOSIS**
90	**DIAGNOSIS**
100	**DIAGNOSIS**
110	**DIAGNOSIS**
120	**DIAGNOSIS**
130	**DIAGNOSIS**
140	**DIAGNOSIS**
150	**DIAGNOSIS**
160	**DIAGNOSIS**
170	**DIAGNOSIS**
180	**DIAGNOSIS**
190	**DIAGNOSIS**
200	**DIAGNOSIS**

210	**DIAGNOSIS**
220	**DIAGNOSIS**
230	**DIAGNOSIS**

These values of Conclusion list will be stored in the text file and read from array of structures of type conclusionList.

```
conclusionlist conlist [27];
```

8.1.4 Conclusion stack:

We created a structure of conclusion stack which consists of rule number and clause number.

```
struct conclusionstack
{
    int rulenumber;
    int clausenumber;
};
```

This is very important part in implementing Backward Chaining. The conclusion stack is the central structure which ties together all of the other structures we've implemented in the implementation of the backward chaining expert system . It is the conclusion stack that tells us which IF-THEN statement contains the conclusion we are trying to reach and which clause in the IF portion is being examined for instantiation.

```
  stack <conclusionstack> conclusionStack;
```

8.2 Data structures needed for Forward chaining:

8.2.1 Clause variable List:

This simple list. If a rule does not utilize all of these locations, they are left blank. It contains clause variables for each rule.

Clause variable List in the project:

1	DIAGNOSIS
2	b
3	b
4	DIAGNOSIS
5	b
6	b
7	DIAGNOSIS
8	b
9	b
10	DIAGNOSIS
11	b
12	b
13	DIAGNOSIS
14	b
15	b
16	DIAGNOSIS
17	b
18	b
19	DIAGNOSIS
20	b
21	b
22	DIAGNOSIS
23	b
24	b

25	DIAGNOSIS
26	b
27	b
28	DIAGNOSIS
29	b
30	b
31	DIAGNOSIS
32	b
33	b
34	DIAGNOSIS
35	b
36	b
37	DIAGNOSIS
38	b
39	b
40	DIAGNOSIS
41	b
42	b
43	DIAGNOSIS
44	b
45	b
46	DIAGNOSIS
47	b
48	b
49	DIAGNOSIS
50	b
51	b
52	DIAGNOSIS
53	b

54	b
55	DIAGNOSIS
56	b
57	b
58	DIAGNOSIS
59	b
60	b
61	DIAGNOSIS
62	b
63	b
64	DIAGNOSIS
65	b
66	b
67	DIAGNOSIS
68	b
69	b

These values of Clause variable list will be stored in the text file and read from array of strings.

```
string CVL[cv_size];
```

In our project, clause variable size = 72.

8.2.2 Conclusion variable queue:

Conclusion Variable Queue It contains all the variables that are needed to be initialized to reach the conclusion. They are served in FCFS manner.

We used a queue for this data structure:

```
queue <string> cv_queue;
```

8.2.3 Variable List :

Variable List Lists all the variables in the rules and its values. The variable list is used to know whether the variable is instantiated or not instantiated.

When user enters some information for a variable, then it is instantiated, and the answer given by the user is stored in one more field called Element. In forward chaining, there is only one variable used i.e. “DIAGNOSIS”.

