PRACTICAL – 4

AIM: Write a python program to create tic-tac-toe game using the minimax algorithm.

Code:

```
import matplotlib.pyplot as plt
from random import choice
import math
import copy
HUMAN = -1
COMP = +1

class StateNode:
    def __init__(self):
        self.board = [[0 for _ in range(3)] for _ in range(3)]

    def getEmptyCells(self):
        return [[x, y] for x in range(3) for y in range(3) if self.board[x][y] == 0]

    def setMove(self, x, y, player):
        if self.isValidMove(x, y):
            self.board[x][y] = player
            return True
        return False

    def isValidMove(self, x, y):
        return self.board[x][y] == 0

    def isWin(self, player):
        b = self.board
        win_states = [
            [b[0][0], b[0][1], b[0][2]],
            [b[1][0], b[1][1], b[1][2]],
            [b[2][0], b[2][1], b[2][2]],
            [b[0][0], b[1][0], b[2][0]],
            [b[0][1], b[1][1], b[2][1]],
            [b[0][2], b[1][2], b[2][2]],
            [b[0][0], b[1][1], b[2][2]],
            [b[0][2], b[1][1], b[2][0]],
        ]
        return [player] * 3 in win_states
```

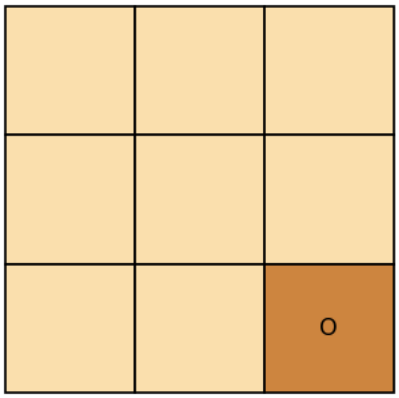
```
    def isGameOver(self):
        return self.isWin(HUMAN) or self.isWin(COMP) or len(self.getEmptyCells()) == 0

    def getScoreValue(self):
        if self.isWin(COMP):
            return 1
        elif self.isWin(HUMAN):
            return -1
        else:
            return 0

    def drawBoard(self, c_choice, h_choice):
        symbols = {HUMAN: h_choice, COMP: c_choice, 0: ' '}
        matrix = [[symbols[self.board[i][j]] for j in range(3)] for i in range(3)]
        plt.figure(figsize=(3, 3))
        tb = plt.table(cellText=matrix, loc='center', cellLoc='center')
        for i in range(3):
            for j in range(3):
                color = "#CD853F" if self.board[i][j] == COMP else "#F9CD81" if self.board[i][j] ==
HUMAN else "#FADFAD"
                tb[(i, j)].set_facecolor(color)
                tb[(i, j)].set_height(1.0 / 3)
                tb[(i, j)].set_width(1.0 / 3)

        ax = plt.gca()
        ax.set_xticks([])
        ax.set_yticks([])
        plt.box(False)
        plt.show()

class TicTacToe:
    def __init__(self):
        self.h_choice = ''
        self.c_choice = ''
        self.first = ''
        self.state = StateNode()

    def playerMove(self):
        moves = {
            1: [0, 0], 2: [0, 1], 3: [0, 2],
            4: [1, 0], 5: [1, 1], 6: [1, 2],
            7: [2, 0], 8: [2, 1], 9: [2, 2]
        }
```

```
        move = -1
        while move not in moves or not self.state.setMove(*moves[move], HUMAN):
            try:
                move = int(input("Enter your move (1-9): "))
            except:
                print("Invalid input!")
        print("You played:")
        self.state.drawBoard(self.c_choice, self.h_choice)

    def computerMove(self):
        depth = len(self.state.getEmptyCells())
        if depth == 0 or self.state.isGameOver():
            return
        print("Computer is thinking...")
        if depth == 9:
            move = choice(self.state.getEmptyCells())
        else:
            _, x, y = self.minimax(copy.deepcopy(self.state), COMP)
            move = [x, y]
        self.state.setMove(*move, COMP)
        print("Computer played:")
        self.state.drawBoard(self.c_choice, self.h_choice)

    def minimax(self, node, player):
        if node.isWin(HUMAN):
            return -1, None, None
        if node.isWin(COMP):
            return 1, None, None
        if len(node.getEmptyCells()) == 0:
            return 0, None, None

        moves = []
        for cell in node.getEmptyCells():
            x, y = cell
            new_node = copy.deepcopy(node)
            new_node.setMove(x, y, player)
            score, _, _ = self.minimax(new_node, -player)
            moves.append((score, x, y))

        if player == COMP:
            return max(moves, key=lambda x: x[0])
        else:
            return min(moves, key=lambda x: x[0])
```

