
# Cancer Diagnostic Software

## AN INTELLIGENT COMPUTER EXPERT SYSTEM FOR DIAGNOSING CANCER

Madhusudhan Ramakrishnaiah

(A04735682)

## TABLE OF CONTENTS

# CHAPTER 1

# INTRODUCTION

## 1.1 Problem Statement

There are many types of Cancers: Blood, Thyroid, Kidney, Melanoma, Liver etc.

In general, whichever part of our body has cells then there is a possibility of having cancers in that part of our body. This a growing condition that needs to be diagnosed as early as possible to prevent the cancer from growing as if it reaches a terminal stage then there will be no stopping it.

Cancer is the uncontrolled growth of abnormal cells in the body. Cancer develops when the body's normal control mechanism stops working. Old cells do not die and instead grow out of control, forming new, abnormal cells. These extra cells may form a mass of tissue, called a tumor. Some cancers, such as leukemia, do not form tumors.

Doctors while diagnosing make mistakes by dismissing patient's symptoms as considering them as common fever, cough, or weight loss due to stress. but sometimes this is not the case. This might be a cancer and because of this mistake the patient's life will be lost. This has happened and will continue to happen since there is no way of stopping human errors completely.

## 1.2 Solution

For the above found problem we propose a solution to this by creating an intelligent computer expert system to identify the type of the cancer and possible treatments for removing the mutated cells and tumors if found and preventing the cancer to grow.

We will be using Backward Chaining Methodology to identify the type of the Cancer and Forward Chaining Methodology to recommend possible treatments. It means that you will be using one set of rules for backward chaining and the other set for the forward chaining.

We started the project by researching the information and developing two decision trees for this application. Later we transformed the decision trees into two sets of rules; one set for Backward Chaining and the second set for Forward Chaining.

The decision trees are big enough to generate a minimum of 25 rules in total. The rules contain variables.

We developed a user-friendly interface which receives input queries in restricted English format, uses string keyword matching, and responds to user.

The inference engine programs are used which were provided to us on TRACS but they were intentionally made inefficient and erroneous. So, we used the forward chaining program from the TRACS and modified it, but for backward chaining we felt that it would be better for us to

write the code from scratch, which we did do it. We made use of the methodologies taught in the class and used those in these programs.

We made use of Code Blocks IDE for writing and compiling the C++ code.

# CHAPTER 2

# CONTRIBUTIONS

## 2.1 Team Members

1. Madhusudhan Ramakrishnaiah

- Knowledge Gathering on types of cancers and treatments.
- Backward Chaining decision tree and rules.
- Backward Chaining conclusion list, variable list and clause variable list.
- Forward Chaining decision tree and rules.
- Forward Chaining variable list and clause variable list.
- Analysis of the give code, Converting the given code to C++.

2. Harsha Vardan Raaj Baskaran

- Knowledge Gathering on types of cancers and treatments.
- Backward Chaining decision tree and rules.
- Backward Chaining conclusion list, variable list and clause variable list.
- Forward Chaining decision tree and rules.
- Forward Chaining variable list and clause variable list.
- Analysis of the give code, Converting the given code to C++.

3. Santosh Jeebula

- Knowledge Gathering on types of cancers and treatments.
- Backward Chaining decision tree and rules.
- Backward Chaining conclusion list, variable list and clause variable list.
- Forward Chaining decision tree and rules.
- Forward Chaining variable list and clause variable list.
- Analysis of the give code, Converting the given code to C++.

# CHAPTER 3

# ANALYSIS

## 3.1 Goal

The main goal of this project was to develop an Artificial Intelligence Expert system tool which will be useful in identifying the type of cancer, if the user has and based on that diagnosis present the user with possible treatments.

## 3.2 Problem with the existing code

We initially collected all the information related to the different types of cancers that we were interested in; we collected the symptoms and the treatments for the same cancers. Then we designed 2 decision trees for backward and forward chaining.

We began with the code that was provided on TRACS , which was in c language. After analyzing the code, we concluded that is better if we use the c code for only forward chaining and not for backward chaining. Since in backward chaining there was a need to use dynamic arrays and structure arrays and many other concepts like overloading functions etc. was needed for us to implement the backward chaining so we wrote it from scratch and designed it using the concepts of how it was thought in our class and also took the reference from the ppt that was available in TRACS with the position, grade example.

## 3.3 Proposed Solution

The code that is used for implementing our backward and forward chaining will have the following advantages.

- We have used dynamic arrays to accommodate large number of data as it grows.
- Used structure arrays for implementing the main data structures (conclusion list, variable list, stack) in our backward chaining algorithm and, we have used in forward chaining.
- There is modularity in the code since we have used and defined functions only to perform specific tasks.
- The code is much more readable and easy to understand.
- The code is also reusable.

# CHAPTER 4

# FORWARD CHAINING

## 4.1 Definition

Forward chaining is one of the two main methods of reasoning when using inference rules (in artificial intelligence) and can be described logically as repeated application of *modus ponens*.

Forward chaining is a popular implementation strategy for expert systems, business and production rule systems. The opposite of forward chaining is backward chaining. Forward chaining starts with the available data and uses inference rules to extract more data (from an end user for example) until a goal is reached. An inference engine using forward chaining searches the inference rules until it finds one where the antecedent (**If** clause) is known to be true. When found it can conclude, or infer, the consequent (**Then** clause), resulting in the addition of new information to its data.

Inference engines will iterate through this process until a goal is reached.

One of the advantages of forward-chaining over backward-chaining is that the reception of new data can trigger new inferences, which makes the engine better suited to dynamic situations in which conditions are likely to change.

## STEPS FOR FORWARD CHAIN:

1. The condition variable is identified from the backward chaining algorithm.

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.

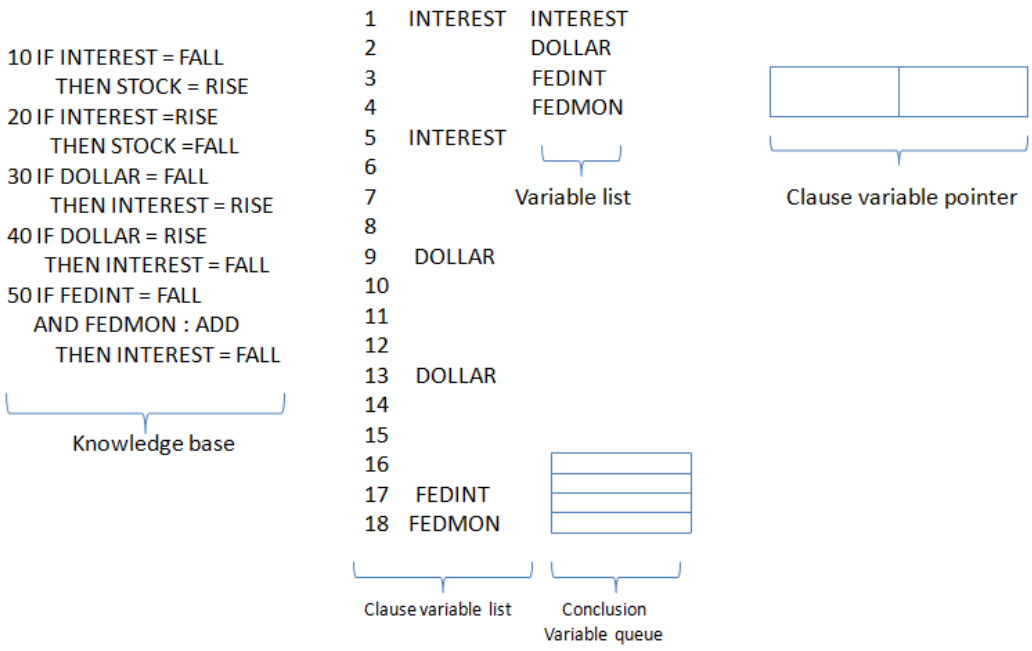
4. Each variable in the IF clause of the rule that is not already instantiated is now instantiated. The variables are in the clause variable list. If all the clauses are true, the THEN part is invoked.

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.

The below figure contains an example that gives an idea about how and what are the data structures used in forward chaining algorithm.




### 4.2 Working of a forward chaining system

Working of a typical forward chaining system is as follows:

1. The system is presented with one or more conditions.

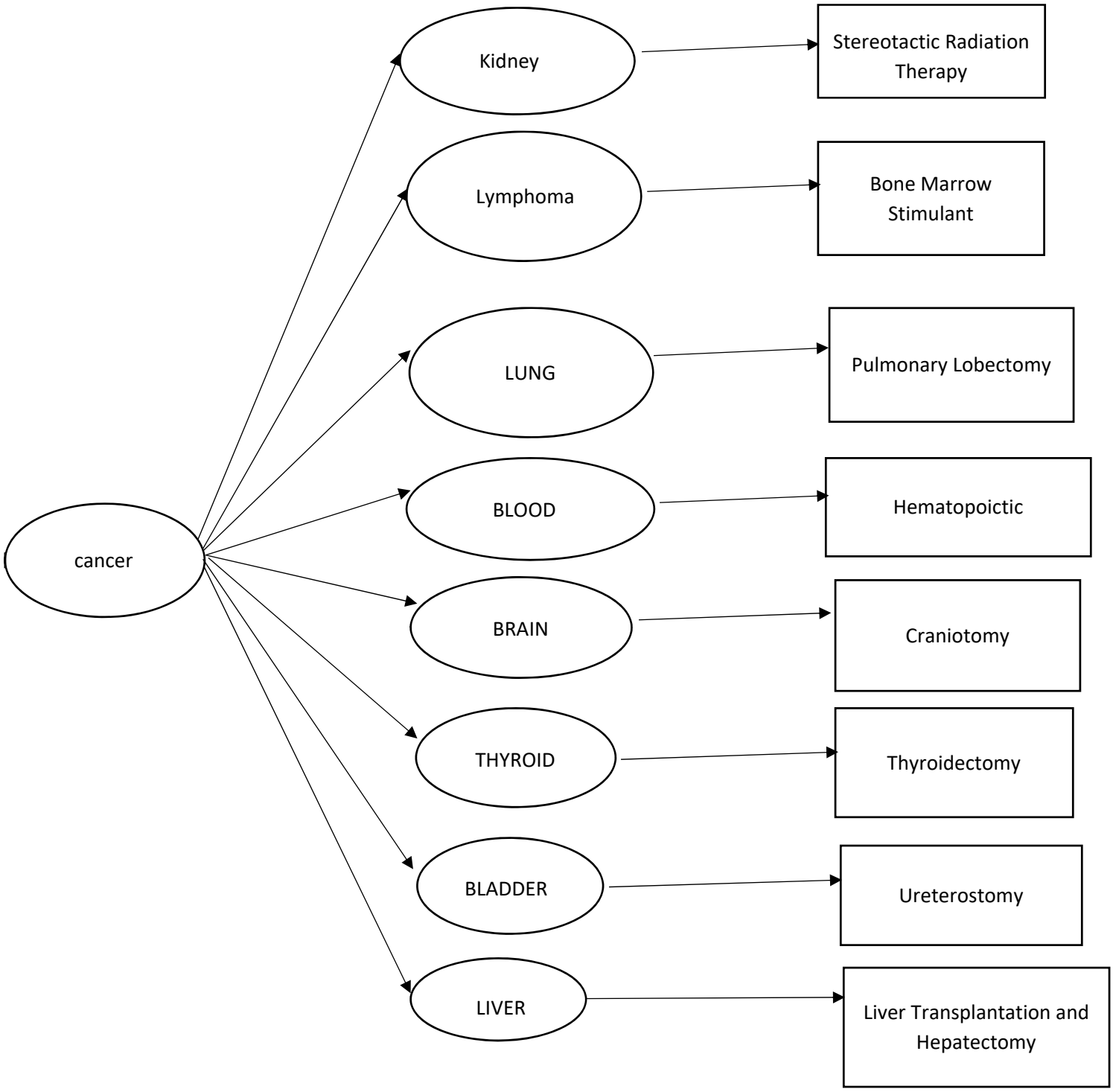
2. For each condition the system searches the rules in the knowledge base for those rules that correspond to the condition in the IF part.

3. Each rule can in turn generate new conditions from the conclusions of the invoked Then part. These new conditions are added to the ones already existing.

4. Any condition that have been added to the system are processed. If there are any such conditions, the system goes back to step 2 and searches the rules in the knowledge base again. If there are no more new conditions, the session ends.

### 4.3 Decision Tree

The decision tree is named so because it branches like a tree. The flat oval shapes are the conditions and the rectangular boxes are the conclusions.



**4.4 Rules**

We have 8 rules in the forward chaining each one for a disorder.

10.

If (cancer == "KIDNEY CANCER")

then Treatment = Stereotactic Radiation Therapy

20.

If (cancer == "LYMPHOMA CANCER")

then Treatment = Bone Marrow Stimulant

30.

If (cancer == "LUNG CANCER")

then Treatment = Pulmonary Lobectomy

40.

If (cancer == "BLOOD CANCER")

then Treatment = Hematopoictic

50.

