Name: Ojus Jaiswal **Subject: Laboratory practice II**

Roll No: TACO19108 **Artificial Intelligence Lab**

Assignment 4: Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph coloring problem.

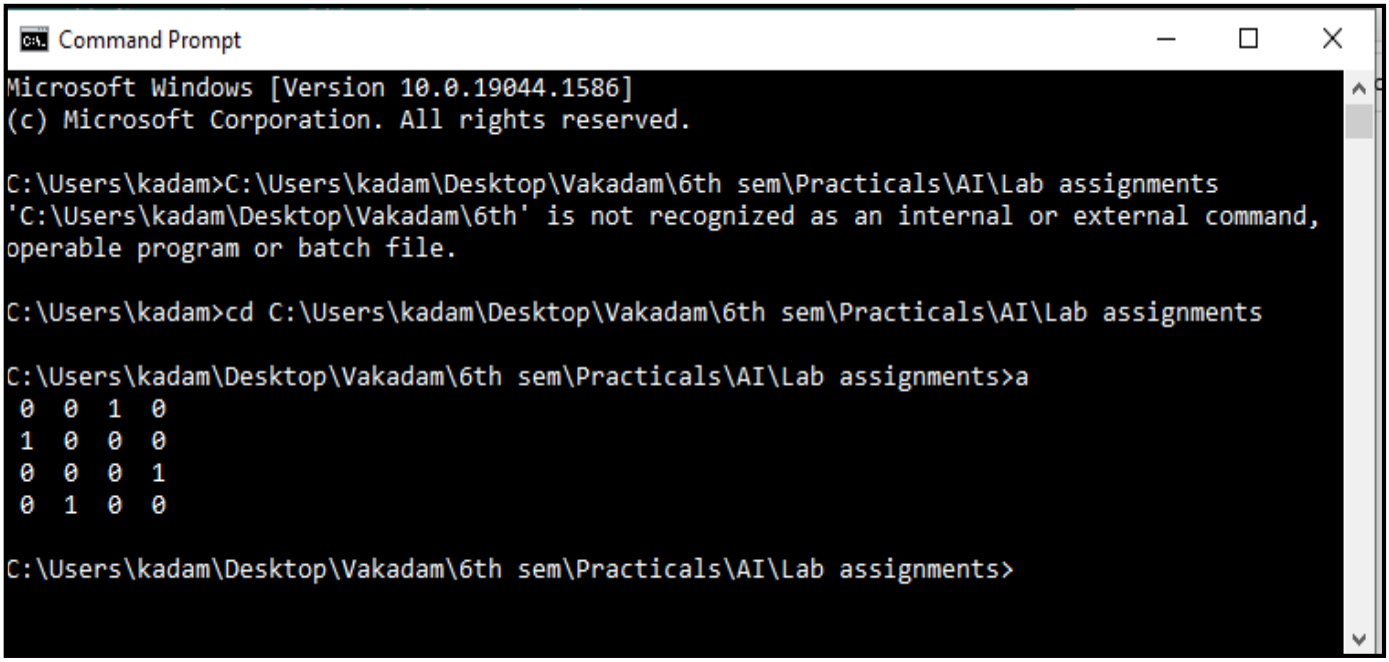
Code: Backtracking.c

```
/* C/C++ program to solve N Queen Problem using backtracking */
#define N 4
#include <stdbool.h>
#include <stdio.h>

/* A utility function to print solution */ void printSolution(int board[N][N])
{
for (int i = 0; i < N; i++) { for (int j = 0; j < N; j++)
printf(" %d ", board[i][j]); printf("\n");
}
}

/* A utility function to check if a queen canbe placed on board[row][col]. Note that this function is called when "col"
queens arealready placed in columns from 0 to col -1. So we need to check only left side for attacking queens */

bool isSafe(int board[N][N], int row, int col)
{
int i, j;

/* Check this row on left side */ for (i = 0; i < col; i++)
if (board[row][i]) return false;

/* Check upper diagonal on left side */
for (i = row, j = col; i >= 0 && j >= 0; i--, j--) if (board[i][j])
return false;

/* Check lower diagonal on left side */
for (i = row, j = col; j >= 0 && i < N; i++, j--) if (board[i][j])
return false;

return true;
}

/* A recursive utility function to solve N Queen problem */
bool solveNQUtil(int board[N][N], int col)
{
/* base case: If all queens are placed then return true */
if (col >= N) return true;

/* Consider this column and try placing this queen in all rows one by one */
for (int i = 0; i < N; i++) {
/* Check if the queen can be placed on board[i][col] */
if (isSafe(board, i, col)) {
```

```
/* Place this queen in board[i][col] */ board[i][col] = 1;

/* recur to place rest of the queens */ if (solveNQUtil(board, col + 1))
return true;

/* If placing queen in board[i][col] doesn't lead to a solution, then remove queen from board[i][col] */
board[i][col] = 0; // BACKTRACK
}
}

/* If the queen cannot be placed in any row in this column col then return false */
return false;
}

/* This function solves the N Queen problem using  Backtracking. It mainly uses solveNQUtil() to solve the
problem. It returns false if queens cannot be placed, otherwise, return true and prints placement of queens in the
form of 1s. Please note that there may be more than one solutions, this function prints one of the feasible solutions.*/
bool solveNQ()
{
int board[N][N] = { { 0, 0, 0, 0 },
{ 0, 0, 0, 0 },
{ 0, 0, 0, 0 },
{ 0, 0, 0, 0 } };

if (solveNQUtil(board, 0) == false) { printf("Solution does not exist"); return false;
}

printSolution(board); return true;
}

// driver program to test above function
int main()
{
solveNQ(); return 0;

}
```