9. SOURCE CODE

9.1 BACKWARD CHAINING:

```
#ifndef BACKWARDCHAINING_H_INCLUDED
#define BACKWARDCHAINING_H_INCLUDED

#include <iostream>
#include <map>
#include <stack>
#include <string>

using namespace std;

struct variable
{
    string varname;
    string varstatus;
    string varvalue;
};

struct conclusionlist
{
    int rulenum;
    string conclusionname;
};

struct conclusionstack
{
    int rulenumber;
    int clausenumber;
};


class BackwardChaining
{
public:
    //variables
    variable variableList[33];
    conclusionlist conlist [27];
    stack <conclusionstack> conclusionStack;
    string clauseVarList[300];
    string KBBC[27];


    //functions
    BackwardChaining(); // Constructor
    string init();
    void knowledgeBaseBC();
    void conclusionStackOperation(int,int); // Conclusion Stack to store Rule number
and Clause Number
    int variableIndex(string); // returns the index of the variable
    string variableValue(string); // returns the value YES/NO of the variable
    void instantiation(string); //to instantiate a variable in the variable list
    variable knowledgeBase(int, map<string,string> &);


};

#endif // BACKWARDCHAINING_H_INCLUDED


#include "Project1-A04824992-BackwardChaining.h"
#include <iostream>
#include <fstream>

/*******************************************************************************
BackwardChaining(): is a special/ member function which initializes objects
                    of the class. It will read the Variable_list.txt,
Conclusion_list.txt
                    and clauseVariablelist list files if present and stores them
                    in respective array of structures like variable, conclusionlist
                    and string clauseVarList.
parameters used: no
returns : constructor does not have a return type.
*******************************************************************************/

BackwardChaining::BackwardChaining()
{
    //Initializing variable list
    ifstream fin;
    string name, status, value;
    fin.open("D://Artificial Intelligence/AkshathaJain/Project1-A04824992-
Variable_List.txt");
    if(!fin){
        cout << "error ! could not open the file";   }
    for(int i=0; i<33;i++)
    {
        fin >> name;
        fin >> status;
        variableList[i].varname = name;
        variableList[i].varstatus = status;
        variableList[i].varvalue = "";
    }

    //Initializing conclusion list
    ifstream fin1;
    int rulenum;
    string conclusionname;
    fin1.open("D://Artificial Intelligence/AkshathaJain/Project1-A04824992-
Conclusion_list.txt");
    if(!fin){
        cout << "error ! could not open the file";   }
    for(int i = 0; i < 27; i++){

        fin1 >> rulenum >> conclusionname;

        conlist[i].rulenum = rulenum;
        conlist[i].conclusionname = conclusionname;
    }

    //Initializing Clause variable list
    ifstream fin2;
    string clausevalue;
    fin2.open("D://Artificial Intelligence/AkshathaJain/Project1-A04824992-
Clausevarlist.txt");
    if(!fin2){
        cout << "error ! could not open the file";    }
    for(int i=1; i<299;i++)
    {
        fin2 >> clausevalue;
        clauseVarList[i] = clausevalue;
    }
}

/*****************************************************************************
init(): IF the rule number is satisfied, function will calculate clause number by
        using rule number and updates conclusion stack by calling function
        conclusionStackOperation, it will store the index of the variable by
        calling variableIndex() function, and instantiates the variable using
        instantiation() function.
        Then function gets user response by calling variableValue(), then it
        identifies the disease by calling knowledgebase().
Parameters used : no
return type: returns disease name if identified by knowledge base else it will
             return null;
*****************************************************************************/

string BackwardChaining::init(){

    for(int i = 0; i<27; i++){
        if(conlist[i].conclusionname == "DIAGNOSIS")
        {
            int ruleNum = conlist[i].rulenum;
            int clauseNum = 11*(ruleNum/10 -1) + 1;//computing clause number

            conclusionStackOperation(ruleNum, clauseNum);
            string symptom = clauseVarList[clauseNum];

            while(symptom != "b"){
                int varIndex = variableIndex(symptom);
                    instantiation(clauseVarList[varIndex]);
                    clauseNum++;
                    symptom = clauseVarList[clauseNum];
                }
            }
            while(conclusionStack.size() != 0){ //until the stack has rule numbers
available
```

```
                conclusionstack top = conclusionStack.top();
                conclusionStack.pop();
                int rule = top.rulenumber;
                int clause = top.clausenumber;

                map<string,string> clausePair;
                for(int i=clause; i<clause+11; i++){
                        string symp = clauseVarList[i];
                    if(symp != "b"){
                        string sympval = variableValue(symp);
                        clausePair.insert(pair<string,string>(symp, sympval));
                    }

                    else{
                        break;
                    }
                }

                variable ruleResult = knowledgeBase(rule, clausePair);
                if(ruleResult.varstatus == "true"){
                    if(ruleResult.varname == "DIAGNOSIS"){
                        cout<<"You are diagnosed with : "<< ruleResult.varvalue <<
endl;

                        conclusionStack.pop();
                        return ruleResult.varvalue;
                        break;
                    }
                }
                else//rule is not satisfied
                    break;
            }
        }

    }


void BackwardChaining::knowledgeBaseBC()
{
        //knowledge base
       ifstream fin;
       string knowledgebase;
       fin.open("D://Artificial Intelligence/AkshathaJain/Project1-A04824992-
KnowledgeBaseBC.txt");
       if(!fin){
        cout << "error ! could not open the file";    }
        for(int i=0; i<27;i++)
        {
        getline(fin, knowledgebase, '\n');
        KBBC[i] = knowledgebase;
        cout << KBBC[i] << endl << endl;
        }

}

/*******************************************************************************
```

```
conclusionStackOperation(): This function will push the rule numbers and clause
                            numbers and stores them in stack.
parameters used :  ruleNumber, clauseNumber.
returns : nothing, as it is a void function.
******************************************************************************/

void BackwardChaining::conclusionStackOperation(int ruleNumber, int clauseNumber){
    conclusionstack stacktemp = { ruleNumber, clauseNumber};
    conclusionStack.push(stacktemp);
}

/*****************************************************************************
variableIndex() : Function will ask user whether he/she suffering from any of the
                  and the data( string value YES/NO) will be stored in the index
                  of variable list.
                  if user enters other than YES/ NO it will ask the user to enter
                  correct value again.
                  Function will check whether Symptom or variable is present in
                  variableList or not, it will instantiate the variable status.

Parameters used : symptom
returns : (i)index of the variable is returned if the variable/symptom is present
           else it will write -1.
******************************************************************************/

int BackwardChaining::variableIndex(string symptom){

    for(int i=0; i< 33; i++){
        if(variableList[i].varname == symptom){
            if(variableList[i].varstatus == "NI"){
                cout << "Are you suffering from "<< variableList[i].varname<< " :
";
                string data;
                cin >> data;
                while((data != "YES")&& (data != "NO"))
                   {
                       cout << "WRONG ENTRY: Please enter YES or NO (uppercase)" <<
endl << endl;
                       cout << "Are you suffering from "<< variableList[i].varname<<
" :   ";
                       cin >> data;
                   }
                variableList[i].varvalue = data;
                variableList[i].varstatus = "I";
            }
            return i;
        }
    }
    return -1;
}

/*****************************************************************************
instantiation(): if a valid symptom(variable) is found from function
variableIndex(),it
                will instantiate the respective variable status in the conclusion
                list.
Parameters used : var
returns : nothing
******************************************************************************/

void BackwardChaining::instantiation(string var){
    for(int i=0; i<33; i++){ //--------->>>>>> update here as well
        if(variableList[i].varname == var){
            if(variableList[i].varstatus == "NI"){
                cout << "Are you suffering from  "<< variableList[i].varname << " : 
";
                string data;
                cin >> data;
                variableList[i].varvalue = data;
                variableList[i].varstatus = "I";
            }
            break;
        }
    }
}

/******************************************************************************
variableValue(): This function will update variable list depending on the user
                 input and instantiate the respective variable status.
Parameters used : symptom
returns : YES/NO, depending on the user input
*****************************************************************************/

string BackwardChaining::variableValue(string symptom){

    string res = "";
    for(int i=0; i<33; i++){ //----------->>>>>>>> update here as well
        if(variableList[i].varname == symptom){
            if(variableList[i].varstatus == "NI"){
                cout << "Are you suffering from "<< variableList[i].varname << " : 
";
                string data;
                cin >> data;
                variableList[i].varvalue = data;
                variableList[i].varstatus = "I";
                return data;
            }
            else{
                return variableList[i].varvalue;
            }
        }
    }
}

/******************************************************************************
knowledgeBase(): depending on the rule number function will check the clause variable
                 list, if all the clause variables are matching with the user entered
                 data then the function will diagnose the disease and updates 
variable
                 name, variable status and variable value.
```

```
Parameters used :ruleNumber, clausePair( is a reference variable)
returns : returns diagnosis of type structure variable.
****************************************************************************/

variable BackwardChaining::knowledgeBase(int ruleNumber, map<string,string>
&clausePair)
{
    variable diagnosis;
    diagnosis.varname = "";
    diagnosis.varstatus = "false";
    diagnosis.varvalue = "";

    if(ruleNumber == 10){

        if(clausePair.find("HEAD_INJURY")->second == "YES"
           && clausePair.find("ANXIETY")->second == "YES" &&
           clausePair.find("INSOMNIA")->second == "YES" &&
           clausePair.find("AGGRESSION")->second == "YES" &&
           clausePair.find("DISORIENTATION")->second == "YES" &&
           clausePair.find("HALLUCINATIONS")->second == "YES" &&
           clausePair.find("DELUSIONS")->second == "YES" &&
           clausePair.find("UNABLE_TO_MOVE")->second
           == "YES"){
            diagnosis.varstatus = "true";
            diagnosis.varname = "DIAGNOSIS";
            diagnosis.varvalue = "Schizophrenia";
            return diagnosis;
        }
    }

        if(ruleNumber == 20){

        if(clausePair.find("HEAD_INJURY")->second == "YES" &&
           clausePair.find("ANXIETY")->second == "YES"
           && clausePair.find("INSOMNIA")->second == "YES" &&
           clausePair.find("AGGRESSION")->second == "YES" &&
           clausePair.find("DISORIENTATION")->second == "YES" &&
           clausePair.find("HALLUCINATIONS")->second == "NO" &&
           clausePair.find("SWEATING")->second =="YES" ){
            diagnosis.varstatus = "true";
            diagnosis.varname = "DIAGNOSIS";
            diagnosis.varvalue = "General_Anxiety_Disorder";
            return diagnosis;
        }
    }

    if(ruleNumber == 30){

        if( clausePair.find("HEAD_INJURY")->second == "YES" &&
           clausePair.find("ANXIETY")->second ==  "YES" &&
           clausePair.find("INSOMNIA")->second ==  "YES"
           &&  clausePair.find("AGGRESSION")->second == "YES" &&
           clausePair.find("DISORIENTATION")->second == "NO" &&
           clausePair.find("HALLUCINATIONS")->second == "YES"
           && clausePair.find("MOOD_SWINGS")->second == "YES"){
            diagnosis.varstatus = "true";
            diagnosis.varname = "DIAGNOSIS";
            diagnosis.varvalue = "Bipolar_I";
            return diagnosis;
        }
    }

    if(ruleNumber == 40){

        if( clausePair.find("HEAD_INJURY")->second == "YES" &&
           clausePair.find("ANXIETY")->second ==  "YES" &&
           clausePair.find("INSOMNIA")->second ==  "YES" &&
           clausePair.find("AGGRESSION")->second == "YES" &&
           clausePair.find("DISORIENTATION")->second == "NO" &&
           clausePair.find("HALLUCINATIONS")->second == "NO" &&
           clausePair.find("FLASHBACKS")->second == "YES") {
            diagnosis.varstatus = "true";
            diagnosis.varname = "DIAGNOSIS";
            diagnosis.varvalue = "Post_Traumatic_Stress_Disorder";
            return diagnosis;
        }
    }

    if(ruleNumber == 50){

        if( clausePair.find("HEAD_INJURY")->second == "YES" &&
           clausePair.find("ANXIETY")->second ==  "YES" &&
           clausePair.find("INSOMNIA")->second ==  "YES" &&
           clausePair.find("AGGRESSION")->second == "NO" &&
           clausePair.find("DEPRESSION")->second == "YES" &&
           clausePair.find("MOOD_SWINGS")->second == "YES" &&
           clausePair.find("FATIGUE")->second == "YES" ){
            diagnosis.varstatus = "true";
            diagnosis.varname = "DIAGNOSIS";
            diagnosis.varvalue = "Major_Depressive_Disorder";
            return diagnosis;
        }
    }

    if(ruleNumber == 60){

        if( clausePair.find("HEAD_INJURY")->second == "NO" &&
           clausePair.find("ANXIETY")->second ==  "YES" &&
           clausePair.find("DEPRESSION")->second ==  "YES" &&
           clausePair.find("DISORIENTATION")->second == "YES" &&
           clausePair.find("INSOMNIA")->second == "YES" &&
           clausePair.find("MEMORY_LOSS")->second == "YES" &&
           clausePair.find("UNABLE_TO_MOVE")->second == "YES" &&
           clausePair.find("PARANOID")->second == "YES" ) {
            diagnosis.varstatus = "true";
            diagnosis.varname = "DIAGNOSIS";
            diagnosis.varvalue = "Alzheimers";
            return diagnosis;
        }
    }
```

```
if(ruleNumber == 70){

    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "YES" &&
       clausePair.find("DEPRESSION")->second ==  "YES" &&
       clausePair.find("DISORIENTATION")->second == "YES" &&
       clausePair.find("INSOMNIA")->second == "YES" &&
       clausePair.find("MEMORY_LOSS")->second == "YES" &&
       clausePair.find("UNABLE_TO_MOVE")->second == "NO" &&
       clausePair.find("HALLUCINATIONS")->second == "YES" &&
       clausePair.find("DIFFICULTY_PLANNING")->second == "YES" ){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Dementia";
        return diagnosis;
    }
}

if(ruleNumber == 80){

    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "YES" &&
       clausePair.find("DEPRESSION")->second ==  "YES" &&
       clausePair.find("DISORIENTATION")->second == "YES" &&
       clausePair.find("INSOMNIA")->second == "YES" &&
       clausePair.find("MEMORY_LOSS")->second == "NO" &&
       clausePair.find("SOCIAL_ISOLATION")->second == "YES"){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Major_Depressive_Disorder";
        return diagnosis;
    }
}

if(ruleNumber == 90){

    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "YES" &&
       clausePair.find("DEPRESSION")->second ==  "YES" &&
       clausePair.find("DISORIENTATION")->second == "YES" &&
       clausePair.find("INSOMNIA")->second == "NO" &&
       clausePair.find("SOCIAL_ISOLATION")->second == "YES"){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Schizo_Affective_Disorder";
        return diagnosis;
    }
}

if(ruleNumber == 100){

    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "YES" &&
       clausePair.find("DEPRESSION")->second ==  "YES" &&
       clausePair.find("DISORIENTATION")->second == "NO" &&
       clausePair.find("AMNESIA")->second == "YES" &&
       clausePair.find("TREMOR")->second == "YES"){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Parkinsons_Disorder";
        return diagnosis;
    }
}

if(ruleNumber == 110){

    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "YES" &&
       clausePair.find("DEPRESSION")->second ==  "YES"
       &&  clausePair.find("DISORIENTATION")->second == "NO" &&
       clausePair.find("AMNESIA")->second == "NO" &&
       clausePair.find("HALLUCINATIONS")->second == "YES"
       && clausePair.find("AGGRESSION")->second == "YES" ){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Bipolar_I";
        return diagnosis;
    }
}

if(ruleNumber == 120){

    if(clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "YES" &&
       clausePair.find("DEPRESSION")->second ==  "YES" &&
       clausePair.find("DISORIENTATION")->second == "NO" &&
       clausePair.find("AMNESIA")->second == "YES" &&
       clausePair.find("TREMOR")->second == "NO" &&
       clausePair.find("IMPULSIVE")->second == "YES" ){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Mild_Cognitive_Impairment";
        return diagnosis;
    }
}

if(ruleNumber == 130){

    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "YES" &&
       clausePair.find("DEPRESSION")->second ==  "NO"
       &&  clausePair.find("AGGRESSION")->second == "YES" &&
       clausePair.find("INSOMNIA")->second == "YES" &&
       clausePair.find("DISORIENTATION")->second == "YES"
       && clausePair.find("TREMOR")->second == "YES" &&
       clausePair.find("HALLUCINATIONS")->second == "YES" &&
       clausePair.find("UNABLE_TO_MOVE")->second == "YES"){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Schizophrenia";
        return diagnosis;
    }
}

if(ruleNumber == 140){

    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "YES" &&
       clausePair.find("DEPRESSION")->second ==  "NO"
       &&  clausePair.find("AGGRESSION")->second == "YES" &&
       clausePair.find("INSOMNIA")->second == "YES" &&
       clausePair.find("DISORIENTATION")->second == "YES"
       && clausePair.find("TREMOR")->second == "YES" &&
       clausePair.find("HALLUCINATIONS")->second == "NO" &&
       clausePair.find("SWEATING")->second == "YES" ){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "General_Anxiety_Disorder";
        return diagnosis;
    }
}

if(ruleNumber == 150){

    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "YES" &&
       clausePair.find("DEPRESSION")->second ==  "NO"
       &&  clausePair.find("AGGRESSION")->second == "YES" &&
       clausePair.find("INSOMNIA")->second == "NO" &&
       clausePair.find("COUNTING_IN_PATTERN")->second == "YES" ){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Obsessive_Compulsive_Disorder";
        return diagnosis;
    }
}

if(ruleNumber == 160){

    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "YES" &&
       clausePair.find("DEPRESSION")->second ==  "NO" &&
       clausePair.find("AGGRESSION")->second == "NO" &&
        clausePair.find("PARESTHESIAS")->second == "YES" &&
       clausePair.find("PALPITATIONS")->second == "YES" ){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Panic_Attack";
        return diagnosis;
    }
}

if(ruleNumber == 170){
    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "YES" &&
       clausePair.find("DEPRESSION")->second ==  "NO" &&
       clausePair.find("AGGRESSION")->second == "NO" &&
       clausePair.find("PARESTHESIAS")->second == "NO" &&
       clausePair.find("URGE_TO_STEAL")->second == "YES"
       && clausePair.find("RELIEF_STEALING")->second == "YES"){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Kleptomania";
        return diagnosis;
    }
}

if(ruleNumber == 180){

    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "YES" &&
       clausePair.find("DEPRESSION")->second ==  "NO"
       &&  clausePair.find("AGGRESSION")->second == "NO" &&
       clausePair.find("PARESTHESIAS ")->second == "NO" &&
       clausePair.find("URGE_TO_STEAL")->second == "NO"
       && clausePair.find("HALLUCINATIONS")->second == "YES" &&
       clausePair.find("MEMORY_LOSS")->second == "YES"){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Hypersomnolence";
        return diagnosis;
    }
}

if(ruleNumber == 190){

    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "NO" &&
       clausePair.find("AGGRESSION")->second ==  "YES" &&
       clausePair.find("MISBEHAVE")->second == "YES" &&
       clausePair.find("RESENTMENT")->second == "YES"){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Oppositional_Defiant_Disorder";
        return diagnosis;
    }
}

if(ruleNumber == 200){

    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "NO" &&
       clausePair.find("AGGRESSION")->second ==  "YES"
       &&  clausePair.find("MISBEHAVE")->second == "YES" &&
       clausePair.find("RESENTMENT")->second == "NO" &&
       clausePair.find("DISTRESSED")->second == "YES" ){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Intermittent_Explosive_Disorder";
        return diagnosis;
    }
}

if(ruleNumber == 210){

    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "NO" &&
       clausePair.find("AGGRESSION")->second ==  "YES"
       &&  clausePair.find("MISBEHAVE")->second == "NO" &&
       clausePair.find("INSOMNIA")->second == "YES" &&
       clausePair.find("DISORIENTATION")->second == "YES"
       && clausePair.find("TALKATIVE")->second == "YES" &&
       clausePair.find("DEPRESSION")->second == "YES"){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Cyclothymia";
        return diagnosis;
    }
}

if(ruleNumber == 220){

    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "NO" &&
       clausePair.find("AGGRESSION")->second ==  "YES"
       &&  clausePair.find("MISBEHAVE")->second == "NO" &&
       clausePair.find("INSOMNIA")->second == "YES" &&
       clausePair.find("DISORIENTATION")->second == "YES"
       && clausePair.find("TALKATIVE")->second == "YES" &&
       clausePair.find("DEPRESSION")->second == "NO" &&
       clausePair.find("EUPHORIA")->second == "YES" ){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Hypomania";
        return diagnosis;
    }
}

if(ruleNumber == 230){

    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "NO" &&
       clausePair.find("AGGRESSION")->second ==  "YES"
       &&  clausePair.find("MISBEHAVE")->second == "NO" &&
       clausePair.find("INSOMNIA")->second == "YES" &&
       clausePair.find("DISORIENTATION")->second == "YES"
       && clausePair.find("TALKATIVE")->second == "NO" &&
       clausePair.find("ANNOYED")->second == "YES"){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Dysthymia";
        return diagnosis;
    }
}

if(ruleNumber == 240){

    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "NO" &&
       clausePair.find("AGGRESSION")->second ==  "YES"
       &&  clausePair.find("MISBEHAVE")->second == "NO" &&
       clausePair.find("INSOMNIA")->second == "NO" &&
       clausePair.find("TREMOR")->second == "YES"){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Delirium_Tremens";
        return diagnosis;
    }
}

if(ruleNumber == 250){

    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "NO" &&
       clausePair.find("AGGRESSION")->second ==  "NO"
       &&  clausePair.find("SOCIAL_ISOLATION")->second == "YES" &&
       clausePair.find("INFERIORITY")->second == "YES"){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Social_Anxiety_Disorder";
        return diagnosis;
    }
}


if(ruleNumber == 260){

    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "NO" &&
       clausePair.find("AGGRESSION")->second ==  "NO"
       &&  clausePair.find("SOCIAL_ISOLATION")->second == "YES" &&
       clausePair.find("INFERIORITY")->second == "NO" &&
       clausePair.find("DELUSIONS")->second == "YES"){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Agrophobia";
        return diagnosis;
    }
}

if(ruleNumber == 270){

    if( clausePair.find("HEAD_INJURY")->second == "NO" &&
       clausePair.find("ANXIETY")->second ==  "NO" &&
       clausePair.find("AGGRESSION")->second ==  "NO"
       &&  clausePair.find("SOCIAL_ISOLATION")->second == "NO" &&
       clausePair.find("FAST_HEART_RATE")->second == "YES" &&
       clausePair.find("SWEATING")->second == "YES"){
        diagnosis.varstatus = "true";
        diagnosis.varname = "DIAGNOSIS";
        diagnosis.varvalue = "Panic_Disorder_with_Agrophobia";
        return diagnosis;
    }
}
return diagnosis;
}
```