If (cancer == "BRAIN CANCER")

then Treatment = Craniotomy

60.

If (cancer == "THYROID CANCER")

then Treatment = Thyroidectomy

70.

If (cancer == "BLADDER CANCER”)

then Treatment = Ureterostomy

80.

If (cancer == "LIVER CANCER")

then Treatment = Liver Transplantation and Hepatectomy

### 4.5 Data Structures

The data structures used for our project are as follows:

1. Clause Variable List:

   This list will let us know which variables in our problem are associated with the IF parts of specific IF-THEN statements.

| | |
|---|---|
| 1 | Cancer |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | Cancer |
| 7 | |
| 8 | |
| 9 | |
| 10 | |
| 11 | Cancer |
| 12 | |
| 13 | |
| 14 | |
| 15 | |
| 16 | Cancer |
| 17 | |
| 18 | |
| 19 | |

| | |
|---|---|
| 20 | |
| 21 | Cancer |
| 22 | |
| 23 | |
| 24 | |
| 25 | |
| 26 | Cancer |
| 27 | |
| 28 | |
| 29 | |
| 30 | |
| 31 | Cancer |
| 32 | |
| 33 | |
| 34 | |
| 35 | |
| 36 | Cancer |
| 37 | |
| 38 | |
| 39 | |
| 40 | |
| 41 | |

2. Conclusion List

| | |
|---|---|
| 1 | Stereotactic Radiation Theraphy |
| 2 | Bone Marrow Stimulant |
| 3 | Pulmonary Lobectomy |
| 4 | Hematopoictic |
| 5 | Craniotomy |
| 6 | Thyroidectomy |
| 7 | Ureterostomy |
| 8 | Liver Transplantation and Hepatectomy |

3. Clause variable pointer

   This pointer keeps track of the clause within the rule we are examining and is made up of the rule number and the clause number.

4. Variable list

   We make use of this data structure because it tells us if a variable has been instantiated or not. A variable list has two fields. The first field indicates whether the variable is instantiated or not (I or NI) and the second field indicated the value for the variable given by the user.

# CHAPTER 5

# BACKWARD CHAINING

## 5.1 Definition

Backward Chaining is an inference method that can be described as working backward from the conclusions.

## 5.2 Working of a backward chaining system

1. Backward chaining starts with a list of goals and works backwards from the consequent to the antecedent to see if there is data available that will support any of these consequents.

2. An inference engine using backward chaining would search the inference rules until it finds one which has a consequent (Then clause) that matches a desired goal.

3. If the antecedent (If clause) of the rule is not known to be true, then it is added to the list of goals

Algorithm for our project is as follows:

1. Get the conclusion from the user

2. search the conclusion stack for it

    1. if it is not present → Fail

    2. if it is present → add it to the conclusion stack and add set clause number as 1 in the conclusion stack and get the rule number

3. get the clause variables for the rule using the clause variable list.

4. For each variable, search the variable list if it is initialized

    1. If not → search the conclusion stack if it is present

        1. if it is not present → get the input from the user and initialize it in the variable list, increment the clause number in the conclusion stack.

        2. if it is present → push it to the conclusion stack and get the rule number, go to step 3

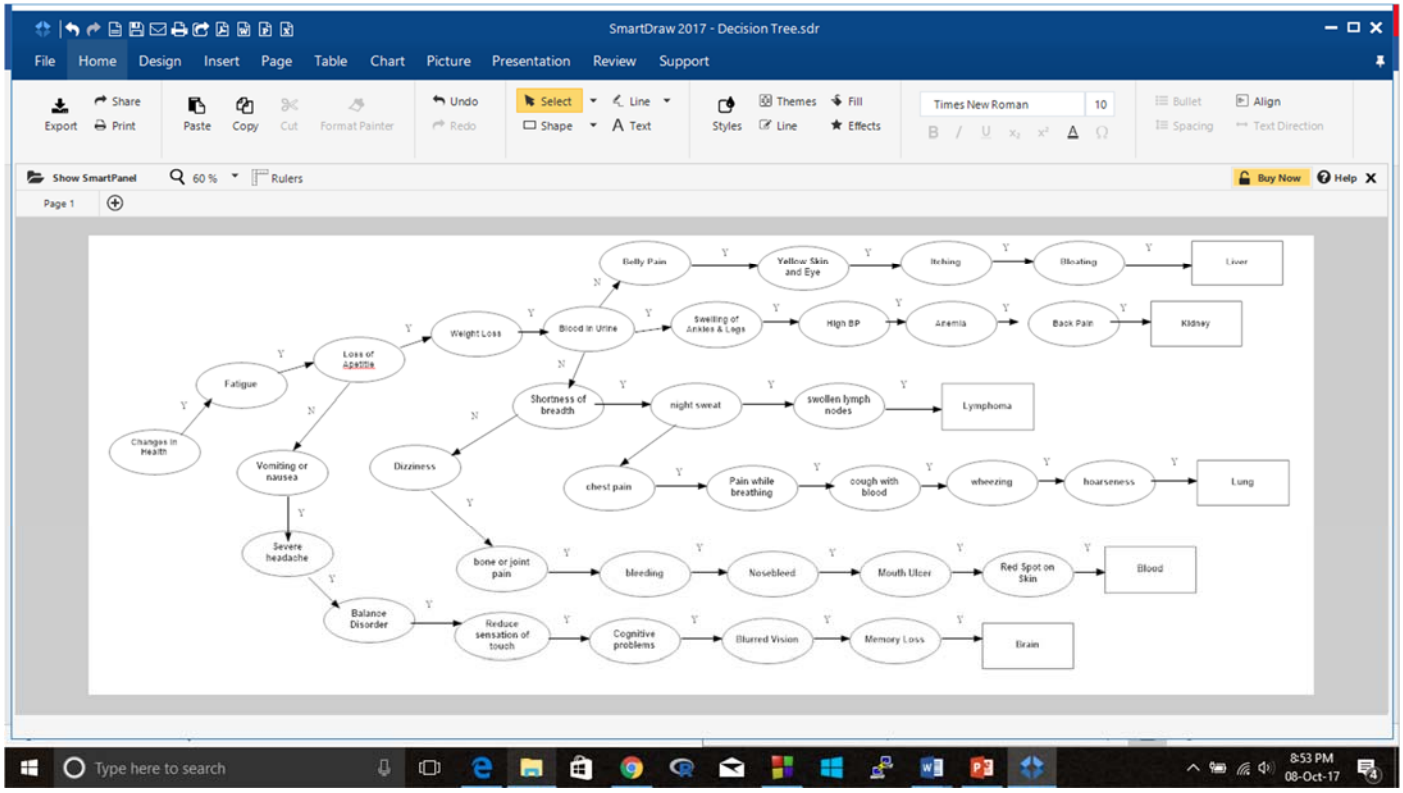
    2. If yes, increment the clause number in the conclusion stack and continue the same for next variables

5. when all the variables have been initialized

6. do this until the conclusion stack is empty

1. pop the conclusion stack
2. execute the rule
    1. if the rule cannot be executed from the given set of clause variables, search the conclusion stack for the next instance of this conclusion
        1. if present → get the rule number, goto step 3
        2. if not present → fail

## 5.3 Decision Tree



**5.4 Rules**

For our project we have considered enough cancer types to build 25 rules into our knowledge base.

10.

If (Change in Health == yes and Fatigue == yes)

Then Loss of Appetite = yes

20.

If (Loss of Appetite == yes and Weight Loss== yes)

Then Blood in Urine = yes

30.

If (Blood in Urine == yes and Swelling of Ankles and Legs== yes)

Then High BP = yes

40.

If (High BP == yes and Anemia == yes and Back Pain == yes)

Then Cancer = Kidney Cancer

50.

If (Fatigue == yes and Weight Loss == yes)

Then Unable to Eat = yes

60.

If (Loss of Appetite == yes and Shortness of Breath == yes)

Then Blood in Urine = yes

70.

If (Blood in Urine == yes and Dizziness == yes)

Then Bone or Joint Pain = yes

80.

If (Bone or Joint Pain == yes and Bleeding == yes)

Then Nosebleed= yes

90.

If (Nosebleed == yes and Mouth Ulcer == yes and Red Spot == yes)

Then Cancer = Blood Cancer

100.

If (Fatigue == yes and Loss of Appetite == yes and Vomiting == yes)

Then Severe Headache = yes

110.

If (Severe Headache == yes and Balance Disorder == yes and Reduced Sensation or Touch == yes)

Then Cognitive Problem = yes

120.

If (Cognitive Problem == yes and Blurred Vision == yes and Memory Loss== yes)

Then Cancer = Brain Cancer

130.

If (Fatigue == yes and Difficult to Swallow == yes)

Then Pain in Throat = yes

140.

If (Pain in Throat == yes and Enlarged Lymph Nodes == yes)

Then Hoarseness = yes

150.

If (Hoarseness == yes and Persisting Cough== yes)

Then Cancer = Thyroid Cancer

160.

If (Fatigue == yes and Loss Of Appetite == yes and Weight Loss == yes)

Then Night Sweat = yes

170.

If (Night Sweat == yes and Shortness of Breath== yes)

Then Losing Weight = yes

180.

If (Losing Weight == yes and Swollen Lymph Nodes == yes)

Then Cancer = Lymphoma Cancer

190.

If (Fatigues == yes and Loss of Appetite == yes and Weight Loss == Yes)

Then Chest Pain = yes

200.

If (Chest Pain == yes and Pain while Breathing == yes)

Then Cough with Blood = yes

210.

If (Cough with Blood == yes and Rough Sensation in Voice == yes)

Then Wheezing = yes

220.

If (Wheezing == yes)

Then Cancer = Lung Cancer

230.

If (Weight Loss == yes and Belly Pain == yes and Vomiting == yes)

Then Yellow Skin or Eye = yes

240.

If (Yellow Skin or Eye == yes and Loss of Appetite == yes)

Then Itching = yes

250.

If (Itching == yes and Bloating == yes)

Then Cancer = Liver Cancer

### 5.5 Data Structures

1. Conclusion List

This is one of the data structure used for implementing backward chaining algorithm. It is made as an array of structure.

It contains 2 fields one being the conclusion variable and the other its corresponding rule number.

| | |
|---|---|
| 10 | LOSS OF APETITE |
| 20 | BLOOD IN URINE |
| 30 | HIGH BP |
| 40 | CANCER |
| 50 | UNABLE TO EAT |
| 60 | BLOOD WHILE URINATING |
| 70 | BONE OR JOINT PAIN |
| 80 | NOSE BLEED |
| 90 | CANCER |
| 100 | SEVERE HEADACHE |
| 110 | COGNITIVE PROBLEM |
| 120 | CANCER |
| 130 | PAIN IN THROAT |
| 140 | HOARSENESS |
| 150 | CANCER |
| 160 | NIGHT SWEAT |
| 170 | LOOSING WEIGHT |
| 180 | CANCER |
| 190 | CHEST PAIN |
| 200 | COUGH WITH BLOOD |
| 210 | WHEEZING |
| 220 | CANCER |
| 230 | YELLOW SKIN AND EYE |
| 240 | ITCHING |
| 250 | CANCER |

2. Variable List

This is an array of structures. It has 3 fields in each of the array elements. Status, value and content. Where status field determines whether the variable has been instantiated or not
Value filed can have 2 possible values that is yes/no depending on what the user enters.
Content field contains the variable string name i.e. each variable contained in the IF part of the knowledge base rules.

| | | |
|---|---|---|
| NI | | CHANGES IN HEALTH |
| NI | | FATIGUE |
| NI | | WEIGHT LOSS |
| NI | | SWELLING OF ANKLES AND LEGS |
| NI | | Anemia |
| NI | | BACK PAIN |
| NI | | SHORTNESS OF BREATH |
| NI | | DIZINESS |
| NI | | BLEEDING |
| NI | | |
| NI | | |
| NI | | |
| NI | | |
| NI | | |
| NI | | |
| NI | | MOUTH ULCER |
| NI | | RED SPOT |
| NI | | |
| NI | | |
| NI | | VOMITING |
| NI | | |
| NI | | BALANCE DISORDER |
| NI | | REDUCED SENSATION OR TOUCH |
| NI | | |
| NI | | BLURRED VISION |
| NI | | MEMORY LOSS |
| NI | | DIFFICULT TO SWALLOW |
| NI | | ENLARGED LYMPH NODES |
| NI | | PERSISTING COUGH |
| NI | | |
| NI | | |

| | | |
|---|---|---|
| NI | | SWOLLEN LYMPH NODES |
| NI | | |
| NI | | |
| NI | | PAIN WHILE BREATHING |
| NI | | |
| NI | | |
| NI | | |
| NI | | ROUGH SENSATION IN VOICE |
| NI | | BELLY PAIN |
| NI | | BLOATING |
| NI | | |
| NI | | |

3. Clause Variable list

All the variables that are present in the knowledge base is added into this data structure.
This is implemented as string array with size 4 for each if-then clause that appears in our knowledge base.