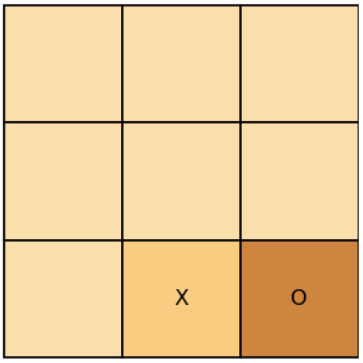
```
    def start_game(self):
        while self.h_choice not in ['X', 'O']:
            self.h_choice = input("Choose X or O: ").upper()
            self.c_choice = 'O' if self.h_choice == 'X' else 'X'
        while self.first not in ['Y', 'N']:
            self.first = input("Do you want to start first? [Y/N]: ").upper()
        while not self.state.isGameOver():
            if self.first == 'Y':
                self.playerMove()
                if self.state.isGameOver():
                    break
                self.computerMove()
            else:
                self.computerMove()
                if self.state.isGameOver():
                    break
                self.playerMove()
        print("Final Result:")
        self.state.drawBoard(self.c_choice, self.h_choice)
        if self.state.isWin(COMP):
            print("Computer wins!")
        elif self.state.isWin(HUMAN):
            print("You win!")
        else:
            print("It's a draw!")

if __name__ == '__main__':
    game = TicTacToe()
    game.start_game()
```

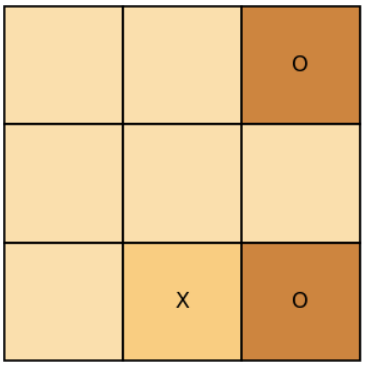
Output:

```
Choose X or O: x
Do you want to start first? [Y/N]: n
Computer is thinking...
Computer played:
```

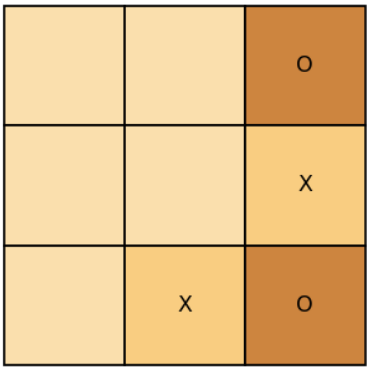


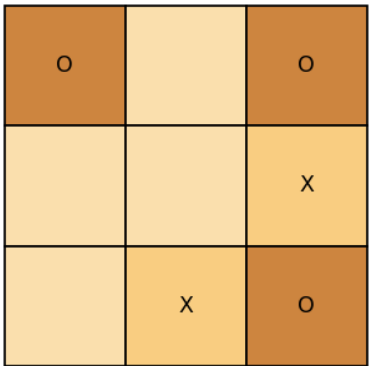
```
Enter your move (1-9): 8
You played:
Computer is thinking...
Computer played:
```



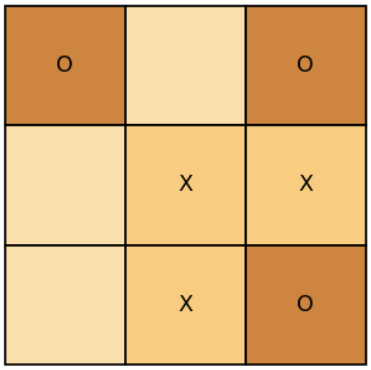


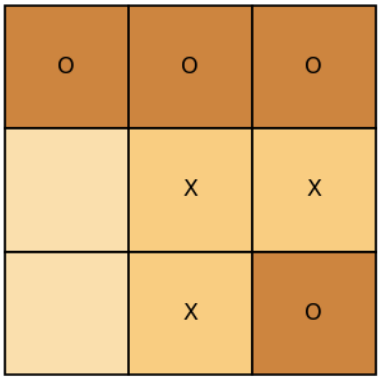

```
Enter your move (1-9): 6
You played:
Computer is thinking...
Computer played:
```






```
Enter your move (1-9): 5
You played:
Computer is thinking...
Computer played:
```






```
Final Result:
Computer wins!
```