Practical No. - 9

Name- Bhavesh Kewalramani

Roll No.- A-25

Section- A

Semester- 6th

Shift- 1st

Aim:

Write a program to solve Tic-Tac-Toe by using Min-Max algorithm and alpha-beta pruning

Code:

```
import random

import time


def drawBoard(board):

    # This function prints out the board that it was passed.

    # "board" is a list of 10 strings representing the board (ignore
index 0)

    print(board[1] + '|' + board[2] + '|' + board[3])

    print('-+-+-')

    print(board[4] + '|' + board[5] + '|' + board[6])

    print('-+-+-')
```

```
    print(board[7] + '|' + board[8] + '|' + board[9])



def inputPlayerLetter():

    # Lets the player type which letter they want to be.

    # Returns a list with the player's letter as the first item, and the computer's letter as the second.

    letter=''

    while not(letter=='X' or letter=='O'):

        print("Do you want to be 'X' or 'O'?")

        letter = input().upper()


    if letter == 'X':

        return ['X','O']

    else:

        return ['O','X']



def whoGoesFirst():

    print('Do you want to go first? (Yes or No)')
```

```
    if  input().lower().startswith('y'):
        return 'player'
    else:
        return 'computer'
    '''
    # Randomly choose the player who goes first.
    if random.randint(0,1) == 0:
        return 'computer'
    else:
        return 'player'
    '''


def playAgain():
    # This function returns True if the player wants to play again,
otherwise it returns False.
    print('Do you want to play again? (Yes or No)')
    return input().lower().startswith('y')
```

```
def makeMove(board, letter, move):
    board[move] = letter



def isWinner(board,letter):
    # Given a board and a player's letter, this function returns True if that player has won.
    return ((board[1]==letter and board[2]==letter and board[3]==letter) or
            (board[4]==letter and board[5]==letter and board[6]==letter) or
            (board[7]==letter and board[8]==letter and board[9]==letter) or
            (board[1]==letter and board[4]==letter and board[7]==letter) or
            (board[2]==letter and board[5]==letter and board[8]==letter) or
            (board[3]==letter and board[6]==letter and board[9]==letter) or
            (board[1]==letter and board[5]==letter and board[9]==letter) or
            (board[3]==letter and board[5]==letter and board[7]==letter))
```

```
def getBoardCopy(board):
    # Make a duplicate of the board list and return it the duplicate.
    dupBoard = []

    for i in board:
        dupBoard.append(i)

    return dupBoard


def isSpaceFree(board, move):
    return board[move] == ' '


def getPlayerMove(board):
    # Let the player type in their move.
    move = ''
    while move not in '1 2 3 4 5 6 7 8 9'.split() or not isSpaceFree(board,int(move)):
```

```
        print('What is your next move? (1-9)')
        move = input()
    return int(move)


def chooseRandomMoveFromList(board, movesList):
    # Returns a valid move from the passed list on the passed board.
    # Returns None if there is no valid move.
    possibleMoves = []
    for i in movesList:
        if isSpaceFree(board, i):
            possibleMoves.append(i)

    if len(possibleMoves) != 0:
        return random.choice(possibleMoves)
    else:
        return None
```

```
def minimax(board, depth, isMax, alpha, beta, computerLetter):
    # Given a board and the computer's letter, determine where to move and return that move.
    if computerLetter == 'X':
        playerLetter = 'O'
    else:
        playerLetter = 'X'


    if isWinner(board, computerLetter):
        return 10
    if isWinner(board, playerLetter):
        return -10
    if isBoardFull(board):
        return 0


    if isMax:
        best = -1000


        for i in range(1,10):
            if isSpaceFree(board, i):
```

```
            board[i] = computerLetter
            best = max(best, minimax(board, depth+1, not isMax, alpha, beta, computerLetter) - depth)
            alpha = max(alpha, best)
            board[i] = ' '

            if alpha >= beta:
                break

        return best
    else:
        best = 1000

        for i in range(1,10):
            if isSpaceFree(board, i):
                board[i] = playerLetter
                best = min(best, minimax(board, depth+1, not isMax, alpha, beta, computerLetter) + depth)
                beta = min(beta, best)
                board[i] = ' '
```

```
            if alpha >= beta:
                break


    return best



def findBestMove(board, computerLetter):
    # Given a board and the computer's letter, determine where to move and return that move.
    if computerLetter == 'X':
        playerLetter = 'O'
    else:
        playerLetter = 'X'

    bestVal = -1000
    bestMove = -1



    for i in range(1,10):
        if isSpaceFree(board, i):
```

```
            board[i] = computerLetter

            moveVal = minimax(board, 0, False, -1000, 1000,
computerLetter)

            board[i] = ' '

            if moveVal > bestVal:
                bestMove = i
                bestVal = moveVal

    return bestMove



def isBoardFull(board):
    # Return True if every space on the board has been taken.
Otherwise return False.
    for i in range(1,10):
        if isSpaceFree(board, i):
            return False
    return True
```

```
print('\nWelcome to Tic Tac Toe!\n')
print('Reference of numbering on the board')
drawBoard('0 1 2 3 4 5 6 7 8 9'.split())
print('')

start=time.time()

while True:
    # Reset the board
    theBoard = [' '] * 10
    playerLetter, computerLetter = inputPlayerLetter()
    turn = whoGoesFirst()
    print('The ' + turn + ' will go first.')
    gameIsPlaying = True

    while gameIsPlaying:
        if turn == 'player':
            drawBoard(theBoard)
```

```
        move = getPlayerMove(theBoard)
        makeMove(theBoard, playerLetter, move)

        if isWinner(theBoard, playerLetter):
            drawBoard(theBoard)
            print('You won the game')
            gameIsPlaying = False
        else:
            if isBoardFull(theBoard):
                drawBoard(theBoard)
                print('The game is a tie')
                break
            else:
                turn = 'computer'
    else:
        move = findBestMove(theBoard, computerLetter)
        makeMove(theBoard, computerLetter, move)

        if isWinner(theBoard, computerLetter):
            drawBoard(theBoard)
```

```
            print('You lose the game')
            gameIsPlaying = False
        else:
            if isBoardFull(theBoard):
                drawBoard(theBoard)
                print('The game is a tie')
                break
            else:
                turn = 'player'
    if not playAgain():
        end=time.time()
        break

print("Time Taken By Algorithm : ",end-start)
```

Output:

```
Welcome to Tic Tac Toe!

Reference of numbering on the board
1|2|3
-+-+-
4|5|6
-+-+-
7|8|9

Do you want to be 'X' or 'O'?
X
Do you want to go first? (Yes or No)
yes
The player will go first.
 | |
-+-+-
 | |
-+-+-
 | |
What is your next move? (1-9)
1
X| |
-+-+-
 |O|
-+-+-
 | |
```

```
 | |
What is your next move? (1-9)
7
X| |
-+-+-
O|O|
-+-+-
X| |
What is your next move? (1-9)
6
X|O|
-+-+-
O|O|X
-+-+-
X| |
What is your next move? (1-9)
8
X|O|
-+-+-
O|O|X
-+-+-
X|X|O
What is your next move? (1-9)
3
X|O|X
-+-+-
O|O|X
-+-+-
X|X|O
```

```
 | |
What is your next move? (1-9)
8
X|O|
-+-+-
O|O|X
-+-+-
X|X|O
What is your next move? (1-9)
3
X|O|X
-+-+-
O|O|X
-+-+-
X|X|O
The game is a tie
Do you want to play again? (Yes or No)
no
Time Taken By Algorithm :  32.07885670661926
```