CS 5346: ADVANCED ARTIFICIAL INTELLIGENCE

EXPERT SYSTEM TO DIAGNOSE MENTAL DISORDERS

AN EXPERT SYSTEM FOR A MENTAL HEALTH CLINIC TO DIAGNOSE MENTAL DISORDERS AND RECOMMEND TREATMENT

AUTHOR: VIDHYASHREE NAGABHUSHANA
NET ID: V_N63
STUDENT ID: A04927921

PROJECT BY:
JAIN, AKSHATHA MANOHAR
SUNDARA RAJ SREENATH, SAHANA
NAGABHUSHANA, VIDHYASHREE

TABLE OF CONTENTS

1.INTRODUCTION 4
1.1 THE PROBLEM DESCRIPTION 4
1.2 THE PROBLEM SOLUTION 4
2.TEAM CONTRIBUTIONS 5
2.1 TEAM MEMBERS 5
2.2 CONTRIBUTIONS 5
3.THE DOMAIN 6
4.METHODOLOGIES 6
3.1 BACKWARD CHAINING 6
3.2 FORWARD CHAINING 7
5.DESIGNING THE KNOWLEDGE BASE 8
5.1 DECISION TREE 8
5.1.1 BACKWARD CHAINING DECISION TREE 9
5.1.2 FORWARD CHAINING DECSION TREE 10
5.2 RULES 11
5.2.1 RULES FOR BACKWARD CHAINING 11
5.2.2 RULES FOR FORWARD CHAINING 15
6.PROGRAM IMPLEMENTATION 19
6.1 BACKWARD CHAINING 19
6.2 FORWARD CHAINING 36
7.SOURCE CODE 40
7.1 BACKWARDCHAINING.H 40
7.2 BACKWARDCHAINING.CPP 42
7.3 FORWARD CHAINING HEADER FILE 65
7.4 FORWARDCHAINING.CPP 66
7.5 MAIN DRIVER PROGRAM 72
8.SAMPLE RUN 77
9.ANALYSIS OF PROGRAM AND RESULT 81
9.1 PROBLEM WITH EXISTING CODE 81
9.2 PROPOSED SOLUTION 81

10.CONCLUSION 82
11.REFERENCES 83

1 INTRODUCTION

1.1 THE PROBLEM DESCRIPTION

Good mental health is the key to a balanced and healthy life. With a good mental health, our physical, emotional and social well-being can be taken care of. The impact of mental health to a person is a lot. The way we think, our decision making and the way we behave each day is an impression of how good our mental health is. Since our mental health is one of the most important aspects for a happy life, we should always make sure to have a healthy mental state. It is equally important to diagnose if there are changes in a person's mood, behavior and decision to make sure if the person is suffering from a mental disorder. An early diagnosis and treatment can help heal the patient better. Having to go to a doctor every time changes are observed and having to get lab tests done each time may get a lot tedious. To make the diagnosis a little easier, in this project, we have developed an expert system which can be used in mental health clinics to diagnose the mental disorder, if any, and recommend the possible medications that can be prescribed for the disorder.

1.2 THE PROBLEM SOLUTION

Developing an expert system will be a good way to make sure the mental health clinics can diagnose the disorder mostly with verbal conversations with the patient or guardian rather than having to go with lab tests for each disorder. The verbal conversation will include the clinical staff asking the patient/guardian a set of questions which would be about symptoms and based on the answers obtained, the diagnosis can be done. Backward Chaining can be used for Diagnosis alongside decision tree and five important data structures. The answers given by the patient/guardian are compared to the available rules in the Knowledge Base and a diagnosis result is given. Once the diagnosis is done successfully, we can get the treatment using Forward Chaining.

2 TEAM CONRIBUTIONS

2.1 TEAM MEMBERS

- Jain, Akshatha Manohar
- Sundara Raj Sreenath, Sahana
- Nagabhushana, Vidhyashree

2.2 CONTRIBUTIONS

AKSHATHA MANOHAR JAIN

- She collected information from the internet for a few 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 changes/ suggestions.
- Discussed and implemented the Backward Chaining and Forward Chaining functions in C++.
- Done the testing of Forward chaining function.
- Done the testing of backward chaining function.

VIDHYASHREE NAGABHUSHANA

- Collected information from the internet for a few mental disorders, treatments and their symptoms.
- Helped in giving base idea of how to approach decision tree.
- Created and finalized Backward Chaining Decision tree.
- Drew the decision tree for Forward Chaining and Backward Chaining using Lucid Charts.
- Formulated the IF – THEN rules for Backward Chaining.
- Discussed and implemented the Backward Chaining and Forward Chaining functions in C++.
- Done the testing of Forward chaining function.
- Done the testing of backward chaining function.

SAHANA SREENATH

- 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.
- Discussed and implemented the Backward Chaining and Forward Chaining functions in C++.
- Done the testing of Forward chaining function.
- Done the testing of backward chaining function.

3 THE DOMAIN

The system domain of the problem is the Mental Health Clinic. The main goal of the project is to develop an expert system to diagnose a mental disorder given symptoms from the patient/guardian using the Backward Chaining technique and recommending treatment for the same using Forward Chaining in Mental Health Clinics.