Output:

gcc Backtracking.c



Name: Ojus Jaiswal **Subject: Laboratory practice II**

Roll No: TACO19108 **Artificial Intelligence Lab**

Assignment 5: Develop an elementary chatbot for any suitable customer interaction application.

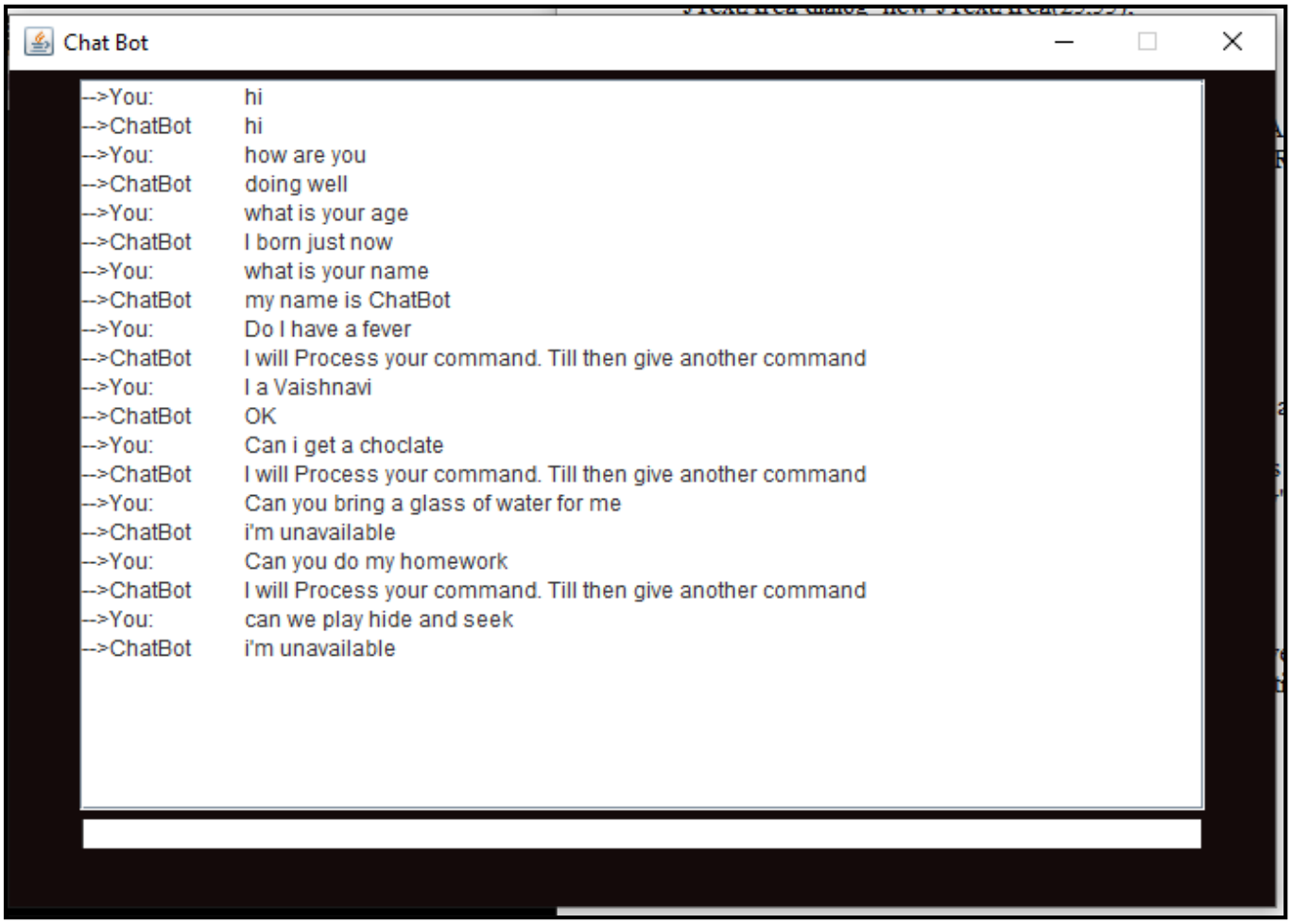
Code:
ChatBot.java

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextArea;
import javax.swing.JScrollPane;

import java.applet.Applet;
import java.applet.AudioClip;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Toolkit;

import java.awt.event.KeyListener;
import java.awt.event.KeyEvent;

import java.io.File;
import java.lang.Math;

public class ChatBot extends JFrame implements KeyListener{

JPanel p=new JPanel();
JFrame f=new JFrame();

JTextArea dialog=new JTextArea(25,55);
JTextArea input=new JTextArea(1,55);
JScrollPane scroll=new JScrollPane(
dialog,
JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
JScrollPane.HORIZONTAL_SCROLLBAR_NEVER
);

String[][] chatBot={
//standard greetings
{"hi","hello","hey"},
{"hi","hello","hey"},
//question greetings
{"how are you","how r you","how r u","how are u"},
{"good","doing well"},
{"what is your name","name"},{"my name is ChatBot"},
{"age","what is your age"},{"I born just now"},
//yes
{"yes"},
{"no","NO","NO!!!!!!!"},
//default
{"I will Process your command. Till then give another command","OK",
"i'm unavailable","Please contact the respective person for this"}
};
```

```
public static void main(String[] args){
new ChatBot();
}

public ChatBot(){
super("Chat Bot");
setSize(700,500);


Dimension d=Toolkit.getDefaultToolkit().getScreenSize();