Forward Chaining :

```
#ifndef FORWARDCHAINING_H
#define FORWARDCHAINING_H
#include <iostream>
#include <map>
#include <queue>
#include <string>

using namespace std;

class ForwardChaining
{
public:
    static const int v_size = 1, r_size = 24, cv_size = 72, var_inc = 10;
    string KB[r_size], Res;
    string CVL[cv_size];

    map<int, string> map_rulenum;
    map<int, string> map_clausenum;
    map<string, string> v_list_Ins = {{"DIAGNOSIS", "NI"}};
    queue <string> cv_queue;
    ForwardChaining();
    void KB_FW();
    void MapRuleNumbertoKB();
    void MapCN_CVL();
    bool Comparetwostr(string str1,string str2);
    void diagnosis_treatment(string str);
};

#endif // FORWARDCHAINING_H
```

```
#include "Project1-A04824992-ForwardChaining.h"
#include <iostream>
#include <fstream>
#include <iterator>
#include <algorithm>



using namespace std;

/*************************************************************************
ForwardChaining(): is a special/ member function which initializes objects
                    of the class. It initializes the clause variable list.
                    It will update the knowledge base by calling KB_FW(),
                    maps the rules number to Knowledge Base by calling
                    MapRuleNumbertoKB(), maps clause number to conclusion
                    variable list by calling MapCN_CVL() function.
parameters used: no
returns : constructor does not have a return type.
*************************************************************************/

ForwardChaining::ForwardChaining()
{
       //initializing clause variable list
       ifstream fin;
       string cVL;
       fin.open("D://Artificial Intelligence/AkshathaJain/Project1-A04824992-
ClausevarlistFC.txt");
       if(!fin){
        cout << "error ! could not open the file";   }
        for(int i=0; i<cv_size;i++)
        {
        fin >> cVL;
        CVL[i] = cVL;
        }

        KB_FW();
        MapRuleNumbertoKB();
        MapCN_CVL();
}

/*************************************************************************
KB_FW(): This function will store the knowledge base for forward chaining using
         string array for all the rule numbers.

Parameters Used : no
return type : nothing, as it is a void function.
*************************************************************************/

void ForwardChaining::KB_FW()
{
       //knowledge base
       ifstream fin;
       string knowledgebase;
       fin.open("D://Artificial Intelligence/AkshathaJain/Project1-A04824992-
KnowledgeBaseFC.txt");
       if(!fin){
        cout << "error ! could not open the file";   }
        for(int i=0; i<r_size;i++)
        {
        getline(fin, knowledgebase, '\n');
        KB[i] = knowledgebase;
        }
}

/*****************************************************************************
MapRuleNumbertoKB() : This function maps the rule number to knowledge base.
Parameters used : no
return type: no, as it is a void function.
****************************************************************************/

void ForwardChaining::MapRuleNumbertoKB()
{
    map_rulenum.insert(pair<int, string>(10, KB[0]));
    map_rulenum.insert(pair<int, string>(20, KB[1]));
    map_rulenum.insert(pair<int, string>(30, KB[2]));
    map_rulenum.insert(pair<int, string>(40, KB[3]));
    map_rulenum.insert(pair<int, string>(50, KB[4]));
    map_rulenum.insert(pair<int, string>(60, KB[5]));
    map_rulenum.insert(pair<int, string>(70, KB[6]));
    map_rulenum.insert(pair<int, string>(80, KB[7]));
    map_rulenum.insert(pair<int, string>(90, KB[8]));
    map_rulenum.insert(pair<int, string>(100, KB[9]));
    map_rulenum.insert(pair<int, string>(110, KB[10]));
    map_rulenum.insert(pair<int, string>(120, KB[11]));
    map_rulenum.insert(pair<int, string>(130, KB[12]));
    map_rulenum.insert(pair<int, string>(140, KB[13]));
    map_rulenum.insert(pair<int, string>(150, KB[14]));
    map_rulenum.insert(pair<int, string>(160, KB[15]));
    map_rulenum.insert(pair<int, string>(170, KB[16]));
    map_rulenum.insert(pair<int, string>(180, KB[17]));
    map_rulenum.insert(pair<int, string>(190, KB[18]));
    map_rulenum.insert(pair<int, string>(200, KB[19]));
    map_rulenum.insert(pair<int, string>(210, KB[20]));
    map_rulenum.insert(pair<int, string>(220, KB[21]));
    map_rulenum.insert(pair<int, string>(230, KB[22]));
    map_rulenum.insert(pair<int, string>(240, KB[23]));
}

/*****************************************************************************
MapCN_CVL(): maps clause number to conclusion variable list.
Parameters used : no
return type : no, as it is a void function.
****************************************************************************/

void ForwardChaining::MapCN_CVL()
{
    map_clausenum.insert(pair<int, string>(1, CVL[0]));
    map_clausenum.insert(pair<int, string>(4, CVL[3]));
```

```
        map_clausenum.insert(pair<int, string>(7, CVL[6]));
        map_clausenum.insert(pair<int, string>(10, CVL[9]));
        map_clausenum.insert(pair<int, string>(13, CVL[12]));
        map_clausenum.insert(pair<int, string>(16, CVL[15]));
        map_clausenum.insert(pair<int, string>(19, CVL[18]));
        map_clausenum.insert(pair<int, string>(22, CVL[21]));
        map_clausenum.insert(pair<int, string>(25, CVL[24]));
        map_clausenum.insert(pair<int, string>(28, CVL[27]));
        map_clausenum.insert(pair<int, string>(31, CVL[30]));
        map_clausenum.insert(pair<int, string>(34, CVL[33]));
        map_clausenum.insert(pair<int, string>(37, CVL[36]));
        map_clausenum.insert(pair<int, string>(40, CVL[39]));
        map_clausenum.insert(pair<int, string>(43, CVL[42]));
        map_clausenum.insert(pair<int, string>(46, CVL[45]));
        map_clausenum.insert(pair<int, string>(49, CVL[48]));
        map_clausenum.insert(pair<int, string>(52, CVL[51]));
        map_clausenum.insert(pair<int, string>(55, CVL[54]));
        map_clausenum.insert(pair<int, string>(58, CVL[57]));
        map_clausenum.insert(pair<int, string>(61, CVL[60]));
        map_clausenum.insert(pair<int, string>(64, CVL[63]));
        map_clausenum.insert(pair<int, string>(67, CVL[66]));
    }

    /*******************************************************************************
    Comparetwostr() : function will cpmapare 2 strings.
    Parameters Used : s1, s2.
    return type : returns true if 2 strings matchelse it will return false.
    *******************************************************************************/

    bool ForwardChaining::Comparetwostr(string s1, string s2)
    {
        transform(s1.begin(), s1.end(), s1.begin(), ::toupper);
        transform(s2.begin(), s2.end(), s2.begin(), ::toupper);

        if (s1 == s2)
        {
            return true;
        }

        else
        {
            return false;
        }
    }

    /*******************************************************************************
    diagnosis_treatment() : when the expert system detects the disease, this function
                            will calculate rule number by using clause variable and
                            clauses in the IF matches with Diagnosis of the expert
                            system it will print the treatment for the corresponding
                            disease.
    Parameters Used: diagnosis.
    return type: no, as it is a void function.
    ******************************************************************************/
```

```
void ForwardChaining::diagnosis_treatment(string diagnosis)
{


    cv_queue.push("DIAGNOSIS");
    v_list_Ins["DIAGNOSIS"] = diagnosis;
    int clausenum = 1, rulenum;

    if (diagnosis != "")
    {
        while (!cv_queue.empty())
        {

            map<int, string>::iterator i;
            for (i = map_clausenum.begin(); i != map_clausenum.end(); i++)
            {
                if (i->second != "b")
                {
                    // Conversion of Clause number to Rule number
                    rulenum = (((clausenum / 3) + 1) * 10);
                }
            }
            // Finding the DISORDER and comparing it with knowledge base
            int pos = map_rulenum[rulenum].find("=");
            int pos1 = map_rulenum[rulenum].find(" THEN");
            string compare = map_rulenum[rulenum].substr(pos + 1, pos1 - (pos + 1));
            if (Comparetwostr(compare, diagnosis))
            {
                int treatment = map_rulenum[rulenum].rfind("=");
                Res = map_rulenum[rulenum].substr(treatment,
map_rulenum[rulenum].length());
                Res.erase(0,1);
                cout << "Treatment for the disorder " << diagnosis << " is : " << Res
<< endl;
                break;
            }
            else
            {
                clausenum = clausenum + 2;

            }
        }
    }

    else
        cout << "Please consult a Psychologist! No treatment can be detected." <<
endl;

}
```