4 METHODOLOGIES

4.1 BACKWARD CHAINING

Backward Chaining is Goal Driven. Backward chaining is an inference method described colloquially as working backward from the goal. It is known as a goal-driven approach as the objective of Backward Chaining is to always prove the **GOAL**. We start from the goal and try tracing back to what caused the goal. In Backward Chaining, the THEN part of the IF-THEN clause, i.e., the consequent is considered first and the flow works backwards from the consequent to the variables in the IF clause, i.e., the antecedents to see if any data supports any of the consequents. Later, the rules are searched to see if the antecedents link to earlier consequents. The chain of

reasoning continues until a rule that satisfies the conditions is met or all rules are visited, and no rule is satisfied. Whichever happens first.

In our expert system, we are using Backward Chaining to diagnose a mental disorder. The Disorder is the consequent, i.e., in the THEN clause and the symptoms that cause the disorder are the antecedents. We ask the patient/guardian if the patient is suffering from a set of symptoms and instantiate the variables as and when answers are received. Then, the knowledge base is checked to see if any rule is satisfied from the information gathered and once the rule is satisfied, the corresponding Disorder is diagnosed.

4.2 FORWARD CHAINING

Forward Chaining is Fact Driven. In Forward chaining, we infer unknown truths from a known data and move forward using a set of rules to find the solution. Forward Chaining always sees if it can execute any available rules and infer the solution with the available data. Forward Chaining being fact driven and a bottom up form of logic, it always starts with a known set of rules and conditions, progresses towards a conclusion using the IF-THEN rules until the limit is reached or until the applicable rule is met. On the contrary to Backward Chaining, Forward Chaining starts with the antecedents, gets the values and later progresses to the consequent only after reasoning its way through the answer after checking the set of rules.

In our expert system, we are using Forward Chaining to recommend treatment for the disorder diagnosed in Backward Chaining. Here, the Treatment is the consequent, i.e., the medicines to be recommended for a disorder is in the THEN clause and the disorder for which the treatment has to be recommended will be the antecedent, i.e., the IF clause will have the disorder for which treatment is to be recommended in the Knowledge Base. The Disorder is checked, and the corresponding treatment is recommended.

5 DESIGNING THE KNOWLEDGE BASE

Designing the knowledge base is one of the most important aspects to building an expert system. It is based on the knowledge base design that we will be able to diagnose the disorder and recommend a treatment for the diagnosed disorder. To design the knowledge base, we as a team of three did extensive research on the Web to gather information about a few mental disorders, their symptoms and possible treatments. All the collected information was organized in a way to build the Decision tree and the rules, which are the most important steps to build and design the knowledge base.

5.1 DECISION TREE

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements. The decision tree branches off just like a tree, and at the very end of each branch or system of branches is a conclusion. The Decision tree consists of *circles* and *rectangles* called *nodes*. The arrow lines that connect these nodes are known as *branches.* The circles which contain questions are *decision nodes*. The rectangles contain the goals of the diagram, i.e., *the conclusions*. Many of the nodes have branches leaving them, providing pathways to other nodes. The path taken from each node is determined by the response to the condition contained in the node's drawing. Below is an example of a decision tree. In Backward Chaining, the rectangles are the Disorders. The symptoms are represented by circles.

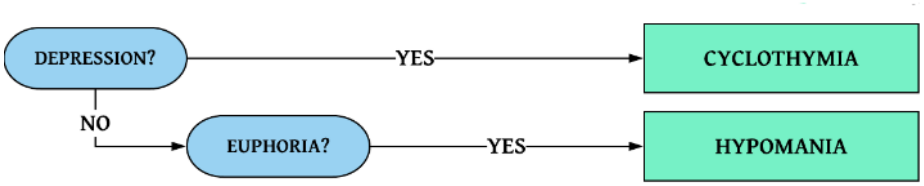
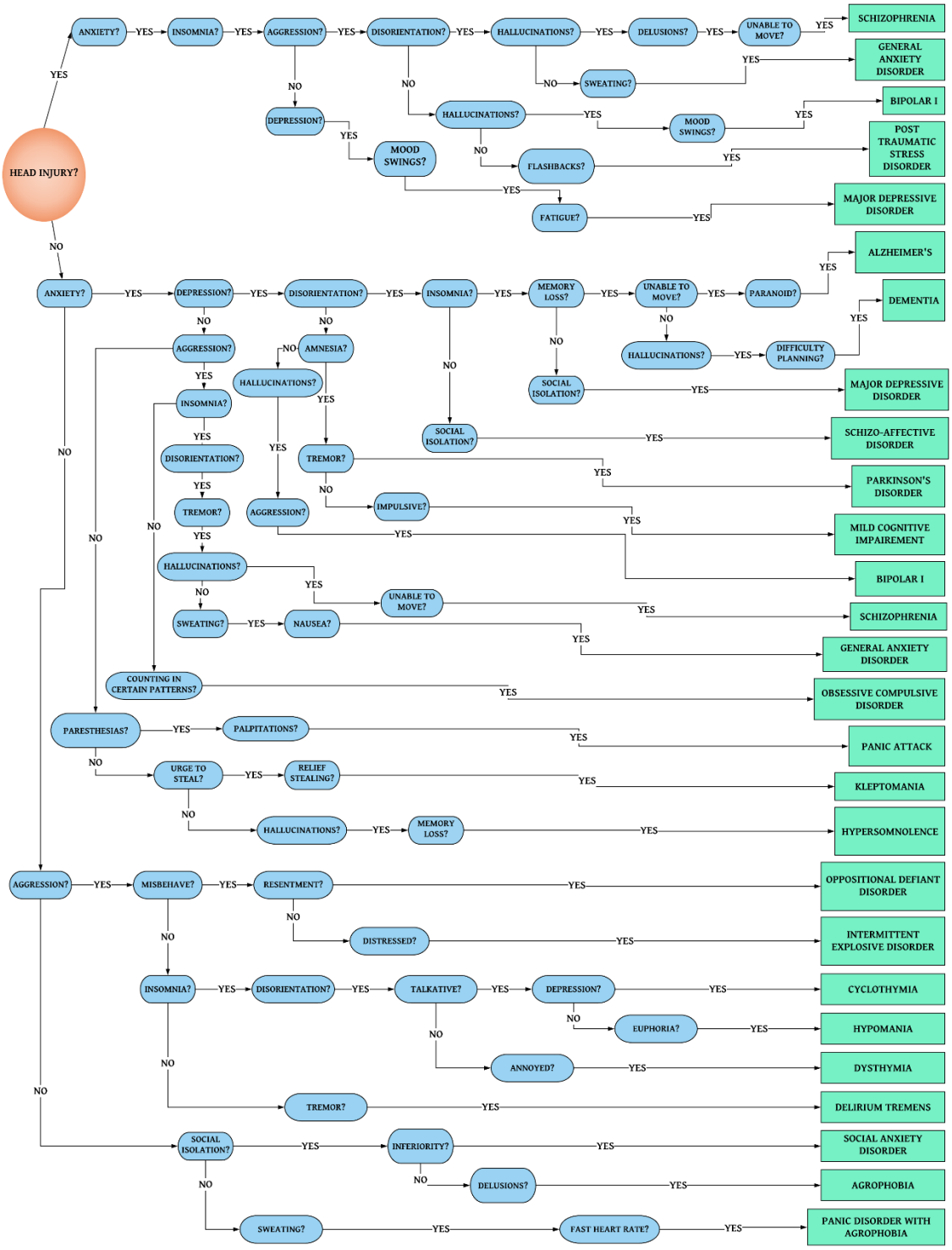


Fig. 1. Example of a Decision Tree

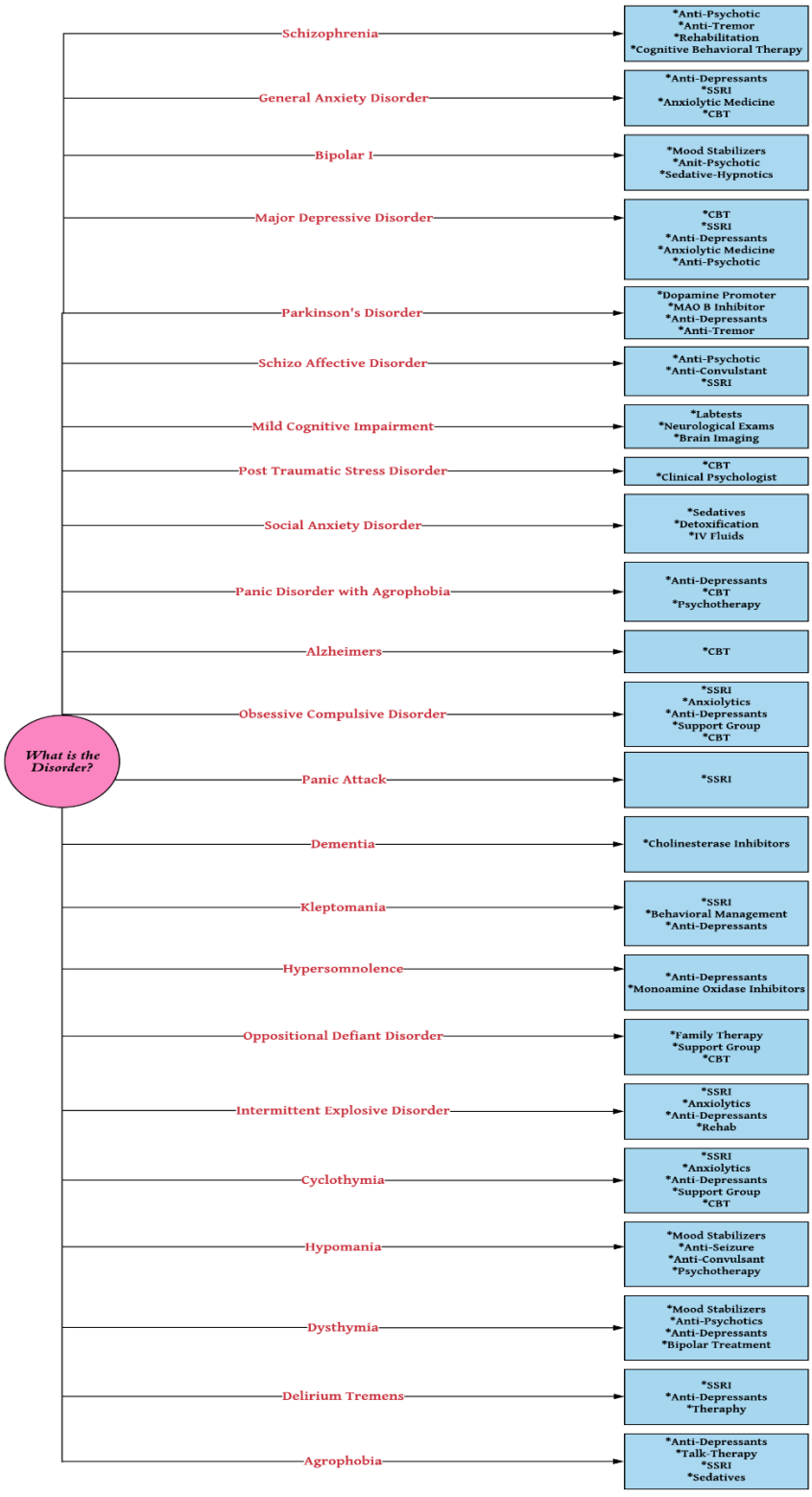
5.1.1BACKWARD CHAINING DECISION TREE

The decision tree for Backward Chaining is developed as below:



5.1.2FORWARD CHAINING DECISION TREE

The Decision tree for Forward Chaining is as below:



5.2 RULES

We now formulate the rules from the above decision trees for both Backward and Forward Chaining.

5.2.1 RULES FOR BACKWARD CHAINING

The IF-THEN rules are made up of two parts. 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. The THEN part is evaluated only if the IF part is true.

RULE NUMBER	RULE
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

5.2.2RULES FOR FORWARD CHAINING

RULE NUMBER	RULE
10	IF DIAGNOSIS=Schizophrenia THEN TREATMENT= Anti-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 Razadyne (galantamine), Exelon (rivastigmine), and Aricept (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, Anti-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=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.
100	IF DIAGNOSIS=Mild_Cognitive_Impairment THEN TREATMENT= Treatment involves Labtests, 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 PROGRAM IMPLEMENTATION

6.1 BACKWARD CHAINING

Backward Chaining is Goal Driven. Backward chaining is an inference method described colloquially as working backward from the goal. It is known as a goal-driven approach as the objective of Backward Chaining is to always prove the **GOAL.** In our expert system, we are using Backward Chaining to diagnose a mental disorder. The Disorder is the consequent, i.e., in the THEN clause and the symptoms that cause the disorder are the antecedents. We ask the patient/guardian if the patient is suffering from a set of symptoms and instantiate the variables as and when answers are received. Then, the knowledge base is checked to see if any rule is satisfied

from the information gathered and once the rule is satisfied, the corresponding Disorder is diagnosed.

We have used five data structures to implement the Backward Chaining Inference Engine. They are as below:

VARIABLE LIST

The Variable list is implemented using a structure “variable”.

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

This structure “variable” has three items. “varname” is the name of the variable in the variable list. The variable list is obtained from the knowledge base. Each variable in the IF part act as variables to the variable list. If multiple rules have the same variables, the variable is included only once in the Variable List as values in the list are unique. “varstatus” is the status of the corresponding variable. The status can be NI/I. Initially, all variables will be in NI state, i.e., Not- Instantiated state. It gets updated to I or Instantiated as and when the corresponding variables are set to a value. “varvalue” is the value a variable has. Initially, all the variables have a “varvalue” of “ “ which is later updated to YES/NO according to the responses for the symptoms.

In our expert system, we have implemented the variable list as an array of structs of type “variable”.

```
class BackwardChaining
{
public:
    //variables
    variable variableList[33];
```

The variable list is stored in the Variable_List.txt file and the values are read from the file to the array of structures for our variable list.

```
//Initializing variable list
ifstream fin;
string name, status, value;
fin.open("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 = "";
}
```

Below is the Variable List:

VARIABLE NAME	VARIABLE STATUS
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

CLAUSE VARIABLE LIST

The Clause variable list consists of all the variables in the IF part of the IF-THEN rule. Each rule will have some allocated space for all the variables in that rule. In our system, we have considered the allocated space to be 11 for each rule. If some slot is left blank, those are marked "b". All the clauses in the IF part are connected by logical AND in our system. Hence, all the variables in the list must first be instantiated for the THEN clause to be executed.

To compute the clause number when the rule number is known, the below formula was used.

Clause number = 11 * ((rule number / 10) – 1) + 1

Example: If the rule number is 150, for our system, the clause number will then be:

11 ((150/10) – 1) + 1 = 155

In our system, the clause variable list is implemented as an array of strings.

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

The list is stored in a file and are read to the array as shown below:

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

Below is the Clause Variable List:

CLAUSE VARIABLE LIST
HEAD_INJURY
ANXIETY
INSOMNIA
AGGRESSION
DISORIENTATION
HALLUCINATIONS
DELUSIONS
UNABLE_TO_MOVE
b
b
b

HEAD_INJURY
ANXIETY
INSOMNIA
AGGRESSION
DISORIENTATION
HALLUCINATIONS
SWEATING
b
b
b
b
HEAD_INJURY
ANXIETY
INSOMNIA
AGGRESSION
DISORIENTATION
HALLUCINATIONS
MOOD_SWINGS
b
b
b
b
HEAD_INJURY
ANXIETY
INSOMNIA
AGGRESSION
DISORIENTATION
HALLUCINATIONS
FLASHBACKS
b
b
b
b

HEAD_INJURY
ANXIETY
INSOMNIA
AGGRESSION
DEPRESSION
MOOD_SWINGS
FATIGUE
b
b
b
b
HEAD_INJURY
ANXIETY
DEPRESSION
DISORIENTATION
INSOMNIA
MEMORY_LOSS
UNABLE_TO_MOVE
PARANOID
b
b
b
HEAD_INJURY
ANXIETY
DEPRESSION
DISORIENTATION
INSOMNIA
MEMORY_LOSS
UNABLE_TO_MOVE
HALLUCINATIONS
DIFFICULTY_PLANNING
b
b

HEAD_INJURY
ANXIETY
DEPRESSION
DISORIENTATION
INSOMNIA
MEMORY_LOSS
SOCIAL_ISOLATION
b
b
b
b
HEAD_INJURY
ANXIETY
DEPRESSION
DISORIENTATION
INSOMNIA
SOCIAL_ISOLATION
b
b
b
b
b
HEAD_INJURY
ANXIETY
DEPRESSION
DISORIENTATION
AMNESIA
TREMOR
b
b
b
b
b

HEAD_INJURY
ANXIETY
DEPRESSION
DISORIENTATION
AMNESIA
HALLUCINATIONS
AGGRESSION
b
b
b
b
HEAD_INJURY
ANXIETY
DEPRESSION
DISORIENTATION
AMNESIA
TREMOR
IMPULSIVE
b
b
b
b
HEAD_INJURY
ANXIETY
DEPRESSION
AGGRESSION
INSOMNIA
DISORIENTATION
TREMOR
HALLUCINATIONS
UNABLE_TO_MOVE
b
b

HEAD_INJURY
ANXIETY
DEPRESSION
AGGRESSION
INSOMNIA
DISORIENTATION
TREMOR
HALLUCINATIONS
SWEATING
b
b
HEAD_INJURY
ANXIETY
DEPRESSION
AGGRESSION
INSOMNIA
COUNTING_IN_PATTERN
b
b
b
b
b
HEAD_INJURY
ANXIETY
DEPRESSION
AGGRESSION
PARESTHESIAS
PALPITATIONS
b
b
b
b
b

HEAD_INJURY
ANXIETY
DEPRESSION
AGGRESSION
PARESTHESIAS
URGE_TO_STEAL
RELIEF_STEALING
b
b
b
b
HEAD_INJURY
ANXIETY
DEPRESSION
AGGRESSION
PARESTHESIAS
URGE_TO_STEAL
HALLUCINATIONS
MEMORY_LOSS
b
b
b
HEAD_INJURY
ANXIETY
AGGRESSION
MISBEHAVE
RESENTMENT
b
b
b
b
b
b

HEAD_INJURY
ANXIETY
AGGRESSION
MISBEHAVE
RESENTMENT
DISTRESSED
b
b
b
b
b
HEAD_INJURY
ANXIETY
AGGRESSION
MISBEHAVE
INSOMNIA
DISORIENTATION
TALKATIVE
DEPRESSION
b
b
b
HEAD_INJURY
ANXIETY
AGGRESSION
MISBEHAVE
INSOMNIA
DISORIENTATION
TALKATIVE
DEPRESSION
EUPHORIA
b
b

HEAD_INJURY
ANXIETY
AGGRESSION
MISBEHAVE
INSOMNIA
DISORIENTATION
TALKATIVE
ANNOYED
b
b
b
HEAD_INJURY
ANXIETY
AGGRESSION
MISBEHAVE
INSOMNIA
TREMOR
b
b
b
b
b
HEAD_INJURY
ANXIETY
AGGRESSION
SOCIAL_ISOLATION
INFERIORITY
b
b
b
b
b
b

HEAD_INJURY
ANXIETY
AGGRESSION
SOCIAL_ISOLATION
INFERIORITY
DELUSIONS
b
b
b
b
b
HEAD_INJURY
ANXIETY
AGGRESSION
SOCIAL_ISOLATION
FAST_HEART_RATE
SWEATING
b
b
b
b
b

CONCLUSION LIST

The conclusion list is implemented using a structure “conclusionlist”.

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

The conclusionlist structure has two items, the “rulenum” and “conclusionname”. In the conclusion list, we list all the possible conclusions for each rule, i.e., all the consequents in the THEN part are stored in the conclusion list along with the respective rule numbers. The conclusions are stored in sequential orders in the conclusion list.

In our expert system, the conclusion list is implemented as an array of structs of type conclusionlist.

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

The values for conclusion list are stored in a text file and are read to the array of structs.

```
//Initializing conclusion list
ifstream fin1;
int rulenum;
string conclusionname;
fin1.open("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;
}
```

The conclusion list is as below:

RULE NUMBER	CONCLUSION
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

CONCLUSION STACK

Conclusion stack is one of the most important data structures in backward chaining. It consists of the rule number and the computed clause number. The conclusion stack is the central structure: It ties together all the other structures in the implementation of the backward chaining expert system tool. 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;
```

IMPLEMENTATION:

The Backward Chaining flow goes as below:

- As we execute the program, we first create an object of BackwardChaining class, and this calls the default constructor.

- Once the default constructor is invoked, the variable list, conclusion list and clause variable lists are initialized.
- Next, the first instance from the conclusion list is fetched and checked if the conclusion is "DIAGNOSIS", which is what we want. Once it finds a match to "DIAGNOSIS" in the conclusion list, the rule number is placed in the conclusion stack and 1 to represent the clause number.

  Formula to compute the Clause Number:
  Clause Number = 11*((rule number / 10) -1) +1
  If it is not found, we notify the user that the answer is not found.

- Instantiate all the variables in the IF clause of the statement.
- If one of the IF clause variables is not instantiated, and is not a consequent, ask the user to enter a value.
- If the statement on the 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.
- If such a statement is found, then instantiate each clause variables of the IF part in the IF-THEN statement.
- 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, if the statement on the top of the stack cannot be instantiated using the present IFTHEN statement, remove the unit from the top of the stack and search the conclusion list for another instance of that conclusion variable's name.
- 9. If the rule on the 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, instantiate each clause variables of the IF part in the IF-THEN statement.

- If no other conclusion variable is underneath, we have answered our question. Hence, we can conclude our findings about the Disorder.

6.2 FORWARD CHAINING

Forward Chaining is Fact Driven. In Forward chaining, we infer unknown truths from a known data and move forward using a set of rules to find the solution. Forward Chaining always sees if it can execute any available rules and infer the solution with the available data. In our expert system, we are using Forward Chaining to recommend treatment for the disorder diagnosed in Backward Chaining. Here, the Treatment is the consequent, i.e., the medicines to be recommended for a disorder is in the THEN clause and the disorder for which the treatment has to be recommended will be the antecedent, i.e., the IF clause will have the disorder for which treatment is to be recommended in the Knowledge Base. The Disorder is checked, and the corresponding treatment is recommended.

The data structures for Forward Chaining are as below:

CLAUSE VARIABLE LIST

Clause variable list is just like the one in the Backward Chaining. It consists of the IF part of the Forward Chaining IF-THEN rules.

In our expert system, we have declared the Clause Variable List as an array of strings.

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

The clause variable list is as below:

DIAGNOSIS
b
b
DIAGNOSIS
b
b
DIAGNOSIS
b
b
DIAGNOSIS
b
b
DIAGNOSIS
b
b
DIAGNOSIS
b
b
DIAGNOSIS
b
b
DIAGNOSIS
b
b
DIAGNOSIS
b
b
DIAGNOSIS
b

b
DIAGNOSIS
b
b
DIAGNOSIS
b
b
DIAGNOSIS
b
b
DIAGNOSIS
b
b
DIAGNOSIS
b
b
DIAGNOSIS
b
b
DIAGNOSIS
b
b
DIAGNOSIS
b
b
DIAGNOSIS
b
b
DIAGNOSIS
b
b
DIAGNOSIS

b
b
DIAGNOSIS
b
b
DIAGNOSIS
b
b

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 C++ STL queue of string type for conclusion variable queue.

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

VARIABLE LIST

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”.

IMPLEMENTATION

The flow of forward chaining goes as follows:

- We create an object of the ForwardChaining class. The default constructor is invoked in which we initialize the Clause Variable List.
- We then invoke the diagnosis_treatment function for which the disorder that was found earlier in Backward Chaining is passed.
- Based on the Disorder, the condition is identified.
- The condition variable is placed on the conclusion variable queue and its value is marked on the variable list.

- 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, when there are no more IF statements containing the variable that is at the front of the conclusion variable queue, that variable is removed.
- 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.
- The instantiated THEN part of the variable is placed in the back of the conclusion variable queue. When there are no more IF statements containing the variable that is at the front of the conclusion variable queue, that variable is removed.
- If there are no more variables on the conclusion variable queue, end the session.
- If there are more variables, 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 and the process repeats.

7 SOURCE CODE

7.1 BACKWARDCHAINING.H

```
#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
```

7.2 BACKWARDCHAINING.CPP

```
#include "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("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("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("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("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;
}
```

7.3 FORWARD CHAINING HEADER FILE

```
#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
```

7.4 FORWARDCHAINING.CPP

```
#include "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("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("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;
}
```

7.5 MAIN DRIVER PROGRAM

```
#include <iostream>
#include "BackwardChaining.h"
#include "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;
```

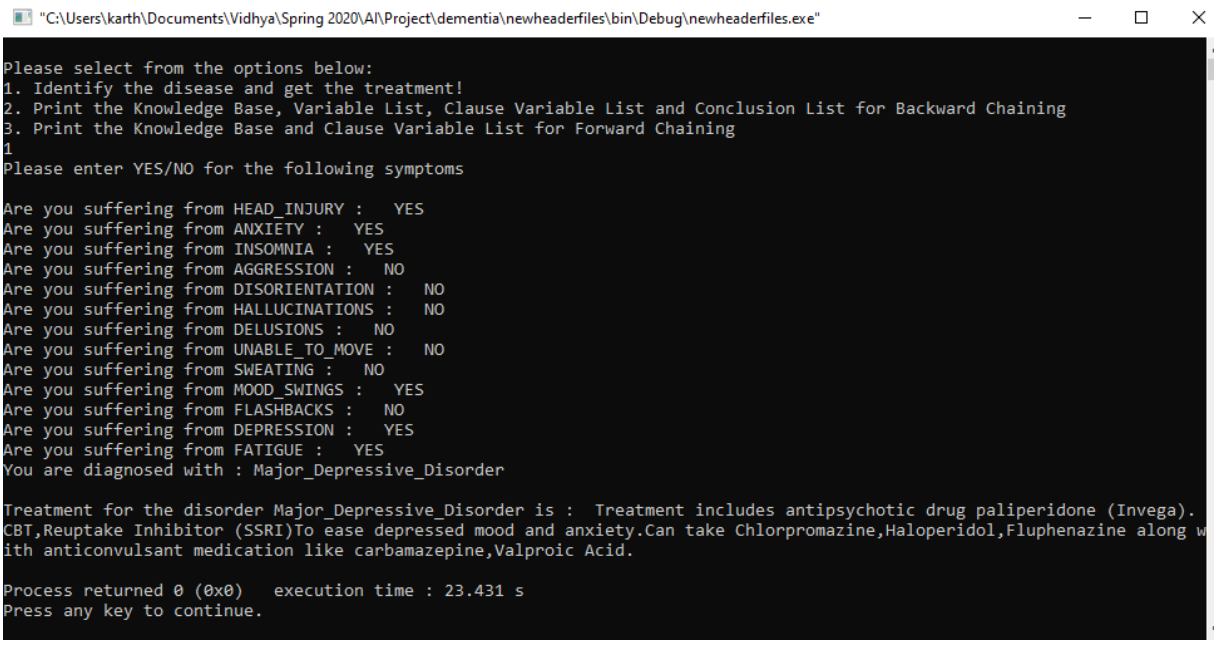
```
            break;
        }
        case 3:{
            cout                                                                    <<
"*****************************************************************
*********************************" << endl;
            cout << "The Knowledge Base for Backward 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;}
```

8 PROGRAM RUN

RUN 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 :   YES
Are you suffering from INSOMNIA :   YES
Are you suffering from AGGRESSION :   NO
Are you suffering from DISORIENTATION :   YES
Are you suffering from HALLUCINATIONS :   YES
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 :   YES
Are you suffering from PARANOID :   NO
Are you suffering from DIFFICULTY_PLANNING :   YES
You are diagnosed with : Dementia

Treatment for the disorder Dementia is :  Major treatment is Cholinesterase Inhibitors. Medications may include:lithium
(Lithobid), valproic acid (Depakene), divalproex sodium (Depakote), carbamazepine (Tegretol, Equetro, others) and lamotr
igine (Lamictal).

Process returned 0 (0x0)   execution time : 65.417 s
Press any key to continue.
```

RUN 2



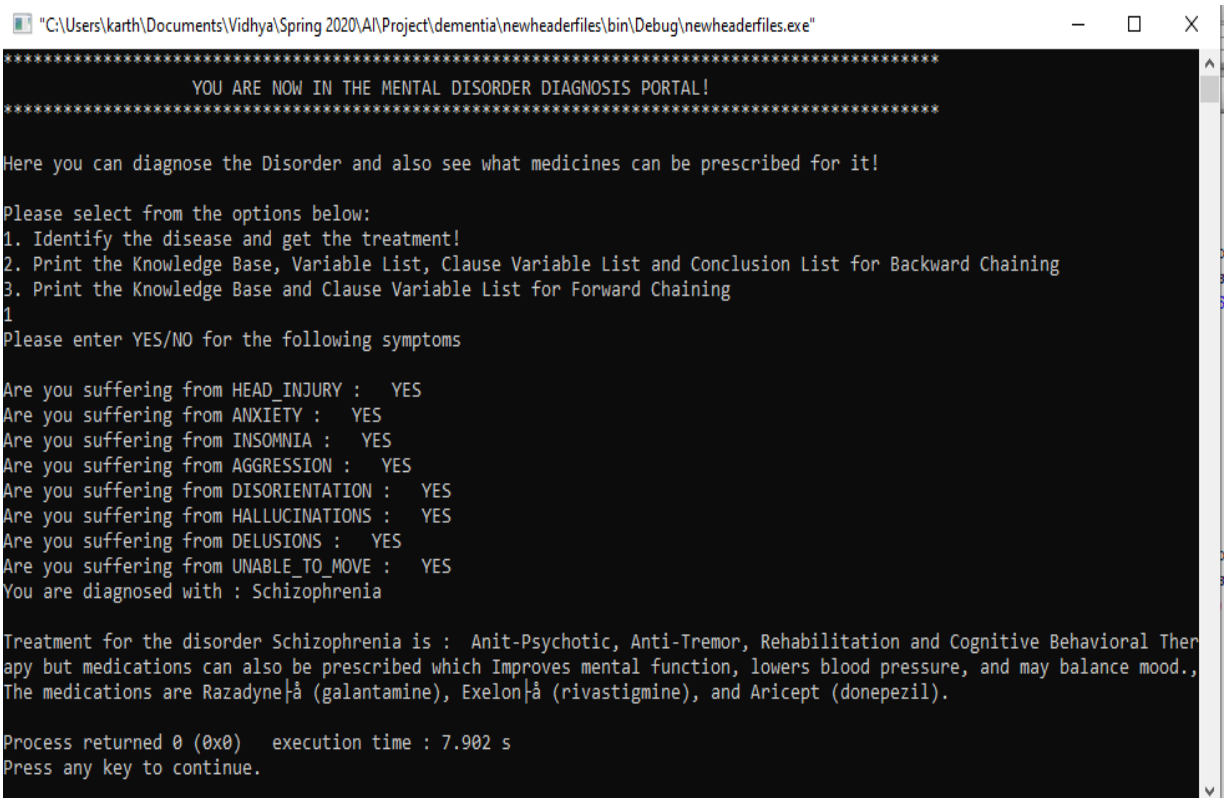
RUN 3

```
*********************************************************************************************
                    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 :   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.
```

RUN 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 :   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 :   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 :   N
WRONG ENTRY: Please enter YES or NO (uppercase)

Are you suffering from PARESTHESIAS :   NO
Are you suffering from PALPITATIONS :   NO
Are you suffering from URGE_TO_STEAL :   YES
Are you suffering from RELIEF_STEALING :   YES
You are diagnosed with : Kleptomania

Treatment for the disorder Kleptomania is : Medications like SSRI,Anti-depressants like Sertraline,Citalopram,Fluoxetine
, Paroxetine, Diazepam, Buspirone, Alprazolam,Lorazepam,Clonazepam will help. Behavioral Management is needed

Process returned 0 (0x0)   execution time : 22.288 s
Press any key to continue.
```

RUN 5


"C:\Users\karth\Documents\Vidhya\Spring 2020\AI\Project\dementia\newheaderfiles\bin\Debug\newheaderfiles.exe"
****
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 : 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
Treatment for the disorder Schizophrenia is : Anit-Psychotic, Anti-Tremor, Rehabilitation and Cognitive Behavioral Ther
apy but medications can also be prescribed which Improves mental function, lowers blood pressure, and may balance mood.,
The medications are Razadyne┼å (galantamine), Exelon┼å (rivastigmine), and Aricept (donepezil).
Process returned 0 (0x0) execution time : 7.902 s
Press any key to continue.

9 ANALYSIS OF THE PROGRAM AND RESULTS

9.1 PROBLEM WITH EXISTING CODE

We started our expert system by analyzing the existing code. There were a few drawbacks with the existing code. The drawbacks are listed as below:

- The existing code for Forward Chaining and Backward Chaining is written in C language which is not an Object-Oriented language. Due to this, we cannot reuse the code and the code is not flexible.
- The readability of the code is very poor due to which it is very hard to understand and modify/use the code for our requirements.
- There is no proper implementation of the different units of the inference engine and hence using this code for our purpose is difficult.
- The presence of GO-TO statements affects the performance of the system. Usage of these are not a good programming practice.
- Global variables usage makes the code less efficient as these can be updated in any part of the code and debugging as to which change affected the system adversely gets very difficult.
- Clause Variable List and Variable Lists are stored in the program which increases the lines of code and makes the code very lengthy.

9.2 PROPOSED SOLUTION

Approaching the Problem:

- We started off with research on the Web about different mental disorders and their symptoms.
- Segregated the diseases based on common symptoms and made sure each disease had a unique symptom so that the diagnosis would be easy.
- Did extensive research on the treatments that can be recommended for each disorder.

- Developed the Decision Tree based on the above collected data for both forward chaining and backward chaining.
- From the decision tree, the IF-THEN rules are formulated.

Result:

- The expert system is then designed and coded in C++ on the contrary to the existing code in C. This makes way for **reusing** the code as and when necessary. The code is now also **flexible**. The individual components can be **reused, updated and developed independently**.
- As we have made separate functions for each functionality and have included comments in the code wherever necessary, the code is now **readable and easy to understand**.
- Usage of stacks and queues, classes and objects and Maps, a few very important STL Library functions thereby **improving the efficiency**.
- No GO-To statements and global variables used.
- Text files are created for Variable List, Clause Variable List and Conclusion List instead of adding those to the code. Reading these values from the files into the program makes it **easy to maintain** and update. Any new additions to the list can be made to the files. If we update the respective sizes in the code, the code will work perfectly. No other changes are needed in the code while we use this approach. This makes the code more **scalable**.

10 CONCLUSION

Designing an Expert system for diagnosis and treatment of mental disorders was a learning experience. Over the course of completion of the project, I now have better understanding of Forward Chaining and Backward Chaining. Conversion of the understanding of concepts to a working project has been a path to learn the practical applications of these inference engines. I was able to use STL Library functions and learn about how it improved the program efficiency. Was able to write code that was more readable, flexible,

maintainable and got a hold of using stacks, queues, structs, classes and objects better. Besides, this project was a platform to learn in a better way to convert a problem in the form of a diagrammatic representation, the decision tree and convert these decisions to IF-THEN rules. Also made me aware of some mental disorders, how they impact day to day lives of humans and how to take care of one's mental health through the extensive research on web.

11 REFERENCES

- https://blog.doctorondemand.com/why-its-important-to-care-for-your-mental-health-34c8670b889
- Russell, Stuart, and Norvig, Peter. Artificial Intelligence, A Modern Approach. 3rd ed. Upper Saddle River, New Jersey, 2010
- https://en.wikipedia.org/wiki/Backward_chaining
- https://whatis.techtarget.com/definition/forward-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