//f.setSize(d.width,d.height);
setLocation((d.width-700)/2,(d.height-500)/2);


setIconImage(Toolkit.getDefaultToolkit().getImage("C:\\Users\\MAHESH\\Documents\\Chatbotproject\\src\\chatbot
image.png"));
setResizable(false);
setDefaultCloseOperation(EXIT_ON_CLOSE);

dialog.setEditable(false);
input.addKeyListener(this);

p.add(scroll);
p.add(input);
p.setBackground(new Color(20,10,10));
add(p);

setVisible(true);
}
public void keyPressed(KeyEvent e){
if(e.getKeyCode()==KeyEvent.VK_ENTER){
        input.setEditable(false);
        //.....grab quote.....
        String quote=input.getText();
        input.setText("");
        addText("-->You:\t"+quote);
        quote.trim();
        while(
                quote.charAt(quote.length()-1)=='!' ||
                quote.charAt(quote.length()-1)=='.' ||
                quote.charAt(quote.length()-1)=='?'
        ){
                quote=quote.substring(0,quote.length()-1);
        }
        quote.trim();
        byte response=0;
        /*
        0:we're searching through chatBot[][] for matches
        1:we didn't find anything
        2:we did find something
        */
        //-----check for matches----
        int j=0;//which group we're checking
        while(response==0){
                if(inArray(quote.toLowerCase(),chatBot[j*2])){
                        response=2;
                        int r=(int)Math.floor(Math.random()*chatBot[(j*2)+1].length);
                        addText("\n-->ChatBot\t"+chatBot[(j*2)+1][r]);
                }
                j++;
                if(j*2==chatBot.length-1 && response==0){
```