INFERENCE – MAIN DRIVER FUNCTION

```
#include <iostream>
#include "Project1-A04824992-BackwardChaining.h"
#include "Project1-A04824992-ForwardChaining.h"
#include <iomanip>

using namespace std;

int main()
{

    BackwardChaining disorder;
    ForwardChaining treatment;

    cout << endl << endl;
    cout << 
"*************************************************************************************
**********" << endl;
    cout << "                   YOU ARE NOW IN THE MENTAL DISORDER DIAGNOSIS PORTAL!" 
<< endl ;
    cout << 
"*************************************************************************************
**********" << endl;
    cout << endl;
    cout << "Here you can diagnose the Disorder and also see what medicines can be 
prescribed for it!" << endl;
    cout << endl;
    cout << "Please select from the options below: " << endl;
    cout << "1. Identify the disease and get the treatment!" << endl << "2. Print the 
Knowledge Base, Variable List, Clause Variable List and Conclusion List for Backward 
Chaining" << endl
         << "3. Print the Knowledge Base and Clause Variable List for Forward 
Chaining" << endl;

    int choice;
    cin >> choice;

    switch(choice)
    {
        case 1:
        {
            cout << "Please enter YES/NO for the following symptoms" << endl << endl;
            string disease = disorder.init();
            cout << endl;
            treatment.diagnosis_treatment(disease);
            break;
        }
        case 2:
        {

            cout << 
"*************************************************************************************
**********" << endl;
            cout << "The Knowledge Base for Backward Chaining is as below:" << endl;
            cout <<
"************************************************************************************
**********" << endl << endl;
            disorder.knowledgeBaseBC();
            cout << endl;

            cout <<
"************************************************************************************
**********" << endl;
            cout << "The Variable List for Backward Chaining is as below:" << endl;
            cout <<
"************************************************************************************
**********" << endl << endl;
            cout << left;
            cout << setw(25) << "NAME" << setw(20) << "STATUS" << setw(10) << "VALUE"
<< endl << endl;
            for(int i=0; i<33;i++)
            {
                cout << left;
                cout << setw(25) << disorder.variableList[i].varname << setw(20) <<
disorder.variableList[i].varstatus << setw(10) << disorder.variableList[i].varvalue
<< endl;

            }
            cout << endl;

            cout <<
"************************************************************************************
**********" << endl;
            cout << "The Clause variable list for Backward Chaining is as below:" <<
endl;
            cout <<
"************************************************************************************
**********" << endl << endl;
            cout << left;
            for(int i = 1; i < 299; i++){
            cout << disorder.clauseVarList[i] << endl;
            }
            cout << endl;

            cout <<
"************************************************************************************
**********" << endl;
            cout << "The Conclusion list for Backward Chaining is as below:" << endl;
            cout <<
"************************************************************************************
**********" << endl << endl;
            cout << left;
            cout << "RULE NUMBER" << setw(10) << "\t" << "CONCLUSION" << endl <<
endl;

            for(int i = 0; i < 27; i++){
                cout << left;
                cout << setw(10) << disorder.conlist[i].rulenum << setw(10) << "\t"
<< disorder.conlist[i].conclusionname << endl;
```

```
            }
            cout << endl;
            break;

        }
        case 3:
        {
            cout <<
"***********************************************************************************
**********" << endl;
            cout << "The Knowledge Base for Backward Chaining is as below:" << endl;
            cout <<
"***********************************************************************************
**********" << endl << endl;
            for(int i = 0; i<24; i++)
            {
                cout << treatment.KB[i] << endl << endl;
            }
            cout << endl;

            cout <<
"***********************************************************************************
**********" << endl;
            cout << "The Clause Variable List for Backward Chaining is as below:" <<
endl;
            cout <<
"***********************************************************************************
**********" << endl << endl;
            for(int i=0; i<72;i++){
                cout << treatment.CVL[i] << endl;
            }
            break;
        }

    }
    return 0;
}
```