| | |
|---|---|
| 1 | CHANGES IN HEALTH |
| 2 | FATIGUE |
| 3 | |
| 4 | |
| 5 | LOSS OF APETITE |
| 6 | WEIGHT LOSS |
| 7 | |
| 8 | |
| 9 | BLOOD IN URINE |
| 10 | SWELLING OF ANKLES AND LEGS |
| 11 | |
| 12 | |
| 13 | HIGH BP |
| 14 | ANEMIA |
| 15 | BACK PAIN |
| 16 | |
| 17 | FATIGUE |
| 18 | WEIGHT LOSS |
| 19 | |
| 20 | |

| | |
|---|---|
| 21 | UNABLE TO EAT |
| 22 | SHORTNESS OF BREATH |
| 23 | |
| 24 | |
| 25 | BLOOD WHILE URINATING |
| 26 | DIZINESS |
| 27 | |
| 28 | |
| 29 | BONE OR JOINT PAIN |
| 30 | BLEEDING |
| 31 | |
| 32 | |
| 33 | NOSE BLEED |
| 34 | MOUTH ULCER |
| 35 | RED SPOT |
| 36 | |
| 37 | FATIGUE |
| 38 | LOSS OF APETITE |
| 39 | VOMITING |
| 40 | |
| 41 | SEVERE HEADACHE |
| 42 | BALANCE DISORDER |
| 43 | REDUCED SENSATION OR TOUCH |
| 44 | |
| 45 | COGNITIVE PROBLEM |
| 46 | BLURRED VISION |
| 47 | MEMORY LOSS |
| 48 | |
| 49 | FATIGUE |
| 50 | DIFFICULT TO SWALLOW |
| 51 | |
| 52 | |
| 53 | PAIN IN THROAT |
| 54 | ENLARGED LYMPH NODES |
| 55 | |
| 56 | |
| 57 | HOARSENESS |
| 58 | PERSISTING COUGH |
| 59 | |
| 60 | |
| 61 | FATIGUE |
| 62 | LOSS OF APETITE |

| | |
|---|---|
| 63 | WEIGHT LOSS |
| 64 | |
| 65 | NIGHT SWEAT |
| 66 | SHORTNESS OF BREATH |
| 67 | |
| 68 | |
| 69 | LOOSING WEIGHT |
| 70 | SWOLLEN LYMPH NODES |
| 71 | |
| 72 | |
| 73 | FATIGUE |
| 74 | LOSS OF APETITE |
| 75 | WEIGHT LOSS |
| 76 | |
| 77 | CHEST PAIN |
| 78 | PAIN WHILE BREATHING |
| 79 | |
| 80 | |
| 81 | COUGH WITH BLOOD |
| 82 | ROUGH SENSATION IN VOICE |
| 83 | |
| 84 | |
| 85 | WHEEZING |
| 86 | |
| 87 | |
| 88 | |
| 89 | WEIGHT LOSS |
| 90 | BELLY PAIN |
| 91 | VOMITING |
| 92 | |
| 93 | YELLOW SKIN AND EYE |
| 94 | FATIGUE |
| 95 | LOSS OF APETITE |
| 96 | |
| 97 | ITCHING |
| 98 | BLOATING |
| 99 | |
| 100 | |

4. Conclusion Stack

The conclusion stack is the central structure. It ties together all of 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.