```
                        response=1;
                }
        }
                        //.....default.....
        if(response==1){
                int r=(int)Math.floor(Math.random()*chatBot[chatBot.length-1].length);
                addText("\n-->ChatBot\t"+chatBot[chatBot.length-1][r]);
        }
        addText("\n");
}
}
public void keyReleased(KeyEvent e){
if(e.getKeyCode()==KeyEvent.VK_ENTER){
        input.setEditable(true);
}
}
public void keyTyped(KeyEvent e){}
public void addText(String str){
dialog.setText(dialog.getText()+str);
}
public boolean inArray(String in,String[] str){
boolean match=false;
for(int i=0;i<str.length;i++){
        if(str[i].equals(in)){
                match=true;
        }
}
return match;
}}
```

Output:



Assignment 6:

```
hypothesis(Disease),
write('I believe that the patient have'),
write(Disease),
nl,
write('TAKE CARE '),
undo.

/*Hypothesis that should be tested*/
hypothesis(cold) :- cold, !.
hypothesis(flu) :- flu, !.
hypothesis(typhoid) :- typhoid, !.
hypothesis(measles) :- measles, !.
hypothesis(malaria) :- malaria, !.
hypothesis(unknown). /* no diagnosis*/

/*Hypothesis Identification Rules*/

cold :-
verify(headache),
verify(runny_nose),
verify(sneezing),
verify(sore_throat),
write('Advices and Sugestions:'),
nl,
write('1: Tylenol/tab'),
nl,
write('2: panadol/tab'),
nl,
write('3: Nasal spray'),
nl,
write('Please weare warm cloths Because'),
nl.

flu :-
verify(fever),
verify(headache),
verify(chills),
verify(body_ache),
write('Advices and Sugestions:'),
```

```
nl,
write('1: Tamiflu/tab'),
nl,
write('2: panadol/tab'),
nl,
write('3: Zanamivir/tab'),
nl,
write('Please take a warm bath and do salt gargling Because'),
nl.

typhoid :-
verify(headache),
verify(abdominal_pain),
verify(poor_appetite),
verify(fever),
write('Advices and Sugestions:'),
nl,
write('1: Chloramphenicol/tab'),
nl,
write('2: Amoxicillin/tab'),
nl,
write('3: Ciprofloxacin/tab'),
nl,
write('4: Azithromycin/tab'),
nl,
write('Please do complete bed rest and take soft Diet Because'),
nl.

measles :-
verify(fever),
verify(runny_nose),
verify(rash),
verify(conjunctivitis),
write('Advices and Sugestions:'),
nl,
write('1: Tylenol/tab'),
nl,
write('2: Aleve/tab'),
nl,
write('3: Advil/tab'),
nl,
```

```
write('4: Vitamin A'),
nl,
write('Please Get rest and use more liquid Because'),
nl.

malaria :-
verify(fever),
verify(sweating),
verify(headache),
verify(nausea),
verify(vomiting),
verify(diarrhea),
write('Advices and Sugestions:'),
nl,
write('1: Aralen/tab'),
nl,
write('2: Qualaquin/tab'),
nl,
write('3: Plaquenil/tab'),
nl,
write('4: Mefloquine'),
nl,
write('Please do not sleep in open air and cover your full skin Because'),
nl.

/* how to ask questions */
ask(Question) :-
write('Does the patient have following symptom:'),
write(Question),
write('? '),
read(Response),
nl,
( (Response == yes ; Response == y)
->
assert(yes(Question)) ;
assert(no(Question)), fail).

:- dynamic yes/1,no/1.
/*How to verify something */
verify(S) :-
(yes(S)
->
```

```
true ;
(no(S)
->
fail ;
ask(S))).
/* undo all yes/no assertions*/
undo :- retract(yes(_)),fail.
undo :- retract(no(_)),fail.
undo.
```

OUTPUT: