****

**School of Computing and Information Technology**

**Intelligent Agents Assignment 1**

6th Semester, Sections B and C Subject Teacher: Prof. Nikhil

Date of Submission: 20th Feb, 2019

Q1. Formulate a solution to the Question – Answering problem. Give the benefits and drawbacks of your solution.

Q2. Demonstrate the use of AI technique for solving the Water Jug Problem. Specify the initial and goal states along with the Production set.

Q3. Differentiate between the following giving examples of the problems in which each of these is suitable:

1. Breadth First Search and Depth First Search
2. Exhaustive Search and Heuristic Search

Q4. Apply Hill Climbing Heuristic to solve the blocks world problem, demonstrate the entire set of steps in solution.

Q5. In the Steepest Ascent Heuristic, what is the meaning of following terms:

Local Maxima, Plateau, Ridge. How are these problems overcome?

Q6. Implement the solution to tic- tac- toe problem using a programming language of your choice.

ADVANTAGES:  
  
Answering machines permit us to get the answers or important informationwhen we are not in the state or busy to attend the phone calls.   
  
Answering machines have made the lives pretty easier as they kept the record of the people who have called aur left messages in your absence.  
  
You can shun the unwanted calls with the help of the advance versions answering machines as it shows the id and phone number of the caller.  
  
With the help magic blend of the caller ID and answering machine you can be ready to receive anything good or bad calls without being anxious about missing any important call.  
  
  
DISADVANTAGES:  
  
There is lack of privacy in using an answering machine as anyone at home can listen to someone's private messages.  
  
Although answering machines can help you in a number of ways, due to increase cell phones , its features does not support the modern world criteria.   
  
If you are not at home you cannot respond to the important messages immediately.

Tic-Tac-Toe Program using

# random number in Python

# importing all necessary libraries

import numpy as np

import random

from time import sleep

# Creates an empty board

def create\_board():

    return(np.array([[0, 0, 0],

                     [0, 0, 0],

                     [0, 0, 0]]))

# Check for empty places on board

def possibilities(board):

    l = []

    for i in range(len(board)):

        for j in range(len(board)):

            if board[i][j] == 0:

                l.append((i, j))

    return(l)

# Select a random place for the player

def random\_place(board, player):

    selection = possibilities(board)

    current\_loc = random.choice(selection)

    board[current\_loc] = player

    return(board)

# Checks whether the player has three

# of their marks in a horizontal row

def row\_win(board, player):

    for x in range(len(board)):

        win = True

        for y in range(len(board)):

            if board[x, y] != player:

                win = False

                continue

        if win == True:

            return(win)

    return(win)

# Checks whether the player has three

# of their marks in a vertical row

def col\_win(board, player):

    for x in range(len(board)):

        win = True

        for y in range(len(board)):

            if board[y][x] != player:

                win = False

                continue

        if win == True:

            return(win)

    return(win)

# Checks whether the player has three

# of their marks in a diagonal row

def diag\_win(board, player):

    win = True

    for x in range(len(board)):

        if board[x, x] != player:

            win = False

    return(win)

# Evaluates whether there is

# a winner or a tie

def evaluate(board):

    winner = 0

    for player in [1, 2]:

        if (row\_win(board, player) or

            col\_win(board,player) or

            diag\_win(board,player)):

            winner = player

    if np.all(board != 0) and winner == 0:

        winner = -1

    return winner

# Main function to start the game

def play\_game():

    board, winner, counter = create\_board(), 0, 1

    print(board)

    sleep(2)

    while winner == 0:

        for player in [1, 2]:

            board = random\_place(board, player)

            print("Board after " + str(counter) + " move")

            print(board)

            sleep(2)

            counter += 1

            winner = evaluate(board)

            if winner != 0:

                break

    return(winner)

# Driver Code

print("Winner is: " + str(play\_game()))