# CHAPTER 6

# PROGRAM IMPLEMENTATION

## 6.1 Backward Chaining Code and Forward Chaining Code

```
#include <iostream>
using namespace std;
#include <stack>
#include <string>
#include <queue>
#include <cstdlib>
#include <stdio.h>



/*VARIABLE LIST --------- 1 data structure
status -> I/NI; value -> Yes/No; content -> names in variable list i.e that appears in IF clauses
*/
struct variable_list {

        string status;
        string value;
        string content;
};

//variable list size
int vl_size = 43;
//declares a dynamic variable list array of size vl_size
variable_list *vl = new variable_list[vl_size];
```

```
/*CONCLUSION STACK --------- 2 data structure
rule_no -> contains the rule number, takes it from conclusion list rule number; clause_no -> calculates the clause number from the rule number
*/
struct conclusion_stack {

        int rule_no;
        int clause_no;
};



/*CONCLUSION LIST --------- 3 data structure
rule_num -> contains the rule number its already stored from the main manually;
conclusion -> contains the list of conclusion variables i.e that appears in THEN clauses;
*/
struct conclusion_list {
        int rule_num;
        string conclusion;
};

//conclusion list size
const int cl_size = 25;
conclusion_list cl[25];
//CLAUSE-VARIABLE LIST SIZE
int cvl_size = 100;

/*for check_conclusion_list function;
keeps track of the array location at present for the conclusion list
```

```
*/
int counter = 0;

int counter2 = 0;

/*for checking if variable not in variable list
initially set to false , assigns true when not in variable list
used for calling function check_clause_variable_list2 if main_check is true,
else if false calls function check_clause_variable_list
*/
bool main_check = false;

//declares clause variable list as string array --------- 4  data structure
string clause_variable_list[100];

//stack for maintaining rule number and clause number --------- 5  data structure
stack <conclusion_stack> cs_stack;

//queue for maintaining which variables is being processed from the clause variable list
queue <string> que;


/*used in function check_conclusion_stack
clause_no_loc -> has the starting array location of clause variable list which is calculated from rule number
end_clause_no_loc -> has the ending array location of clause variable list which is clause_no_loc + Z
where Z is the no.of locations alloted for each rule
*/
```

```
int clause_no_loc;
int end_clause_no_loc;
int rule = -1;

/*declares size for storing variable list's variables value i.e Yes/No
and a string array for it
size must be set to no.of variables in variable list + A
where A is the no.of variables not in variable list while checking every variable in clause variable list but in conclusion list */
const int size = 5;
string x[size] = { " "," "," "," "," " };
bool change = false;


//hold elements from old queue while modifying queue each time in check clause variable list
string *temp = new string[100];


//**********************************************************************for backward chaining
void check_kb(int rule_no, string x[], string &value, string user_ip);
void check_conclusion_list(conclusion_list cl[], string user_ip);
void check_conclusion_list2(conclusion_list cl[], string x, int index);
void check_conclusion_stack(stack<conclusion_stack> &stak, string user);
bool check_instantiation(variable_list vl[], int loc);
void instantiate(variable_list vl[], int loc, string cvl[]);
void check_variable_list(variable_list vl[], string x, string cvl[]);
void check_clause_variable_list(string cvl[], variable_list vl[], string user_ip);
```

```
//**********************************************************************for backward chaining



//#############################################################################################FORWARD CHAINING
void search(void);
void check_instantiation(void);
void instantiate(void);
void display();

int flag;
string conclusionlist[9];
string variablelist[9], /* variable list*/ clausevarlist[41]; /* clause var list */
string conditionvariable, variablename /* condition variable */; /*variable */
string cancer, treatment;
string ctreatment[] = { "Surgery","Radiation Therapy","Chemotherapy","Immunotherapy","Targeted Therapy","Hormone Therapy","Stem Cell Transplant","Precision Medicine" };
int instlt[37];        /* instantiated list*/
int f, i, j, k, s, fpointer   /* front pointer */;
int  bponiter  /* back pointer */, gr /* grade */, rulenumber;
int clausenumber;  /* clause number */

string bc;
//#############################################################################################FORWARD CHAINING



void check_kb(int rule_no, string x[], string &value, string user_ip) {
```

```
switch (rule_no) {

case 10:

        if (vl[0].value == "Yes" && vl[1].value == "Yes") {

                vl[10].content = "LOSS OF APETITE";
                vl[10].value = "Yes";
                vl[10].status = "I";
                cs_stack.pop();
                change = true;
        }

        else {
                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();

                for (int i = 0; i<100; i++) {

                        temp[i] = " ";
                }


                counter = counter + 1;
                check_conclusion_list(cl, user_ip);
```

```
        }

        break;


case 20:

        if (vl[10].value == "Yes" && vl[2].value == "Yes") {

                vl[11].content = "BLOOD IN URINE";
                vl[11].value = "Yes";
                vl[11].status = "I";


                cs_stack.pop();

                change = true;

        }


        else {

                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();
```

```
            for (int i = 0; i<100; i++) {

                temp[i] = " ";
            }



            counter = counter + 1;
            check_conclusion_list(cl, user_ip);
        }
        break;


case 30:

        if (vl[11].value == "Yes" && vl[3].value == "Yes") {

            vl[12].content = "HIGH BP";
            vl[12].value = "Yes";
            vl[12].status = "I";



            cs_stack.pop();
            change = true;

        }
```

```
        else {


                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();

                for (int i = 0; i<100; i++) {

                        temp[i] = " ";
                }

                counter = counter + 1;
                check_conclusion_list(cl, user_ip);
        }

        break;

case 40:

        if (vl[12].value == "Yes" && vl[4].value == "Yes" && vl[5].value == "Yes") {
                bc = "KIDNEY CANCER";

                cout << endl;
                cout << "Diagnosis: " << bc << endl;
                return;
```

```
        }
        else {

                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();

                for (int i = 0; i<100; i++) {

                        temp[i] = " ";
                }


                counter = counter + 1;
                check_conclusion_list(cl, user_ip);
        }
        break;

case 50:

        if (vl[1].value == "Yes" && vl[2].value == "Yes") {
                vl[18].content = "UNABLE TO EAT";
                vl[18].value = "Yes";
                vl[18].status = "I";
                cs_stack.pop();
                change = true;
```

```
        }

        else {
                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();

                for (int i = 0; i<100; i++) {

                        temp[i] = " ";
                }

                counter = counter + 1;
                check_conclusion_list(cl, user_ip);
        }
        break;

case 60:

        if (vl[18].value == "Yes" && vl[6].value == "Yes") {
                vl[17].content = "BLOOD WHILE URINATING";
                vl[17].status = "I";
                vl[17].value = "Yes";
                cs_stack.pop();
                change = true;
        }
```

```
else {
                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();

                for (int i = 0; i<100; i++) {

                        temp[i] = " ";
                }

                counter = counter + 1;
                check_conclusion_list(cl, user_ip);
        }
        break;

case 70:

        if (vl[17].value == "Yes" && vl[7].value == "Yes") {

                vl[13].content = "BONE OR JOINT PAIN";
                vl[13].status = "I";
                vl[13].value = "Yes";
                cs_stack.pop();
                change = true;
        }
```

```
        else {
                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();

                for (int i = 0; i<100; i++) {

                        temp[i] = " ";
                }

                counter = counter + 1;
                check_conclusion_list(cl, user_ip);
        }
        break;

case 80:

        if (vl[13].value == "Yes" && vl[8].value == "Yes") {

                vl[14].content = "NOSE BLEED";
                vl[14].status = "I";
                vl[14].value = "Yes";
                cs_stack.pop();
                change = true;
        }
```

```
        else {

                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();



                for (int i = 0; i<100; i++) {
                        temp[i] = " ";
                }

                counter = counter + 1;
                check_conclusion_list(cl, user_ip);
        }
        break;

case 90:

        if (vl[14].value == "Yes" && vl[15].value == "Yes" && vl[16].value == "Yes") {
                bc = "BLOOD CANCER";
                cout << endl;
                cout << "Diagnosis: " << bc;
                return;
        }
```

```
        else {

                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();

                for (int i = 0; i<100; i++) {
                        temp[i] = " ";
                }

                counter = counter + 1;
                check_conclusion_list(cl, user_ip);
        }
        break;

case 100:

        if (vl[1].value == "Yes" && vl[10].value == "Yes" &&  vl[19].value == "Yes") {
                vl[20].content = "SEVERE HEADACHE";
                vl[20].status = "I";
                vl[20].value = "Yes";
                cs_stack.pop();
                change = true;
        }

        else {
```

```
                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();

                for (int i = 0; i<100; i++) {
                        temp[i] = " ";
                }

                counter = counter + 1;
                check_conclusion_list(cl, user_ip);
        }
        break;

case 110:

        if (vl[20].value == "Yes" && vl[21].value == "Yes" &&  vl[22].value == "Yes") {
                vl[23].content = "COGNITIVE PROBLEM";
                vl[23].status = "I";
                vl[23].value = "Yes";
                cs_stack.pop();
                change = true;
        }
```

```
        else {

                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();

                for (int i = 0; i<100; i++) {
                        temp[i] = " ";
                }

                counter = counter + 1;
                check_conclusion_list(cl, user_ip);
        }
        break;


case 120:

        if (vl[23].value == "Yes" && vl[24].value == "Yes" &&  vl[25].value == "Yes") {
                bc = "BRAIN CANCER";
                cout << endl;
                cout << "Diagnosis: " << bc << endl;
                return;
        }
```

```
else {

                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();

                for (int i = 0; i<100; i++) {
                        temp[i] = " ";
                }

                counter = counter + 1;
                check_conclusion_list(cl, user_ip);
        }
        break;

case 130:

        if (vl[1].value == "No" && vl[26].value == "Yes") {
                vl[29].content = "PAIN IN THROAT";
                vl[29].status = "I";
                vl[29].value = "Yes";
                cs_stack.pop();
                change = true;
        }
```

```
else {

                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();

                for (int i = 0; i<100; i++) {
                        temp[i] = " ";
                }

                counter = counter + 1;
                check_conclusion_list(cl, user_ip);
        }
        break;

case 140:

        if (vl[29].value == "Yes" && vl[27].value == "Yes") {
                vl[30].content = "HOARSENESS";
                vl[30].status = "I";
                vl[30].value = "Yes";
                cs_stack.pop();
                change = true;
        }
```

```
else {

                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();

                for (int i = 0; i<100; i++) {
                        temp[i] = " ";
                }

                counter = counter + 1;
                check_conclusion_list(cl, user_ip);
        }
        break;

case 150:

        if (vl[30].value == "Yes" && vl[28].value == "Yes") {

                bc = "THYROID CANCER";
                cout << endl;
                cout << "Diagnosis: " << bc << endl;
                return;
        }
```

```
        else {

                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();

                for (int i = 0; i<100; i++) {
                        temp[i] = " ";
                }

                counter = counter + 1;
                check_conclusion_list(cl, user_ip);
        }
        break;

case 160:

        if (vl[1].value == "Yes" && vl[10].value == "Yes"  && vl[2].value == "Yes") {

                vl[32].content = "NIGHT SWEAT";
                vl[32].status = "I";
                vl[32].value = "Yes";
                cs_stack.pop();
                change = true;
        }
```

```
        else {

                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();

                for (int i = 0; i<100; i++) {
                        temp[i] = " ";
                }

                counter = counter + 1;
                check_conclusion_list(cl, user_ip);
        }
        break;

case 170:

        if (vl[32].value == "Yes" && vl[6].value == "Yes") {

                vl[33].content = "LOOSING WEIGHT";
                vl[33].status = "I";
                vl[33].value = "Yes";
                cs_stack.pop();
                change = true;
        }
```

```
        else {

                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();

                for (int i = 0; i<100; i++) {
                        temp[i] = " ";
                }

                counter = counter + 1;
                check_conclusion_list(cl, user_ip);
        }
        break;

case 180:

        if (vl[33].value == "Yes" && vl[31].value == "Yes") {

                bc = "LYMPHOMA CANCER";
                cout << endl;
                cout << "Diagnosis: " << bc << endl;
                return;
        }
```

```
        else {

                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();

                for (int i = 0; i<100; i++) {
                        temp[i] = " ";
                }

                counter = counter + 1;
                check_conclusion_list(cl, user_ip);
        }
        break;

case 190:

        if (vl[1].value == "Yes" && vl[10].value == "Yes" && vl[2].value == "Yes") {
                vl[35].content = "CHEST PAIN";
                vl[35].status = "I";
                vl[35].value = "Yes";
                cs_stack.pop();
                change = true;
        }

        else {
```

```
                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();

                for (int i = 0; i<100; i++) {
                        temp[i] = " ";
                }

                counter = counter + 1;
                check_conclusion_list(cl, user_ip);
        }
        break;

case 200:

        if (vl[35].value == "Yes" && vl[34].value == "Yes") {
                vl[36].value = "Yes";
                vl[36].content = "COUGH WITH BLOOD";
                vl[36].status = "I";
                vl[36].value = "Yes";
                cs_stack.pop();
                change = true;
        }
```

```
else {
                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();

                for (int i = 0; i<100; i++) {
                temp[i] = " ";
                }

                counter = counter + 1;
                check_conclusion_list(cl, user_ip);
        }
        break;

case 210:
        if (vl[36].value == "Yes" && vl[38].value == "Yes") {

                vl[37].content = "WHEEZING";
                vl[37].status = "I";
                vl[37].value = "Yes";
                cs_stack.pop();
                change = true;
        }


        else {
```

```
            while (!cs_stack.empty())
                cs_stack.pop();

            while (!que.empty())
                que.pop();

            for (int i = 0; i<100; i++) {
                temp[i] = " ";
            }

            counter = counter + 1;
            check_conclusion_list(cl, user_ip);
        }
        break;

case 220:

        if (vl[37].value == "Yes") {

            bc = "LUNG CANCER";
            cout << endl;
            cout << "Diagnosis: " << bc;
            return;
        }
        else {

            while (!cs_stack.empty())
```

```
            cs_stack.pop();

            while (!que.empty())
                que.pop();

            for (int i = 0; i<100; i++) {
                temp[i] = " ";
            }

            counter = counter + 1;
            check_conclusion_list(cl, user_ip);
        }
        break;


case 230:


        if (vl[2].value == "Yes" && vl[39].value == "Yes" && vl[19].value == "Yes") {

            vl[41].content = "YELLOW SKIN AND EYE";
            vl[41].status = "I";
            vl[41].value = "Yes";
            cs_stack.pop();
            change = true;
        }
```

```
        else {
                while (!cs_stack.empty())
                        cs_stack.pop();

                while (!que.empty())
                        que.pop();

                for (int i = 0; i<100; i++) {
                        temp[i] = " ";
                }

                bc = "Diagnosis: NO CANCER";

                cout << endl;
                cout << "Diagnosis: NO CANCER" << endl;
                exit(0);
        }
        break;



case 240:
        if (vl[41].value == "Yes" && vl[1].value == "Yes" && vl[10].value == "Yes") {

                vl[42].content = "ITCHING";
                vl[42].status = "I";
                vl[42].value = "Yes";
                cs_stack.pop();
```

```
            change = true;
        }

        else {

            while (!cs_stack.empty())
                cs_stack.pop();

            while (!que.empty())
                que.pop();

            for (int i = 0; i<100; i++) {
                temp[i] = " ";
            }

            bc = "Diagnosis: NO CANCER";

            cout << endl;
            cout << "Diagnosis: NO CANCER" << endl;
            exit(0);
        }
        break;

case 250:
        if (vl[42].value == "Yes" && vl[40].value == "Yes") {
            bc = "LIVER CANCER";
            cout << endl;
            cout << "Diagnosis: " << bc;
```

```
                return;

            }

            else {
                bc = "Diagnosis: NO CANCER";
                cout << endl;
                cout << "Diagnosis: NO CANCER" << endl;
                exit(0);
            }
            break;
        }
}


void check_conclusion_list(conclusion_list cl[], string user_ip) {
        for (; counter<25; counter++) {
            if (cl[counter].conclusion == "CANCER") {
                rule = cl[counter].rule_num;
                break;
            }
        }

        if (rule == -1) {
            cout << "condition variable that is entered was not found" << endl;

        }
```

```
else {

        check_conclusion_stack(cs_stack, user_ip);
    }

}

void check_conclusion_stack(stack<conclusion_stack> &stak, string user) {
    clause_no_loc = 4 * (rule / 10 - 1) + 1;
    end_clause_no_loc = clause_no_loc + 4;
    conclusion_stack var;
    var.rule_no = rule;
    var.clause_no = clause_no_loc;
    cs_stack.push(var);
    check_clause_variable_list(clause_variable_list, vl, user);
}


void check_clause_variable_list(string cvl[], variable_list vl[], string user_ip) {
    int c = 0;
    clause_no_loc = cs_stack.top().clause_no;
    end_clause_no_loc = clause_no_loc + 4;

    if (!main_check) {

        for (int i = clause_no_loc - 1; i < end_clause_no_loc - 1; i++) {

            if (cvl[i] != " ") {
```

```
                que.push(cvl[i]);
                        c++;
                }
        }
}

else {
        //makes the queue empty by collecting all values to temp array and then populates with new found variable then adds the values from temp array

        int j = 0;

        while (!que.empty()) {
                temp[j] = que.front();
                que.pop();
                j++;
        }

        for (int i = clause_no_loc - 1; i < end_clause_no_loc - 1; i++) {

                if (cvl[i] != " ") {
                        que.push(cvl[i]);

                }


        }
```

```
            for (int i = 0; i<j; i++) {
                  if (temp[i] != " ") {
                        que.push(temp[i]);

                  }
            }

      }

      int i = 0;
      while (que.size()) {

            string queue_value;
            queue_value = que.front();
            x[i] = queue_value;
            que.pop();
            check_variable_list(vl, queue_value, cvl);
      }
      //making sure queue is empty before proceeding
      if (que.size() != 0) {
            check_variable_list(vl, que.front(), cvl);
            que.pop();
      }
      string v;
      rule = (cs_stack.top()).rule_no;
      check_kb(rule, x, v, user_ip);

}
```

```
void check_variable_list(variable_list vl[], string x1, string cvl[]) {
	bool v, found = false;
	int i;

	for (i = 0; i<vl_size; i++) {
		if (vl[i].content == x1) {
			found = true;
			v = check_instantiation(vl, i);
			break;
		}
	}

	if (v) {
		//cout<<x1<<" already instantiated"<<endl;
	}

	if (found == true && !v) {
		instantiate(vl, i, cvl);
		//cout<<"instantiated "<<vl[i].content<<endl;
	}

	if (!found) {

		main_check = true;
		check_conclusion_list2(cl, x1, 0);
	}
}
```

```
bool check_instantiation(variable_list vl[], int loc) {
	if (vl[loc].status == "NI" && vl[loc].value == " ")
		return false;
	else
		return true;
}

void instantiate(variable_list vl[], int loc, string cvl[]) {

	vl[loc].status = "I";
	cout << endl;
	cout << "enter possible values for " << vl[loc].content << " " << endl;
	cout << "1.Yes" << endl << "2.No" << endl;
	int n;
	cin >> n;

	if (n == 1)
		vl[loc].value = "Yes";

	else
		vl[loc].value = "No";
}

//checks whether user_ip is presesnt and returns its rule number

void check_conclusion_list2(conclusion_list cl[], string x, int index) {

	for (index = 0; index<25; index++) {
```

```
            if (x == cl[index].conclusion) {

                rule = cl[index].rule_num;

                break;
            }
        }

        if (rule == -1) {
            cout << "conclusion not found";

        }
        else {
            check_conclusion_stack(cs_stack, x);

        }

}

//#####################################FORWARD CHAINING FUNCTIONS

//=============================================================================
======

/* Routine to instantiate a variable (v) if it irule_number't already.
The instantiate indication (instlt) is a 0 if not, a 1 if it is.
The variable list (variable_list) contains the variable (v) */
```

```
void check_instantiation()
{
        i = 1;

        /* find variable in the variable list */
        while ((variablename != variablelist[i]) && (i <= 9)) i = i + 1;

        /* check if already instantiated */
        if (instlt[i] != 1)
        {
                /* mark instantiated */
                instlt[i] = 1;
                /* the designer of this knowledge base places the input
                statements to instantiate the variables in this case
                statement */
                switch (i)
                {
                case 1:
                        cout << "\n**********Specific Treatment For The Above Diagnosis **********\n" << endl;
                        // cout << "\n Enter the Cancer Type to find Specific Treatment: ";
                        cancer = bc;

                        break;
                }
        }
}
```

```
//=================================================================
======

/* Search clause variable list for a varialbe (clause_var_list) equal to the
one in front of the conclusion queue (clause_numberdvar). Return the statement
number (rule_number). If there is no match, i.e., rule_number=0, the first statement
for the space is f. */
void search()
{
        flag = 0;
        rulenumber = f;

        while ((flag == 0) && (rulenumber <= 9))
        {
                clausenumber = 1;
                k = (rulenumber - 1) * 5 + clausenumber;
                while ((clausevarlist[k] != conclusionlist[fpointer]) && (clausenumber < 9))
                {
                        clausenumber = clausenumber + 1;
                        k = (rulenumber - 1) * 5 + clausenumber;
                }

                if (clausevarlist[k] == conclusionlist[fpointer]) flag = 1;
                if (flag == 0) rulenumber = rulenumber + 1;
        }
        if (flag == 0) rulenumber = 0;
}

//=================================================================
======
```

```
/* Routine to instantiate a varialbe (v) and then place it on the
back of the queu (clause_numberdvar[back_poniter]), if it is not already there. */
void instantiate()
{
        i = 1;
        /* find varialbe in the varialbe list (variable_list) */
        while ((variablename != variablelist[i]) && (i <= 9)) i = i + 1;

        /* instantiate it */
        instlt[i] = 1;
        i = 1;

        /* determine if (variable_name) is or already has been on the queue (clause_numberdvar)
*/
        while ((variablename != conclusionlist[i]) && (i <= 9)) i = i + 1;
        /* variable has not been on the queue. Store it in the back of the queue */
        if (variablename != conclusionlist[i])
        {
                conclusionlist[bponiter] = variablename;
                bponiter = bponiter + 1;
        }
}
/*Function to return full names for the abbreviations used */
void display()
{

        /*
        cout<<"\n**********LIST OF CANCER
TYPES**********\n"<<endl;
```

```
	cout<<"1. KIDNEY"<<endl;
	cout<<"2. LYMPHOMA"<<endl;
	cout<<"3. LUNG"<<endl;
	cout<<"4. BLOOD"<<endl;
	cout<<"5. BRIAN"<<endl;
	cout<<"6. THYROID"<<endl;
	cout<<"7. BLADDER"<<endl;
	cout<<endl;
	*/
	cout << "\n**********LIST OF COMMON CANCER TREATMENTS**********\n" << endl;
	for (int j = 1; j <= 7; j++)
	{
		cout << j << ". " << ctreatment[j] << endl;
	}
	//cout<<"\n\n Enter the cancer to find specific treatment"<<endl;

}

//**************************************************************************
**********MAIN function
int main() {



	cl[0].conclusion = "LOSS OF APETITE";
	cl[0].rule_num = 10;

	cl[1].conclusion = "BLOOD IN URINE";
	cl[1].rule_num = 20;
```

```
cl[2].conclusion = "HIGH BP";
cl[2].rule_num = 30;

cl[3].conclusion = "CANCER";
cl[3].rule_num = 40;

cl[4].conclusion = "UNABLE TO EAT";
cl[4].rule_num = 50;

cl[5].conclusion = "BLOOD WHILE URINATING";
cl[5].rule_num = 60;

cl[6].conclusion = "BONE OR JOINT PAIN";
cl[6].rule_num = 70;

cl[7].conclusion = "NOSE BLEED";
cl[7].rule_num = 80;

cl[8].conclusion = "CANCER";
cl[8].rule_num = 90;

cl[9].conclusion = "SEVERE HEADACHE";
cl[9].rule_num = 100;

cl[10].conclusion = "COGNITIVE PROBLEM";
cl[10].rule_num = 110;
```

```
cl[11].conclusion = "CANCER";
cl[11].rule_num = 120;

cl[12].conclusion = "PAIN IN THROAT";
cl[12].rule_num = 130;

cl[13].conclusion = "HOARSENESS";
cl[13].rule_num = 140;

cl[14].conclusion = "CANCER";
cl[14].rule_num = 150;


cl[15].conclusion = "NIGHT SWEAT";
cl[15].rule_num = 160;


cl[16].conclusion = "LOOSING WEIGHT";
cl[16].rule_num = 170;


cl[17].conclusion = "CANCER";
cl[17].rule_num = 180;
cl[18].conclusion = "CHEST PAIN";
cl[18].rule_num = 190;

cl[19].conclusion = "COUGH WITH BLOOD";
cl[19].rule_num = 200;
```

```
cl[20].conclusion = "WHEEZING";
cl[20].rule_num = 210;

cl[21].conclusion = "CANCER";
cl[21].rule_num = 220;

cl[22].conclusion = "YELLOW SKIN AND EYE";
cl[22].rule_num = 230;

cl[23].conclusion = "ITCHING";
cl[23].rule_num = 240;

cl[24].conclusion = "CANCER";
cl[24].rule_num = 250;

vl[0].status = "NI";
vl[0].value = " ";
vl[0].content = "CHANGES IN HEALTH";

vl[1].status = "NI";
vl[1].value = " ";
vl[1].content = "FATIGUE";
vl[2].status = "NI";
vl[2].value = " ";
vl[2].content = "WEIGHT LOSS";

vl[3].status = "NI";
vl[3].value = " ";
```

```
vl[3].content = "SWELLING OF ANKLES AND LEGS";

vl[4].status = "NI";
vl[4].value = " ";
vl[4].content = "A";

vl[5].status = "NI";
vl[5].value = " ";
vl[5].content = "BACK PAIN";

vl[6].status = "NI";
vl[6].value = " ";
vl[6].content = "SHORTNESS OF BREATH";

vl[7].status = "NI";
vl[7].value = " ";
vl[7].content = "DIZINESS";

vl[8].status = "NI";
vl[8].value = " ";
vl[8].content = "BLEEDING";

//NOT REQUIRED
vl[9].status = "NI";
vl[9].value = " ";
vl[9].content = " ";

//LOSS OF APETITE
```

```
vl[10].status = "NI";
vl[10].value = " ";
vl[10].content = " ";

//BLOOD IN URINE
vl[11].status = "NI";
vl[11].value = " ";
vl[11].content = " ";

//HIGH BP
vl[12].status = "NI";
vl[12].value = " ";
vl[12].content = " ";

//BONE OR JOINT PAIN
vl[13].status = "NI";
vl[13].value = " ";
vl[13].content = " ";

//NOSE BLEED
vl[14].status = "NI";
vl[14].value = " ";
vl[14].content = " ";

vl[15].status = "NI";
vl[15].value = " ";
vl[15].content = "MOUTH ULCER";
```

```
vl[16].status = "NI";
vl[16].value = " ";
vl[16].content = "RED SPOT";

//BLODD IN URINEE
vl[17].status = "NI";
vl[17].value = " ";
vl[17].content = " ";

//UNABLE TO EAT
vl[18].status = "NI";
vl[18].value = " ";
vl[18].content = " ";

vl[19].status = "NI";
vl[19].value = " ";
vl[19].content = "VOMITING";

//SEVERE HEADACHE
vl[20].status = "NI";
vl[20].value = " ";
vl[20].content = " ";

vl[21].status = "NI";
vl[21].value = " ";
vl[21].content = "BALANCE DISORDER";

vl[22].status = "NI";
```

```
vl[22].value = " ";
vl[22].content = "REDUCED SENSATION OR TOUCH";

//COGNITIVE PROBLEM
vl[23].status = "NI";
vl[23].value = " ";
vl[23].content = " ";

vl[24].status = "NI";
vl[24].value = " ";
vl[24].content = "BLURRED VISION";

vl[25].status = "NI";
vl[25].value = " ";
vl[25].content = "MEMORY LOSS";

vl[26].status = "NI";
vl[26].value = " ";
vl[26].content = "DIFFICULT TO SWALLOW";

vl[27].status = "NI";
vl[27].value = " ";
vl[27].content = "ENLARGED LYMPH NODES";

vl[28].status = "NI";
vl[28].value = " ";
vl[28].content = "PERSISTING COUGH";
```

```
//PAIN IN THROAT
vl[29].status = "NI";
vl[29].value = " ";
vl[29].content = " ";

//HOARSENESS
vl[30].status = "NI";
vl[30].value = " ";
vl[30].content = " ";

vl[31].status = "NI";
vl[31].value = " ";
vl[31].content = "SWOLLEN LYMPH NODES";

//NIGHT SWEAT
vl[32].status = "NI";
vl[32].value = " ";
vl[32].content = " ";

//LOOSING WEIGHT
vl[33].status = "NI";
vl[33].value = " ";
vl[33].content = " ";


vl[34].status = "NI";
vl[34].value = " ";
vl[34].content = "PAIN WHILE BREATHING";
```

```
//CHEST PAIN
        vl[35].status = "NI";
        vl[35].value = " ";
        vl[35].content = " ";

        //COUGH WITH BLOOD
        vl[36].status = "NI";
        vl[36].value = " ";
        vl[36].content = " ";

        //WHEEZING
        vl[37].status = "NI";
        vl[37].value = " ";
        vl[37].content = " ";

        vl[38].status = "NI";
        vl[38].value = " ";
        vl[38].content = "ROUGH SENSATION IN VOICE";

        vl[39].status = "NI";
        vl[39].value = " ";
        vl[39].content = "BELLY PAIN";

        vl[40].status = "NI";
        vl[40].value = " ";
        vl[40].content = "BLOATING";

        //YELLOW SKIN AND EYE
```

```
vl[41].status = "NI";
vl[41].value = " ";
vl[41].content = " ";

//ITCHING
vl[42].status = "NI";
vl[42].value = " ";
vl[42].content = " ";
clause_variable_list[0] = "CHANGES IN HEALTH";
clause_variable_list[1] = "FATIGUE";
clause_variable_list[2] = " ";
clause_variable_list[3] = " ";

clause_variable_list[4] = "LOSS OF APETITE";
clause_variable_list[5] = "WEIGHT LOSS";
clause_variable_list[6] = " ";
clause_variable_list[7] = " ";

clause_variable_list[8] = "BLOOD IN URINE";
clause_variable_list[9] = "SWELLING OF ANKLES AND LEGS";
clause_variable_list[10] = " ";
clause_variable_list[11] = " ";

clause_variable_list[12] = "HIGH BP";
clause_variable_list[13] = "A";
clause_variable_list[14] = "BACK PAIN";
clause_variable_list[15] = " ";
```

```
clause_variable_list[16] = "FATIGUE";
clause_variable_list[17] = "WEIGHT LOSS";
clause_variable_list[18] = " ";
clause_variable_list[19] = " ";

clause_variable_list[20] = "UNABLE TO EAT";
clause_variable_list[21] = "SHORTNESS OF BREATH";
clause_variable_list[22] = " ";
clause_variable_list[23] = " ";

clause_variable_list[24] = "BLOOD WHILE URINATING";
clause_variable_list[25] = "DIZINESS";
clause_variable_list[26] = " ";
clause_variable_list[27] = " ";

clause_variable_list[28] = "BONE OR JOINT PAIN";
clause_variable_list[29] = "BLEEDING";
clause_variable_list[30] = " ";
clause_variable_list[31] = " ";

clause_variable_list[32] = "NOSE BLEED";
clause_variable_list[33] = "MOUTH ULCER";
clause_variable_list[34] = "RED SPOT";
clause_variable_list[35] = " ";

clause_variable_list[36] = "FATIGUE";
clause_variable_list[37] = "LOSS OF APETITE";
clause_variable_list[38] = "VOMITING";
```

```
clause_variable_list[39] = " ";

clause_variable_list[40] = "SEVERE HEADACHE";
clause_variable_list[41] = "BALANCE DISORDER";
clause_variable_list[42] = "REDUCED SENSATION OR TOUCH";
clause_variable_list[43] = " ";

clause_variable_list[44] = "COGNITIVE PROBLEM";
clause_variable_list[45] = "BLURRED VISION";
clause_variable_list[46] = "MEMORY LOSS";
clause_variable_list[47] = " ";

//130
clause_variable_list[48] = "FATIGUE";
clause_variable_list[49] = "DIFFICULT TO SWALLOW";
clause_variable_list[50] = " ";
clause_variable_list[51] = " ";

//140
clause_variable_list[52] = "PAIN IN THROAT";
clause_variable_list[53] = "ENLARGED LYMPH NODES";
clause_variable_list[54] = " ";
clause_variable_list[55] = " ";

//150
clause_variable_list[56] = "HOARSENESS";
clause_variable_list[57] = "PERSISTING COUGH";
clause_variable_list[58] = " ";
```

```
clause_variable_list[59] = " ";

//160
clause_variable_list[60] = "FATIGUE";
clause_variable_list[61] = "LOSS OF APETITE";
clause_variable_list[62] = "WEIGHT LOSS";
clause_variable_list[63] = " ";

//170
clause_variable_list[64] = "NIGHT SWEAT";
clause_variable_list[65] = "SHORTNESS OF BREATH";
clause_variable_list[66] = " ";
clause_variable_list[67] = " ";

//180
clause_variable_list[68] = "LOOSING WEIGHT";
clause_variable_list[69] = "SWOLLEN LYMPH NODES";
clause_variable_list[70] = " ";
clause_variable_list[71] = " ";

//190
clause_variable_list[72] = "FATIGUE";
clause_variable_list[73] = "LOSS OF APETITE";
clause_variable_list[74] = "WEIGHT LOSS";
clause_variable_list[75] = " ";

//200
clause_variable_list[76] = "CHEST PAIN";
```

```
clause_variable_list[77] = "PAIN WHILE BREATHING";
clause_variable_list[78] = " ";
clause_variable_list[79] = " ";

//210
clause_variable_list[80] = "COUGH WITH BLOOD";
clause_variable_list[81] = "ROUGH SENSATION IN VOICE";
clause_variable_list[82] = " ";
clause_variable_list[83] = " ";

//220
clause_variable_list[84] = "WHEEZING";
clause_variable_list[85] = "";
clause_variable_list[86] = " ";
clause_variable_list[87] = " ";


//230
clause_variable_list[88] = "WEIGHT LOSS";
clause_variable_list[89] = "BELLY PAIN";
clause_variable_list[90] = "VOMITING";
clause_variable_list[91] = " ";


//240
clause_variable_list[92] = "YELLOW SKIN AND EYE";
clause_variable_list[93] = "FATIGUE";
clause_variable_list[94] = "LOSS OF APETITE";
```

```
clause_variable_list[95] = " ";



//250
clause_variable_list[96] = "ITCHING";
clause_variable_list[97] = "BLOATING";
clause_variable_list[98] = " ";
clause_variable_list[99] = " ";



//takes conclusion from user as i/p
string user;
cout << "Please enter condition variable (enter CANCER as the condition variable):" <<
endl;
cin >> user;
cout << endl;



check_conclusion_list(cl, user);



//################################################################  FORWARD
CHAINING
/******** INITIALIZATION SECTION **********/
fpointer = 1;
bponiter = 1;

for (i = 1; i < 41; i++)
        clausevarlist[i] = "";
```

```
for (i = 1; i < 9; i++)
        conclusionlist[i] = "";
for (i = 1; i < 37; i++)
        instlt[i] = 0;
for (i = 1; i < 9; i++)
        variablelist[i] = "";

/****** comment 367 **********/
for (i = 1; i < 9; i++)
{
        variablelist[i] = "cancer";
}

clausevarlist[1] = "cancer";
clausevarlist[6] = "cancer";
clausevarlist[11] = "cancer";
clausevarlist[16] = "cancer";
clausevarlist[21] = "cancer";
clausevarlist[26] = "cancer";
clausevarlist[31] = "cancer";
clausevarlist[36] = "cancer";


conclusionlist[1] = "Stereotactic Radiation Theraphy";
conclusionlist[2] = "Bone Marrow Stimulant";
conclusionlist[3] = "Pulmonary Lobectomy";
conclusionlist[4] = "Hematopoictic";
conclusionlist[5] = "Craniotomy";
```

```
        conclusionlist[6] = "Thyroidectomy";
        conclusionlist[7] = "Ureterostomy";
        conclusionlist[8] = "Liver Transplantation and Hepatectomy";

        /****** INFERENCE SECTION **********/
        // cout<<"Type\"Cancer\" as condition variable"<<endl;

        conditionvariable = "cancer";
        //cout <<conditionvariable<<endl;
        display();
        conclusionlist[bponiter] = conditionvariable;

        bponiter = bponiter + 1;

        rulenumber = 1;
        clausenumber = 1;

        f = 1;
b496: search();
        /* point to first clause in statement */
        clausenumber = 1;
        if (rulenumber != 0)
                /* more statements */
        {
                /* locate the clause */
                i = 5 * (rulenumber - 1) + clausenumber;
                /* clause variable */
                variablename = clausevarlist[i];
```

```
/* are there any more clauses for this statement */
while (variablename != "")
        /* more clauses */
{
        /* check instantiation of this clause */
        check_instantiation();
        clausenumber = clausenumber + 1;
        /* check next clause */
        i = 5 * (rulenumber - 1) + clausenumber;
        variablename = clausevarlist[i];
}

/* no more clauses - check IF part of statement */
s = 0;
/* sample IF-THEN statements from the position knowledge base */
switch (rulenumber)
{
        /* statement 1 */
        /***** comment 1500 *****/
case 1: if (cancer == "KIDNEY CANCER") s = 1;
        break;
case 2: if (cancer == "LYMPHOMA CANCER") s = 1;
        break;
case 3: if (cancer == "LUNG CANCER") s = 1;
        break;
case 4: if (cancer == "BLOOD CANCER") s = 1;
        break;
case 5: if (cancer == "BRAIN CANCER") s = 1;
```

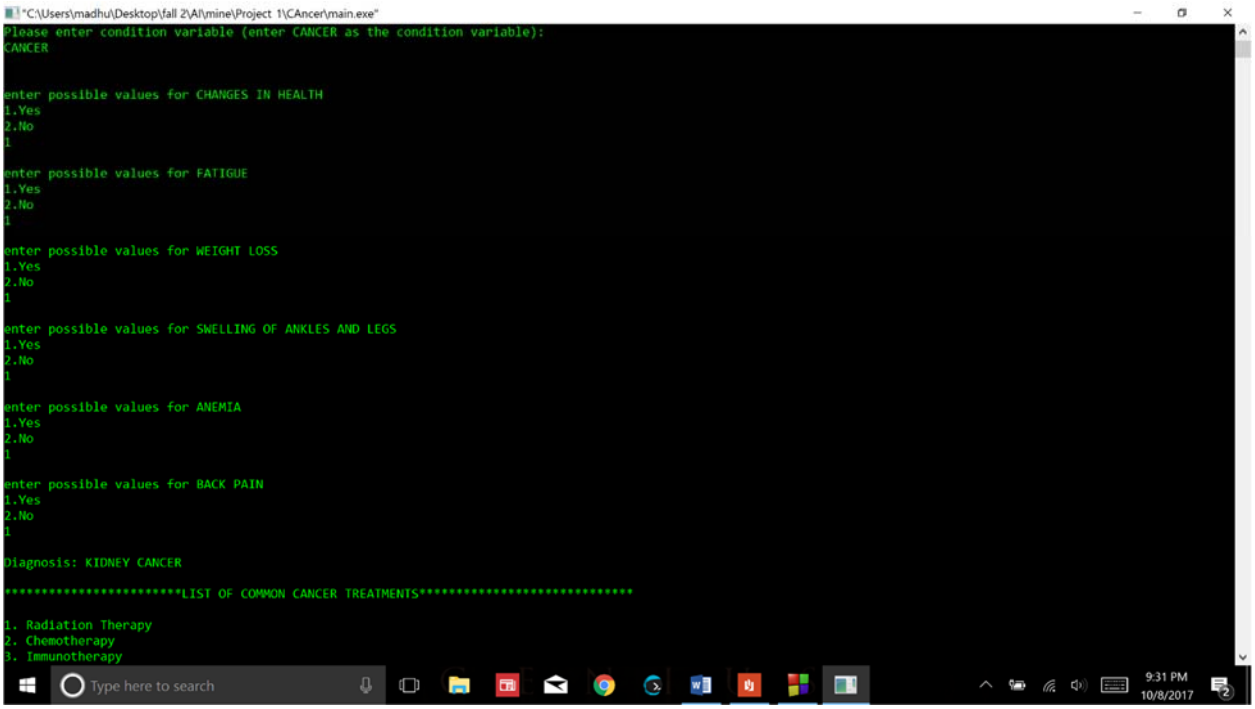
```
            break;
        case 6: if (cancer == "THYROID CANCER") s = 1;
            break;
        case 7: if (cancer == "BLADDER CANCER") s = 1;
            break;
        case 8: if (cancer == "LIVER CANCER") s = 1;
            break;
        }

        /* see if the THEN part should be inovked, i.e., s=1 */
        if (s != 1)
        {
            f = rulenumber + 1;
            goto b496;
        }

        /* invoke THEN part */
        switch (rulenumber)
        {
            /********** comment 1500 **********/
            /* put variable on the conclusion variable queue */
        case 1:
            cout << " CANCER TYPE : KIDNEY" << endl;
            //treatment[] = {'Stereotactic Radiation Theraphy','Radio Frequency Ablation','Surgery','Chemotheraphy','Immunotheraphy'};
            //for(int i=0;i<sizeof(treatment);i++)
            //{
            //  cout << "\n Treatment is: " << treatment << endl;
```

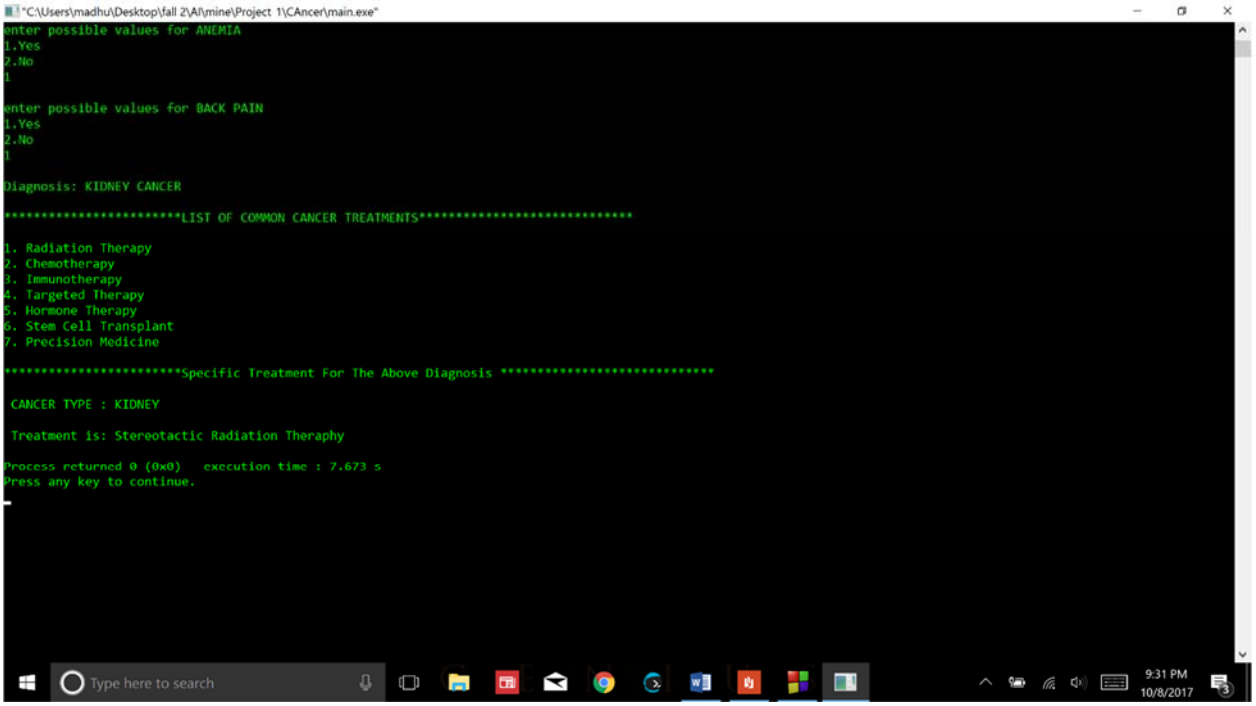
```
        //}
        treatment = "Stereotactic Radiation Theraphy";
        cout << "\n Treatment is: " << treatment << endl;
        variablename = treatment;
        instantiate();
        break;

case 2:
        cout << " CANCER TYPE  : LYMPHOMA " << endl;
        treatment = "Bone Marrow Stimulant";
        cout << "\n Treatment is: " << treatment << endl;
        variablename = treatment;
        instantiate();
        break;

case 3:
        cout << " CANCER TYPE : LUNG " << endl;
        treatment = "Pulmonary Lobectomy";
        cout << "\n Treatment is: " << treatment << endl;
        variablename = treatment;
        instantiate();
        break;

case 4:
        cout << " CANCER TYPE : BLOOD " << endl;
        treatment = "Hematopoictic";
        cout << "\n Treatment is: " << treatment << endl;
        variablename = treatment;
```

```
        instantiate();
        break;

case 5:
        cout << " CANCER TYPE : BRAIN " << endl;
        treatment = "Craniotomy";
        cout << "\n Treatment is: " << treatment << endl;
        variablename = treatment;
        instantiate();
        break;

case 6:
        cout << " CANCER TYPE : THYROID " << endl;
        treatment = "Thyroidectomy";
        cout << "\n Treatment is: " << treatment << endl;
        variablename = treatment;
        instantiate();
        break;

case 7:
        cout << " CANCER TYPE : BLADDER " << endl;
        treatment = "Ureterostomy";
        cout << "\n Treatment is: " << treatment << endl;
        variablename = treatment;
        instantiate();
        break;

case 8:
```

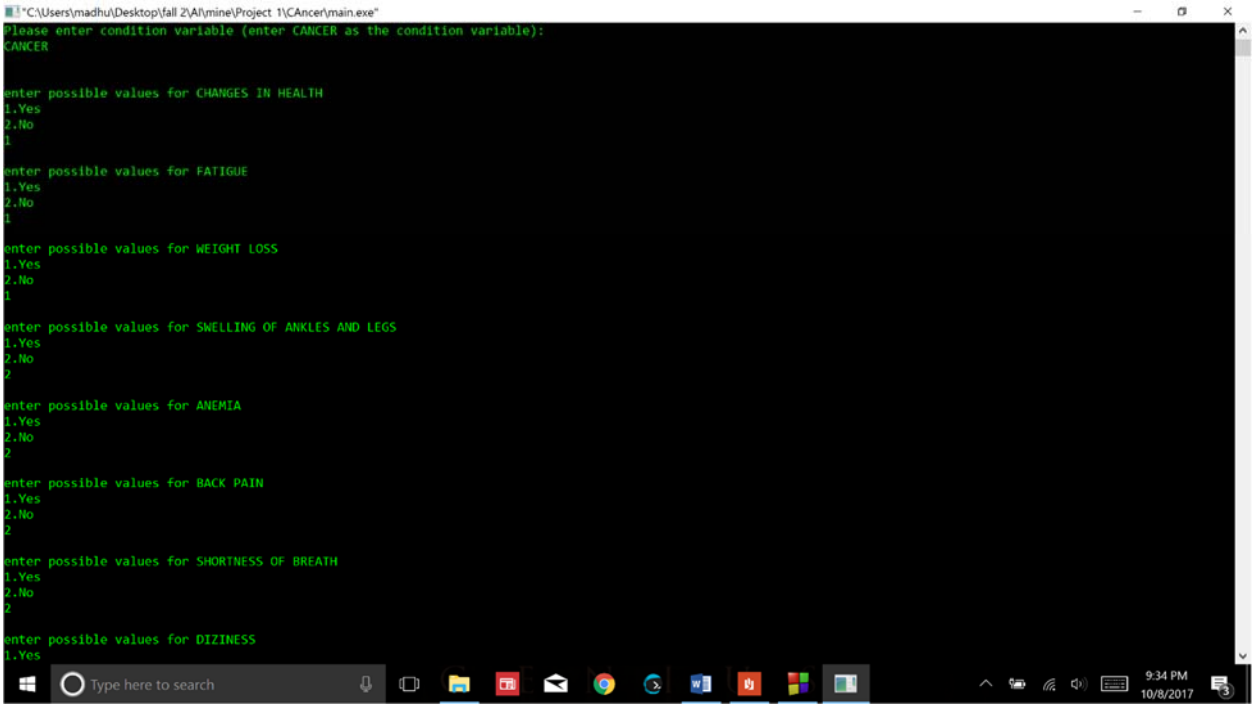
```
                cout << " CANCER TYPE : LIVER " << endl;
                treatment = "Liver Transplantation and Hepatectomy";
                cout << "\n Treatment is: " << treatment << endl;
                exit(0);
                //variablename = treatment;
                //instantiate();
                // break;

        }
        f = rulenumber + 1;
        goto b496;
    }


    fpointer = fpointer + 1;
    if (fpointer < bponiter)
    {
        f = 1;
        goto b496;
    }
    return 0;
}
```