10. PROGRAM RUN

10.1 Sample Run1:

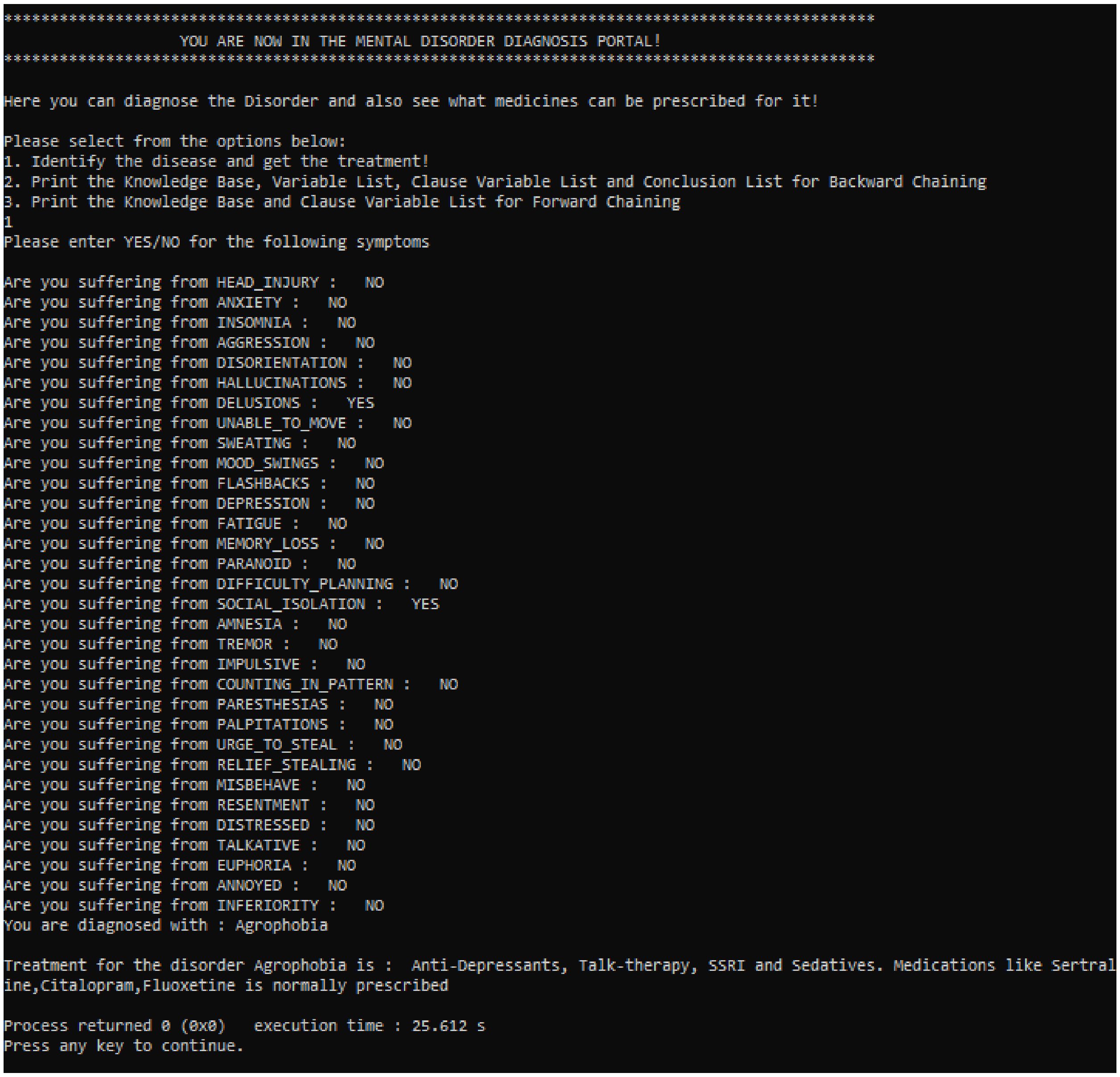
BACKWARD RULE:

IF HEAD_INJURY = NO && ANXIETY = NO && AGGRESSION = NO && SOCIAL_ISOLATION = YES && INFERIORITY = NO && DELUSIONS = YES THEN DIAGNOSIS = Agrophobia.

FORWARD RULE:

IF DIAGNOSIS=Agrophobia THEN TREATMENT= Anti-Depressants, Talk-therapy, SSRI and Sedatives. Medications like Sertraline,Citalopram,Fluoxetine is normally prescribed

Output 1:



```
**********************************************************************************************
                   YOU ARE NOW IN THE MENTAL DISORDER DIAGNOSIS PORTAL!
**********************************************************************************************

Here you can diagnose the Disorder and also see what medicines can be prescribed for it!

Please select from the options below:
1. Identify the disease and get the treatment!
2. Print the Knowledge Base, Variable List, Clause Variable List and Conclusion List for Backward Chaining
3. Print the Knowledge Base and Clause Variable List for Forward Chaining
1
Please enter YES/NO for the following symptoms

Are you suffering from HEAD_INJURY :   NO
Are you suffering from ANXIETY :   NO
Are you suffering from INSOMNIA :   NO
Are you suffering from AGGRESSION :   NO
Are you suffering from DISORIENTATION :   NO
Are you suffering from HALLUCINATIONS :   NO
Are you suffering from DELUSIONS :   YES
Are you suffering from UNABLE_TO_MOVE :   NO
Are you suffering from SWEATING :   NO
Are you suffering from MOOD_SWINGS :   NO
Are you suffering from FLASHBACKS :   NO
Are you suffering from DEPRESSION :   NO
Are you suffering from FATIGUE :   NO
Are you suffering from MEMORY_LOSS :   NO
Are you suffering from PARANOID :   NO
Are you suffering from DIFFICULTY_PLANNING :   NO
Are you suffering from SOCIAL_ISOLATION :   YES
Are you suffering from AMNESIA :   NO
Are you suffering from TREMOR :   NO
Are you suffering from IMPULSIVE :   NO
Are you suffering from COUNTING_IN_PATTERN :   NO
Are you suffering from PARESTHESIAS :   NO
Are you suffering from PALPITATIONS :   NO
Are you suffering from URGE_TO_STEAL :   NO
Are you suffering from RELIEF_STEALING :   NO
Are you suffering from MISBEHAVE :   NO
Are you suffering from RESENTMENT :   NO
Are you suffering from DISTRESSED :   NO
Are you suffering from TALKATIVE :   NO
Are you suffering from EUPHORIA :   NO
Are you suffering from ANNOYED :   NO
Are you suffering from INFERIORITY :   NO
You are diagnosed with : Agrophobia

Treatment for the disorder Agrophobia is :  Anti-Depressants, Talk-therapy, SSRI and Sedatives. Medications like Sertral
ine,Citalopram,Fluoxetine is normally prescribed

Process returned 0 (0x0)   execution time : 25.612 s
Press any key to continue.
```

10.2 Sample Run2:

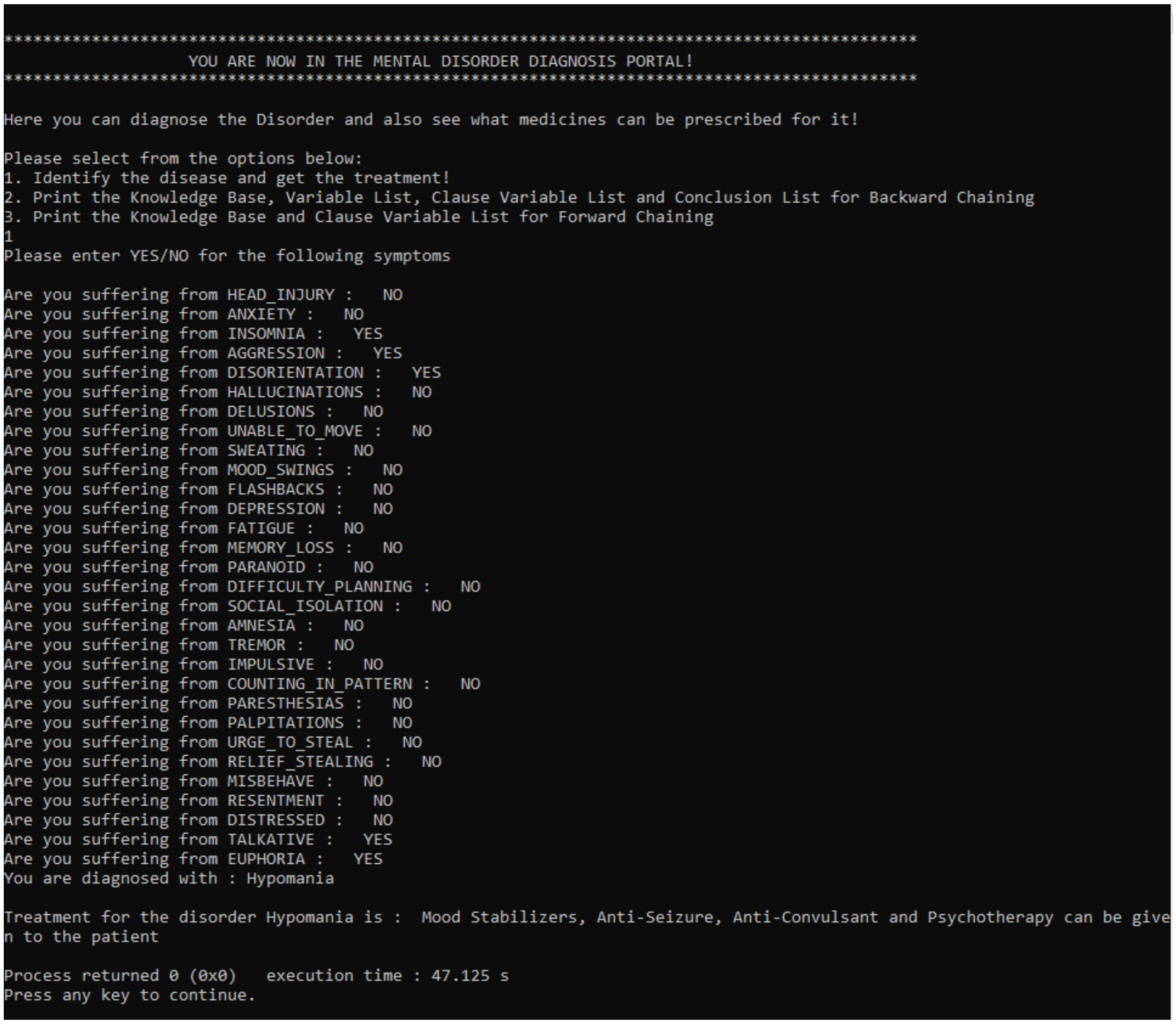
BACKWARD RULE:

IF HEAD_INJURY = NO && ANXIETY = NO && AGGRESSION = YES && MISBEHAVE = NO && INSOMNIA = YES && DISORIENTATION = YES && TALKATIVE = YES && DEPRESSION = NO && EUPHORIA = YES THEN DIAGNOSIS = Hypomania.

FORWARD RULE:

IF DIAGNOSIS=Hypomania THEN TREATMENT= Mood Stabilizers, Anti-Seizure, Anti-Convulsant and Psychotherapy can be given to the patient.

Output 2:



```
*********************************************************************************************
                   YOU ARE NOW IN THE MENTAL DISORDER DIAGNOSIS PORTAL!
*********************************************************************************************

Here you can diagnose the Disorder and also see what medicines can be prescribed for it!

Please select from the options below:
1. Identify the disease and get the treatment!
2. Print the Knowledge Base, Variable List, Clause Variable List and Conclusion List for Backward Chaining
3. Print the Knowledge Base and Clause Variable List for Forward Chaining
1
Please enter YES/NO for the following symptoms

Are you suffering from HEAD_INJURY :   NO
Are you suffering from ANXIETY :   NO
Are you suffering from INSOMNIA :   YES
Are you suffering from AGGRESSION :   YES
Are you suffering from DISORIENTATION :   YES
Are you suffering from HALLUCINATIONS :   NO
Are you suffering from DELUSIONS :   NO
Are you suffering from UNABLE_TO_MOVE :   NO
Are you suffering from SWEATING :   NO
Are you suffering from MOOD_SWINGS :   NO
Are you suffering from FLASHBACKS :   NO
Are you suffering from DEPRESSION :   NO
Are you suffering from FATIGUE :   NO
Are you suffering from MEMORY_LOSS :   NO
Are you suffering from PARANOID :   NO
Are you suffering from DIFFICULTY_PLANNING :   NO
Are you suffering from SOCIAL_ISOLATION :   NO
Are you suffering from AMNESIA :   NO
Are you suffering from TREMOR :   NO
Are you suffering from IMPULSIVE :   NO
Are you suffering from COUNTING_IN_PATTERN :   NO
Are you suffering from PARESTHESIAS :   NO
Are you suffering from PALPITATIONS :   NO
Are you suffering from URGE_TO_STEAL :   NO
Are you suffering from RELIEF_STEALING :   NO
Are you suffering from MISBEHAVE :   NO
Are you suffering from RESENTMENT :   NO
Are you suffering from DISTRESSED :   NO
Are you suffering from TALKATIVE :   YES
Are you suffering from EUPHORIA :   YES
You are diagnosed with : Hypomania

Treatment for the disorder Hypomania is :  Mood Stabilizers, Anti-Seizure, Anti-Convulsant and Psychotherapy can be give
n to the patient

Process returned 0 (0x0)   execution time : 47.125 s
Press any key to continue.
```

10.3 Sample Run3:

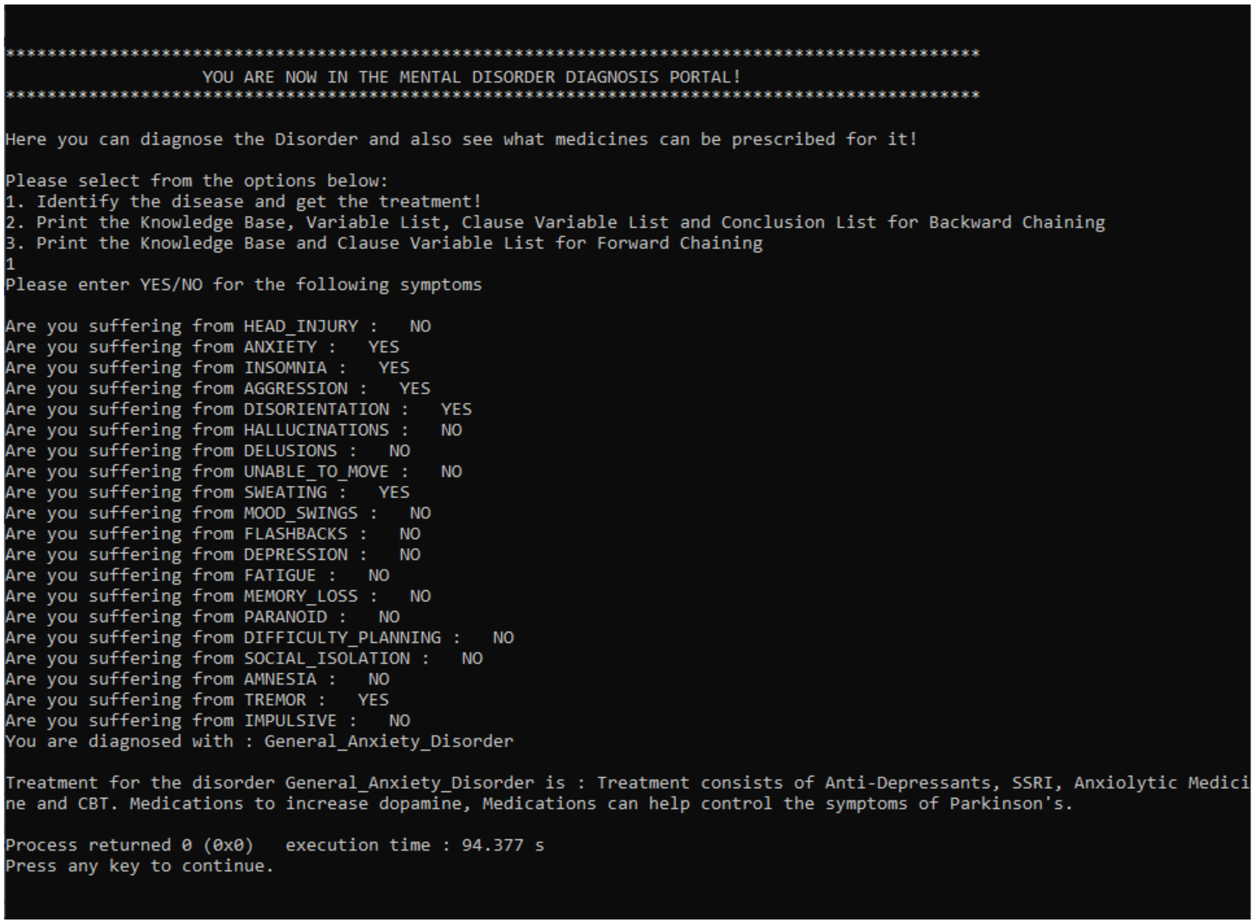
BACKWARD RULE:

IF HEAD_INJURY = NO && ANXIETY = YES && DEPRESSION = NO && AGGRESSION = YES && INSOMNIA = YES && DISORIENTATION = YES && TREMOR = YES && HALLUCINATIONS = NO && SWEATING = YES && NAUSEA = YES THEN DIAGNOSIS = Generalized Anxiety Disorder.

FORWARD RULE:

IF DIAGNOSIS=General_Anxiety_Disorder THEN TREATMENT=Treatment consists of Anti-Depressants, SSRI, Anxiolytic Medicine and CBT. Medications to increase dopamine, Medications can help control the symptoms of Parkinson's.

Output 3:



```
*********************************************************************************************
                   YOU ARE NOW IN THE MENTAL DISORDER DIAGNOSIS PORTAL!
*********************************************************************************************

Here you can diagnose the Disorder and also see what medicines can be prescribed for it!

Please select from the options below:
1. Identify the disease and get the treatment!
2. Print the Knowledge Base, Variable List, Clause Variable List and Conclusion List for Backward Chaining
3. Print the Knowledge Base and Clause Variable List for Forward Chaining
1
Please enter YES/NO for the following symptoms

Are you suffering from HEAD_INJURY :   NO
Are you suffering from ANXIETY :   YES
Are you suffering from INSOMNIA :   YES
Are you suffering from AGGRESSION :   YES
Are you suffering from DISORIENTATION :   YES
Are you suffering from HALLUCINATIONS :   NO
Are you suffering from DELUSIONS :   NO
Are you suffering from UNABLE_TO_MOVE :   NO
Are you suffering from SWEATING :   YES
Are you suffering from MOOD_SWINGS :   NO
Are you suffering from FLASHBACKS :   NO
Are you suffering from DEPRESSION :   NO
Are you suffering from FATIGUE :   NO
Are you suffering from MEMORY_LOSS :   NO
Are you suffering from PARANOID :   NO
Are you suffering from DIFFICULTY_PLANNING :   NO
Are you suffering from SOCIAL_ISOLATION :   NO
Are you suffering from AMNESIA :   NO
Are you suffering from TREMOR :   YES
Are you suffering from IMPULSIVE :   NO
You are diagnosed with : General_Anxiety_Disorder

Treatment for the disorder General_Anxiety_Disorder is : Treatment consists of Anti-Depressants, SSRI, Anxiolytic Medici
ne and CBT. Medications to increase dopamine, Medications can help control the symptoms of Parkinson's.

Process returned 0 (0x0)   execution time : 94.377 s
Press any key to continue.
```

10.4 Sample Run4:

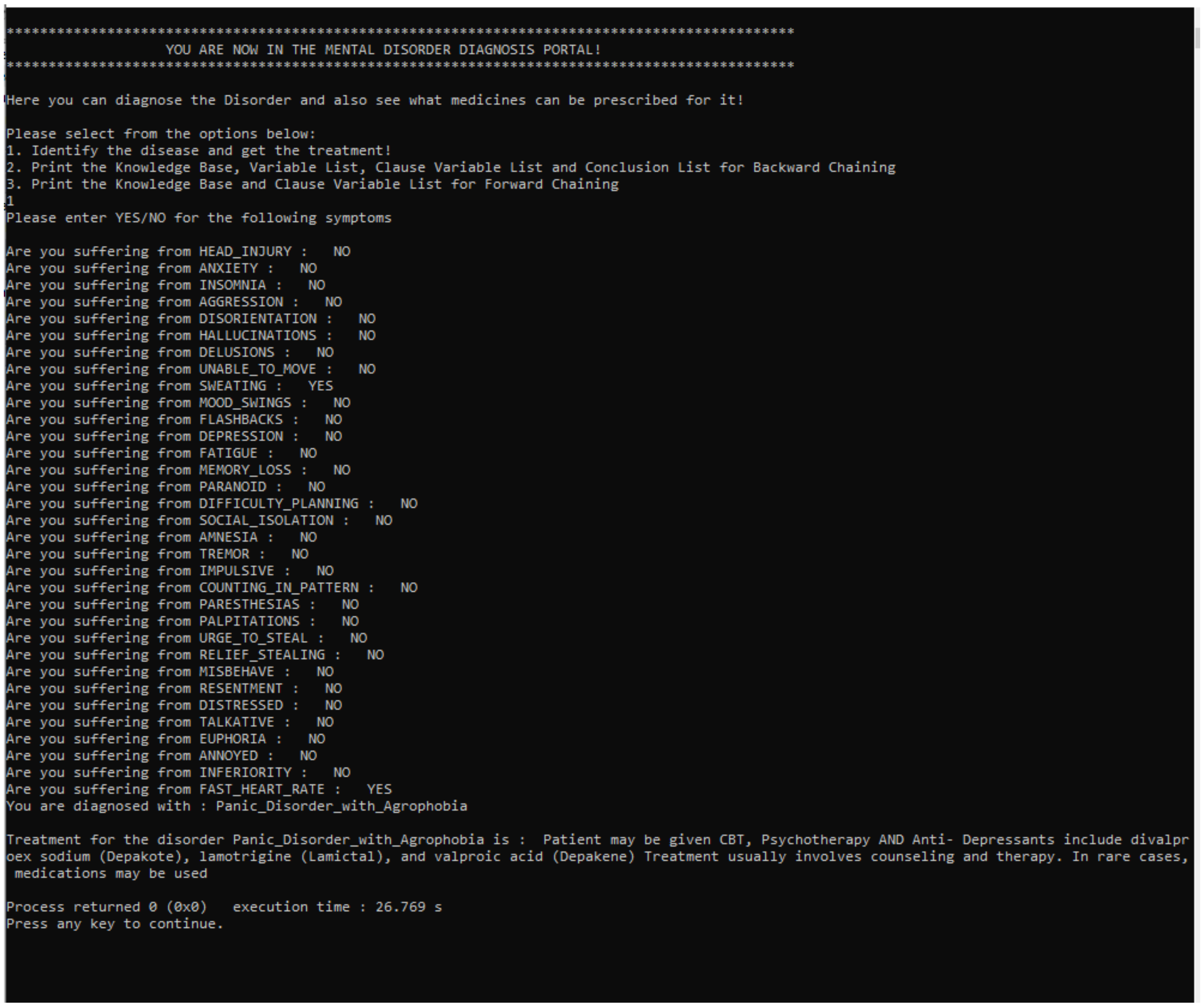
BACKWARD RULE:

IF HEAD_INJURY = NO && ANXIETY = NO && AGGRESSION = NO && SOCIAL_ISOLATION = NO && FAST_HEART_RATE = YES && SWEATING = YES THEN DIAGNOSIS = Panic Disorder with Agrophobia.

FORWARD RULE:

IF DIAGNOSIS=Panic_Disorder_with_Agrophobia THEN TREATMENT= Patient may be given CBT, Psychotherapy AND Anti- Depressants include divalproex sodium (Depakote), lamotrigine (Lamictal), and valproic acid (Depakene) Treatment usually involves counseling and therapy. In rare cases, medications may be used

Output 4:



```
*********************************************************************************************
                   YOU ARE NOW IN THE MENTAL DISORDER DIAGNOSIS PORTAL!
*********************************************************************************************

Here you can diagnose the Disorder and also see what medicines can be prescribed for it!

Please select from the options below:
1. Identify the disease and get the treatment!
2. Print the Knowledge Base, Variable List, Clause Variable List and Conclusion List for Backward Chaining
3. Print the Knowledge Base and Clause Variable List for Forward Chaining
1
Please enter YES/NO for the following symptoms

Are you suffering from HEAD_INJURY :   NO
Are you suffering from ANXIETY :   NO
Are you suffering from INSOMNIA :   NO
Are you suffering from AGGRESSION :   NO
Are you suffering from DISORIENTATION :   NO
Are you suffering from HALLUCINATIONS :   NO
Are you suffering from DELUSIONS :   NO
Are you suffering from UNABLE_TO_MOVE :   NO
Are you suffering from SWEATING :   YES
Are you suffering from MOOD_SWINGS :   NO
Are you suffering from FLASHBACKS :   NO
Are you suffering from DEPRESSION :   NO
Are you suffering from FATIGUE :   NO
Are you suffering from MEMORY_LOSS :   NO
Are you suffering from PARANOID :   NO
Are you suffering from DIFFICULTY_PLANNING :   NO
Are you suffering from SOCIAL_ISOLATION :   NO
Are you suffering from AMNESIA :   NO
Are you suffering from TREMOR :   NO
Are you suffering from IMPULSIVE :   NO
Are you suffering from COUNTING_IN_PATTERN :   NO
Are you suffering from PARESTHESIAS :   NO
Are you suffering from PALPITATIONS :   NO
Are you suffering from URGE_TO_STEAL :   NO
Are you suffering from RELIEF_STEALING :   NO
Are you suffering from MISBEHAVE :   NO
Are you suffering from RESENTMENT :   NO
Are you suffering from DISTRESSED :   NO
Are you suffering from TALKATIVE :   NO
Are you suffering from EUPHORIA :   NO
Are you suffering from ANNOYED :   NO
Are you suffering from INFERIORITY :   NO
Are you suffering from FAST_HEART_RATE :   YES
You are diagnosed with : Panic_Disorder_with_Agrophobia

Treatment for the disorder Panic_Disorder_with_Agrophobia is :  Patient may be given CBT, Psychotherapy AND Anti- Depressants include divalpr
oex sodium (Depakote), lamotrigine (Lamictal), and valproic acid (Depakene) Treatment usually involves counseling and therapy. In rare cases,
 medications may be used

Process returned 0 (0x0)   execution time : 26.769 s
Press any key to continue.
```

10.5 Sample Run5:

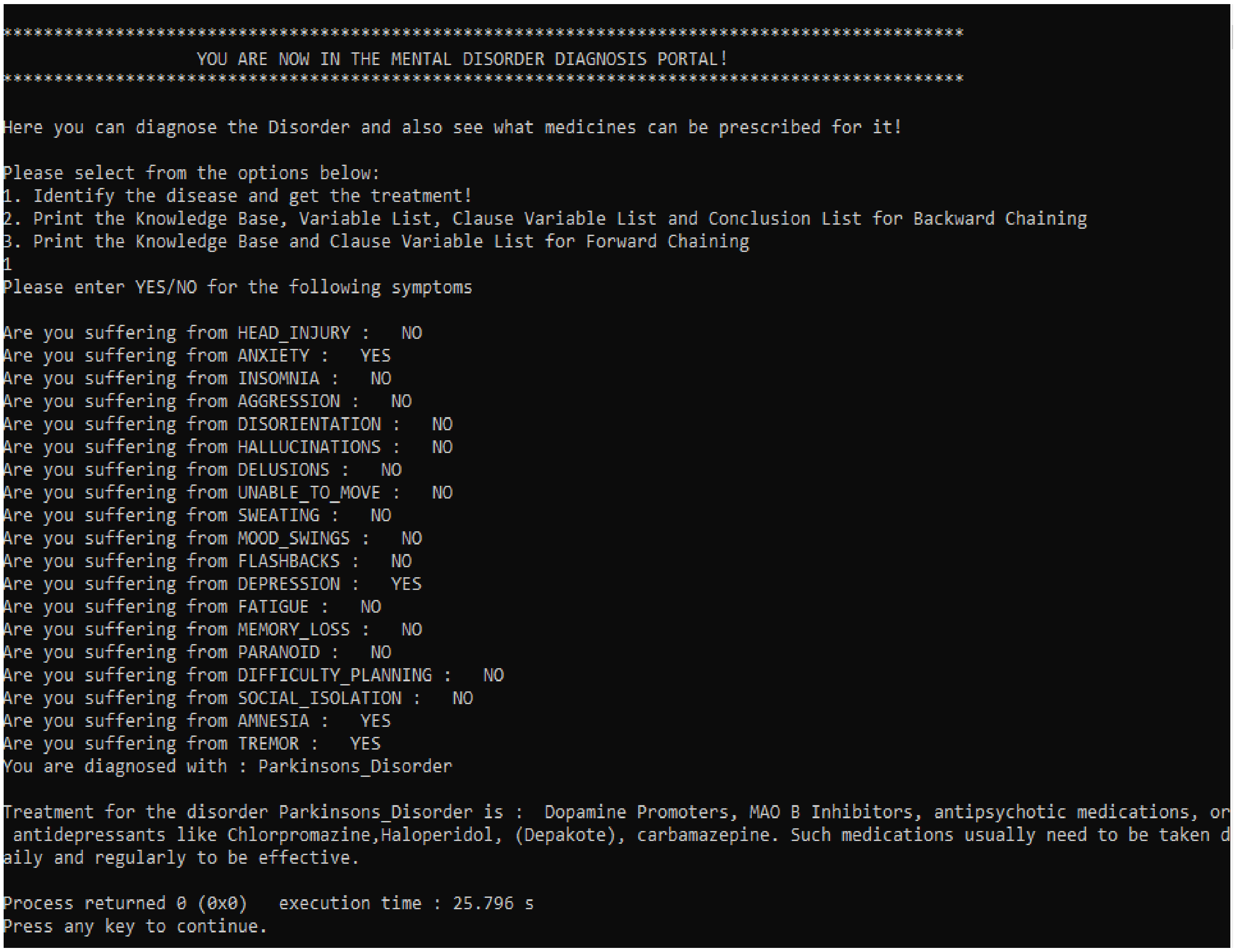
BACKWARD RULE:

IF HEAD_INJURY = NO && ANXIETY = YES && DEPRESSION = YES && DISORIENTATION = NO && AMNESIA = YES && TREMOR = YES THEN DIAGNOSIS = Parkinsons Disorder

FORWARD RULE:

IF DIAGNOSIS=Parkinsons_Disorder THEN TREATMENT= Dopamine Promoters, MAO B Inhibitors, antipsychotic medications, or antidepressants like Chlorpromazine,Haloperidol, (Depakote), carbamazepine. Such medications usually need to be taken daily and regularly to be effective.

Output 5:



```
*********************************************************************************************
                   YOU ARE NOW IN THE MENTAL DISORDER DIAGNOSIS PORTAL!
*********************************************************************************************

Here you can diagnose the Disorder and also see what medicines can be prescribed for it!

Please select from the options below:
1. Identify the disease and get the treatment!
2. Print the Knowledge Base, Variable List, Clause Variable List and Conclusion List for Backward Chaining
3. Print the Knowledge Base and Clause Variable List for Forward Chaining
1
Please enter YES/NO for the following symptoms

Are you suffering from HEAD_INJURY :   NO
Are you suffering from ANXIETY :   YES
Are you suffering from INSOMNIA :   NO
Are you suffering from AGGRESSION :   NO
Are you suffering from DISORIENTATION :   NO
Are you suffering from HALLUCINATIONS :   NO
Are you suffering from DELUSIONS :   NO
Are you suffering from UNABLE_TO_MOVE :   NO
Are you suffering from SWEATING :   NO
Are you suffering from MOOD_SWINGS :   NO
Are you suffering from FLASHBACKS :   NO
Are you suffering from DEPRESSION :   YES
Are you suffering from FATIGUE :   NO
Are you suffering from MEMORY_LOSS :   NO
Are you suffering from PARANOID :   NO
Are you suffering from DIFFICULTY_PLANNING :   NO
Are you suffering from SOCIAL_ISOLATION :   NO
Are you suffering from AMNESIA :   YES
Are you suffering from TREMOR :   YES
You are diagnosed with : Parkinsons_Disorder

Treatment for the disorder Parkinsons_Disorder is :  Dopamine Promoters, MAO B Inhibitors, antipsychotic medications, or
 antidepressants like Chlorpromazine,Haloperidol, (Depakote), carbamazepine. Such medications usually need to be taken d
aily and regularly to be effective.

Process returned 0 (0x0)   execution time : 25.796 s
Press any key to continue.
```

11. ANALYSIS OF THE PROGRAM AND RESULTS

11.1 Problems with existing code:

- Firstly, we started analyzing the existing code. We found it very difficult to analyze or to understand the given code, so the readability of the given code is not good.
- The given code is hard to modify and maintain as there is lack of separation of functionality in the program into distinct and independent units and causes difficulty in debugging.
- Existing code is written in C language, So it cannot be reused and not flexible as C is not object oriented language.
- Existing code uses many global variables, which can be read and modified by any part of the program, which is not a good practice in programming and it uses more memory.
- GOTO statements in the existing code affects the performance and efficiency.
- Existing code uses many switch cases which affects the optimization. i.e. if we have thousands of rules, we cannot write all switch case statements for them.
- We need to use the variable list and clause variable list in a efficient way. In the given code they are stored in the program which increases the code lines which is not necessary.

11.2 Proposed solution:

- We analyzed the flaws with the existing code which made us to build the entire system from scratch.
- After collecting the information about the Diseases, symptoms and their treatments we choose diseases which will have at least one unique symptom which is the main criteria as we cannot find disease or its treatment if we have all common symptoms.
- As our goal is to develop a system which has the following advantages like Modularity, Understandability, Usability, User friendly, efficient, Extensibility.

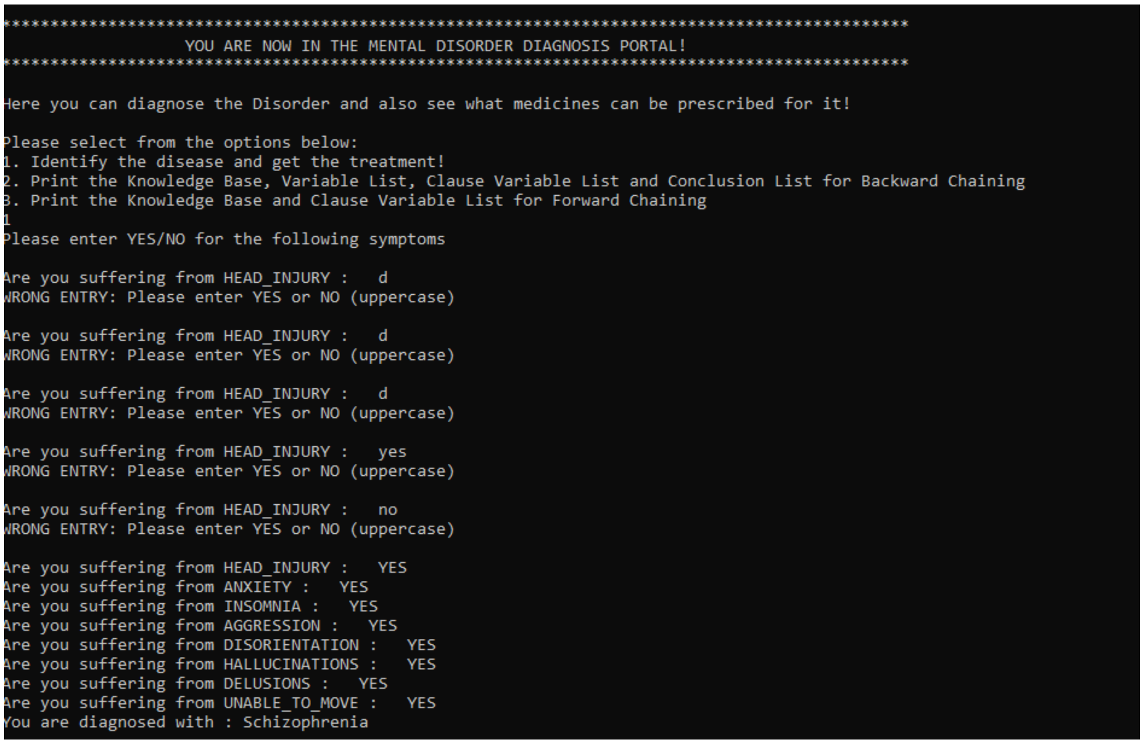
Our code has the following advantages:

- **Modularity:**
  As the given code was not correctly organized, we started dividing the lines of code into a smaller units or modules or functions. Functions are important because they allow us to divide large complicated program into smaller manageable pieces. Then, we concentrated on building the required functions, we tested each of them to make sure functions are working properly.

- **Understandable:**
  We found it really hard to determine how a given/ existing code is working after reading the source code. Then, we decided to develop a code which is more understandable. Understandability of the code helps in manual analysis of source code and makes it easier to spot defects and vulnerabilities.

- **Usability:**
  We saved the all the data of diseases and their treatments on the permanent memory which can be used by specialized doctors for analysis and improvements.

- **User friendliness:**
  We have designed an interface engine in such a way that the user can select the conclusion variables from the conclusion list which are displayed on the screen such that patents with zero knowledge on mental illness can also use our Expert system also it saves the patients time. It also helps the patient for the treatment of the illness and helps them to start the treatment or prescriptions as soon as possible.
  If the user enters the wrong choice(other than YES or NO), Expert system ask the user to enter the correct choice which is shown below.



```
*********************************************************************************************
                   YOU ARE NOW IN THE MENTAL DISORDER DIAGNOSIS PORTAL!
*********************************************************************************************

Here you can diagnose the Disorder and also see what medicines can be prescribed for it!

Please select from the options below:
1. Identify the disease and get the treatment!
2. Print the Knowledge Base, Variable List, Clause Variable List and Conclusion List for Backward Chaining
3. Print the Knowledge Base and Clause Variable List for Forward Chaining
1
Please enter YES/NO for the following symptoms

Are you suffering from HEAD_INJURY :   d
WRONG ENTRY: Please enter YES or NO (uppercase)

Are you suffering from HEAD_INJURY :   d
WRONG ENTRY: Please enter YES or NO (uppercase)

Are you suffering from HEAD_INJURY :   d
WRONG ENTRY: Please enter YES or NO (uppercase)

Are you suffering from HEAD_INJURY :   yes
WRONG ENTRY: Please enter YES or NO (uppercase)

Are you suffering from HEAD_INJURY :   no
WRONG ENTRY: Please enter YES or NO (uppercase)

Are you suffering from HEAD_INJURY :   YES
Are you suffering from ANXIETY :   YES
Are you suffering from INSOMNIA :   YES
Are you suffering from AGGRESSION :   YES
Are you suffering from DISORIENTATION :   YES
Are you suffering from HALLUCINATIONS :   YES
Are you suffering from DELUSIONS :   YES
Are you suffering from UNABLE_TO_MOVE :   YES
You are diagnosed with : Schizophrenia
```

- **Extensibility:**

  For storing all the variable list, conclusion list and Clause variable list we have created the files So that in future if we want to increase the size of variable list, conclusion list and clause variable list we need to change only the size declared and extra variables can be added to respective files directly.

- **Efficiency:**
  - ✓ The most important part of the project is to improve the efficiency of the given code by using expert systems. By keeping this in mind we designed our expert system which gives the desired results in limited amount of time.
  - ✓ The given code was written in C, we have used C++ which is Object Oriented Language.
  - ✓ GOTO statements and Global variables are not used.
  - ✓ We used classes and objects.

12. CONCLUSION

The expert system we build as part of this project was for the hospital. This project has helped us understand the logic behind backward chaining and forward chaining. We have also paid attention to build an efficient and user friendly system. I was able to get a better idea and understanding in designing a decision tree through this project. Above all I was able to follow the design principles while coding the program for the better readability , maintainability and for the efficiency as well.

13. REFERENCES

Russell, Stuart, and Norvig, Peter. Artificial Intelligence, A Modern Approach. 3rd ed. Upper Saddle River, New Jersey, 2010

Robert M. Kliegman, Bonita M.D. Stanton, Joseph St. Geme, Nina F Schor: Nelson Texbook of pediatrics.

https://blog.doctorondemand.com/why-its-important-to-care-for-your-mental-health-34c8670b889

https://en.wikipedia.org/wiki/Backward_chaining

https://en.wikipedia.org/wiki/Forward_chaining

https://en.wikipedia.org/wiki/Decision_tree

https://www.mayoclinic.org/diseases-conditions/mental-illness/symptoms-causes/syc-20374968