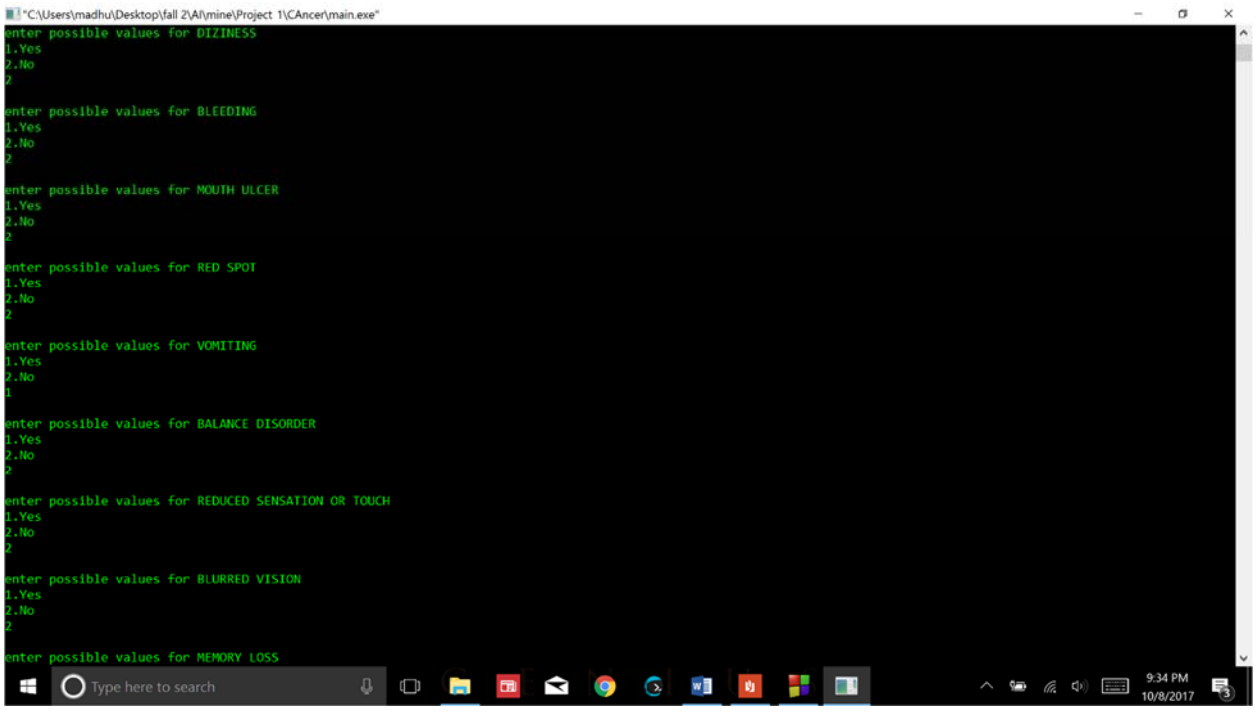
**Sample Output -1:**

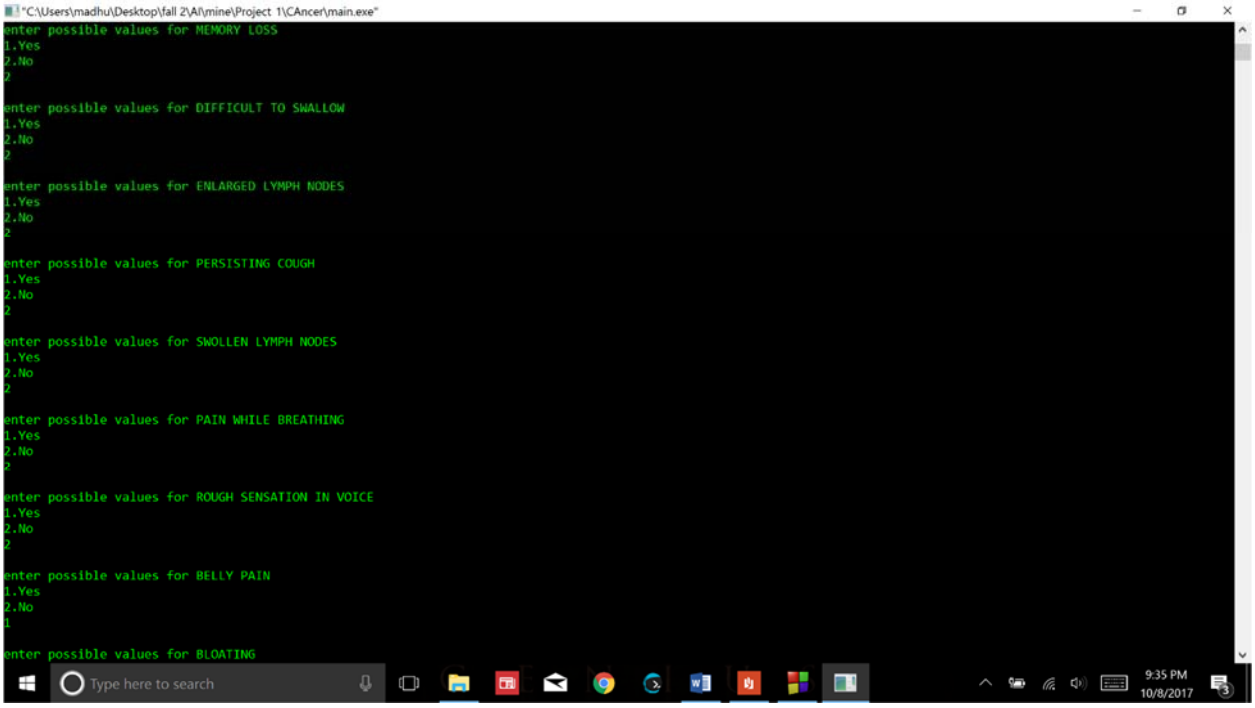


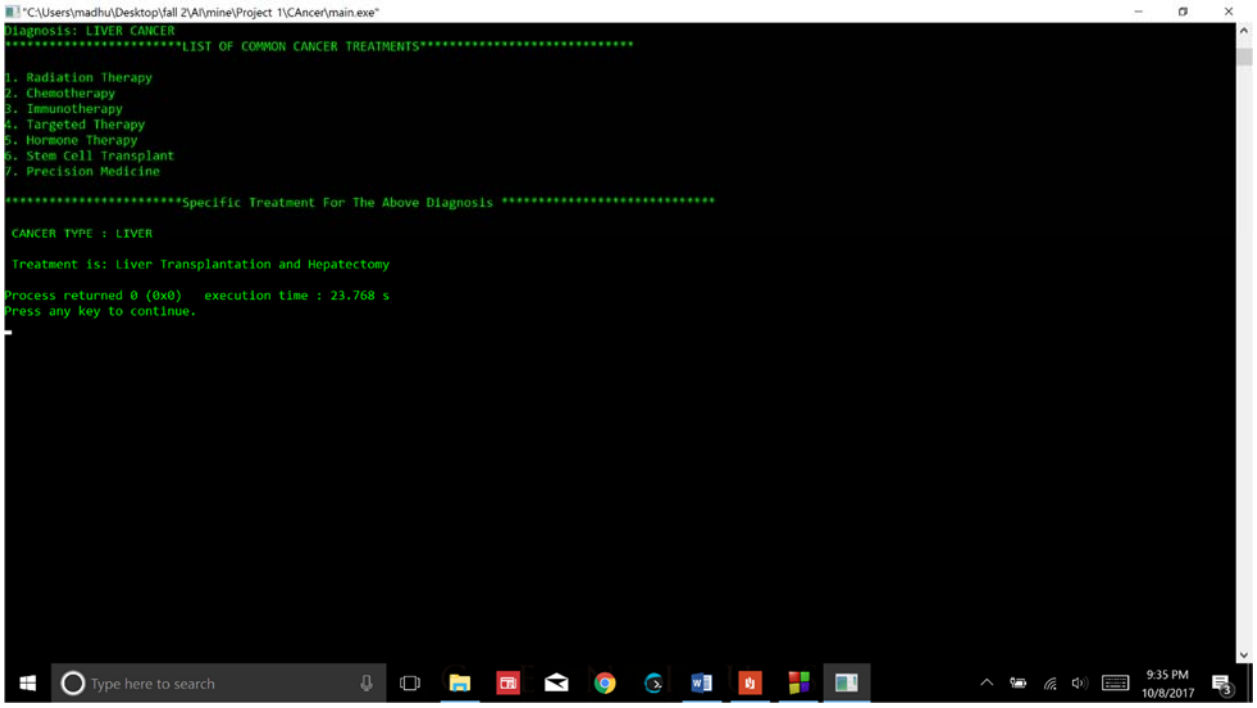

"C:\Users\madhu\Desktop\fall 2\AI\mine\Project 1\CAncer\main.exe"
enter possible values for ANEMIA
1.Yes
2.No
1
enter possible values for BACK PAIN
1.Yes
2.No
1
Diagnosis: KIDNEY CANCER
**********LIST OF COMMON CANCER TREATMENTS**********
1. Radiation Therapy
2. Chemotherapy
3. Immunotherapy
4. Targeted Therapy
5. Hormone Therapy
6. Stem Cell Transplant
7. Precision Medicine
**********Specific Treatment For The Above Diagnosis **********
CANCER TYPE : KIDNEY
Treatment is: Stereotactic Radiation Theraphy
Process returned 0 (0x0) execution time : 7.673 s
Press any key to continue.
Type here to search
9:31 PM
10/8/2017

## Sample Output – 2:

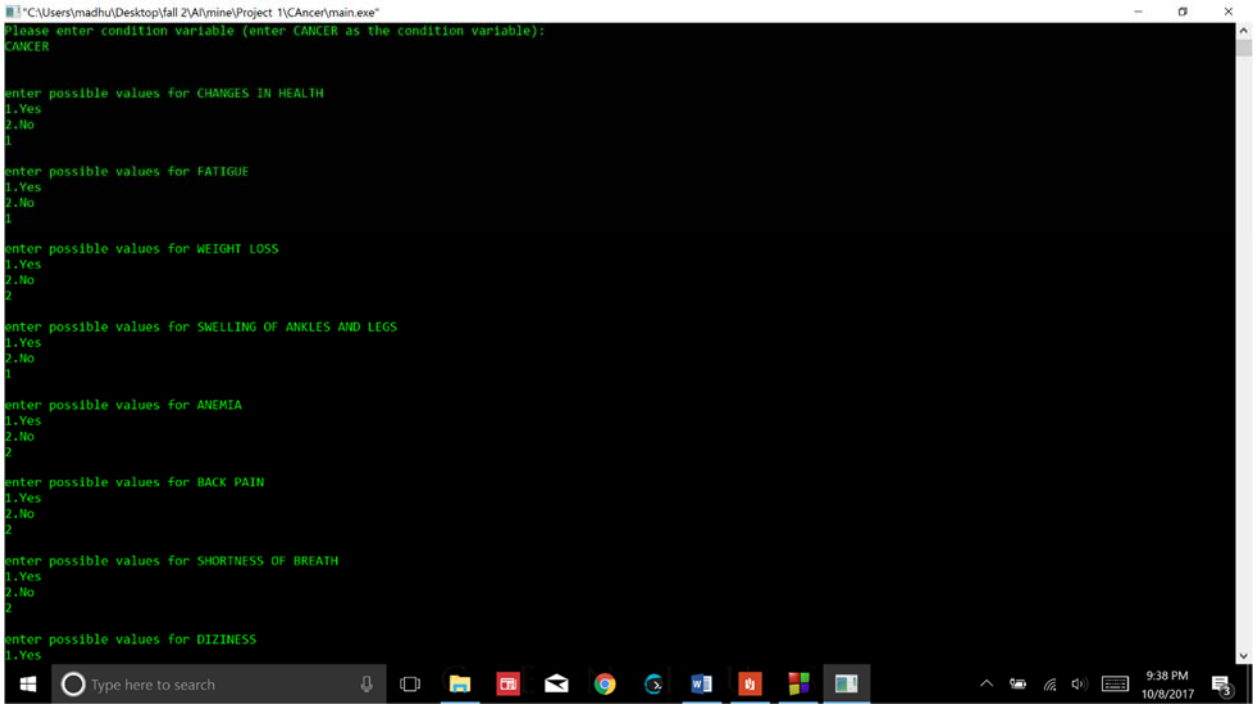


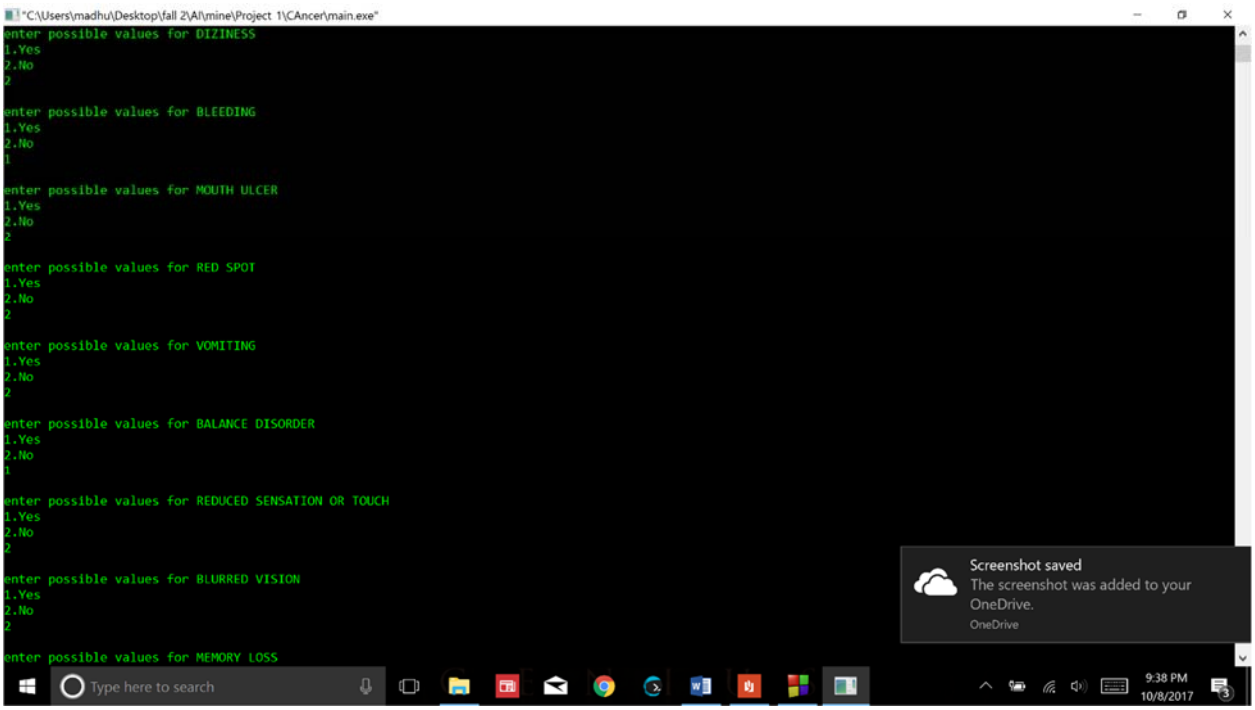



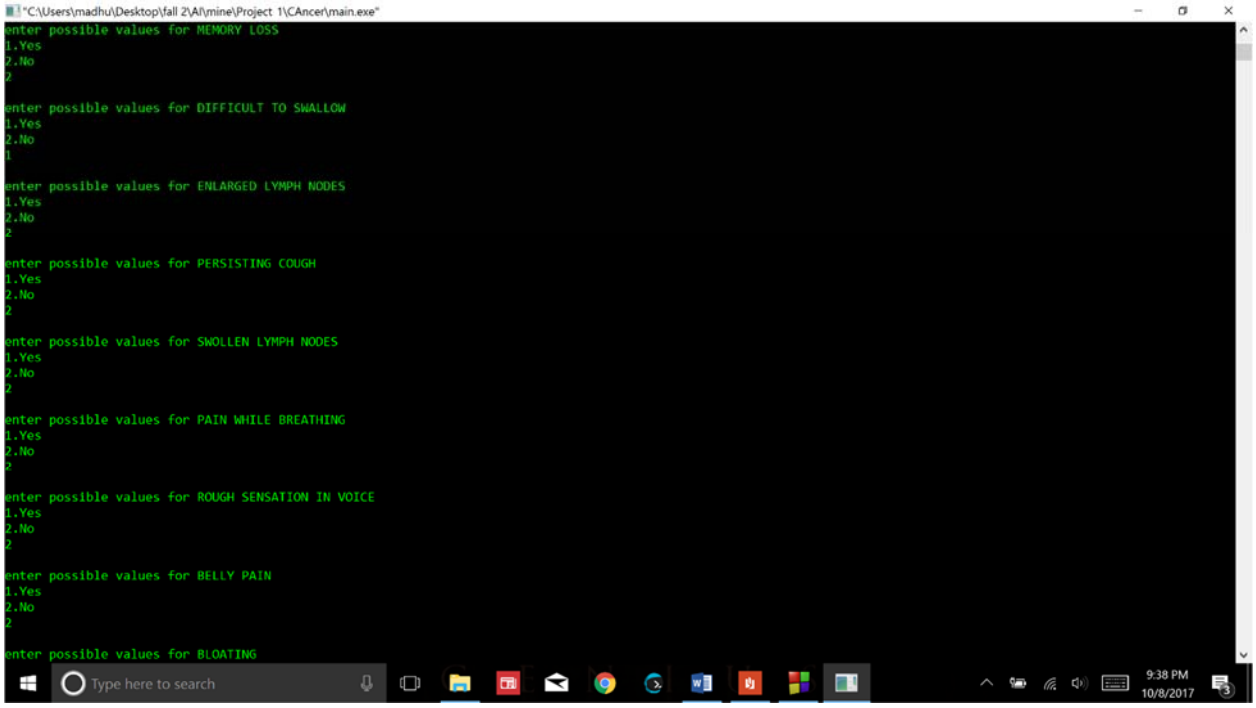

"C:\Users\madhu\Desktop\fall 2\AI\mine\Project 1\CAncer\main.exe"
enter possible values for MEMORY LOSS
1.Yes
2.No
2
enter possible values for DIFFICULT TO SWALLOW
1.Yes
2.No
2
enter possible values for ENLARGED LYMPH NODES
1.Yes
2.No
2
enter possible values for PERSISTING COUGH
1.Yes
2.No
2
enter possible values for SWOLLEN LYMPH NODES
1.Yes
2.No
2
enter possible values for PAIN WHILE BREATHING
1.Yes
2.No
2
enter possible values for ROUGH SENSATION IN VOICE
1.Yes
2.No
2
enter possible values for BELLY PAIN
1.Yes
2.No
1
enter possible values for BLOATING
Type here to search
9:35 PM
10/8/2017

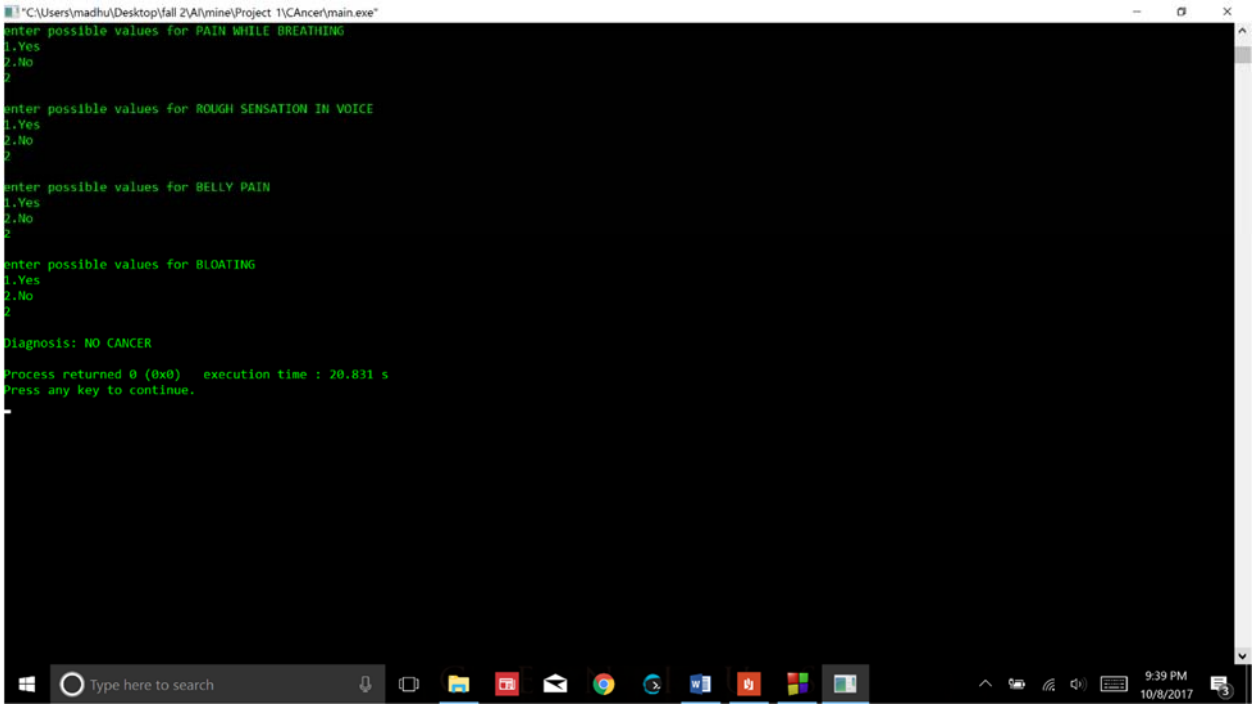

"C:\Users\madhu\Desktop\fall 2\AI\mine\Project 1\CAncer\main.exe"
Diagnosis: LIVER CANCER
**********LIST OF COMMON CANCER TREATMENTS**********
1. Radiation Therapy
2. Chemotherapy
3. Immunotherapy
4. Targeted Therapy
5. Hormone Therapy
6. Stem Cell Transplant
7. Precision Medicine
**********Specific Treatment For The Above Diagnosis **********
CANCER TYPE : LIVER
Treatment is: Liver Transplantation and Hepatectomy
Process returned 0 (0x0) execution time : 23.768 s
Press any key to continue.
Type here to search
9:35 PM
10/8/2017

## Sample Output – 3




"C:\Users\madhu\Desktop\fall 2\AI\mine\Project 1\CAncer\main.exe"
enter possible values for MEMORY LOSS
1.Yes
2.No
2
enter possible values for DIFFICULT TO SWALLOW
1.Yes
2.No
1
enter possible values for ENLARGED LYMPH NODES
1.Yes
2.No
2
enter possible values for PERSISTING COUGH
1.Yes
2.No
2
enter possible values for SWOLLEN LYMPH NODES
1.Yes
2.No
2
enter possible values for PAIN WHILE BREATHING
1.Yes
2.No
2
enter possible values for ROUGH SENSATION IN VOICE
1.Yes
2.No
2
enter possible values for BELLY PAIN
1.Yes
2.No
2
enter possible values for BLOATING
Type here to search
9:38 PM
10/8/2017


"C:\Users\madhu\Desktop\fall 2\AI\mine\Project 1\CAncer\main.exe"
enter possible values for PAIN WHILE BREATHING
1.Yes
2.No
2
enter possible values for ROUGH SENSATION IN VOICE
1.Yes
2.No
2
enter possible values for BELLY PAIN
1.Yes
2.No
2
enter possible values for BLOATING
1.Yes
2.No
2
Diagnosis: NO CANCER
Process returned 0 (0x0) execution time : 20.831 s
Press any key to continue.
Type here to search
9:39 PM
10/8/2017

# CHAPTER 7

# CONCLUSION

I would like to extend my thanks to Prof. Dr. Moonis Ali, for providing an opportunity to work on the concepts of artificial intelligence in real time. This project which I have implemented will definitely have a huge impact on how I will build other projects in the future. This is a full use of programming and as well as rational thinking. This made me to realize how do a machine think and how to develop an intelligent system that could solve a problem. Future implementation of this project on a web-interface will be helpful for the naïve computer users on how to diagnose cancer at an early stage itself and help people by also providing them with possible treatments to stop the growth of cancer.

# CHAPTER 8

# REFERENCES


- We found information regarding the symptoms and treatments by searching them in www.google.com.

- For collecting info and for building knowledge base for the system we used http://www.cancercenter.com

- For understanding how the Backward Chaining algorithm and Forward Chaining algorithm works we used the ppts provided on TRACS

- For other information we used the book Artificial Intelligence (2nd ed) by Elaine Rich and Kevin Knight, McGraw Hill (1991). ISBN 0-07-100894-2