AIES_Assignment_2

```
def isMovesLeft(board):
    for i in range(3):
        for j in range(3):patiadkvnaslvnakvsc
            if board[i][j] == '_':
                return True
    return False

def evaluate(b):
    for row in range(3):
        if b[row][0] == b[row][1] == b[row][2]:
            if b[row][0] == player:
                return 10
            elif b[row][0] == opponent:
                return -10
    for col in range(3):
        if b[0][col] == b[1][col] == b[2][col]:
            if b[0][col] == player:
                return 10
            elif b[0][col] == opponent:
                return -10
    if b[0][0] == b[1][1] == b[2][2]:
        if b[0][0] == player:
            return 10
        elif b[0][0] == opponent:
            return -10
    if b[0][2] == b[1][1] == b[2][0]:
        if b[0][2] == player:
            return 10
        elif b[0][2] == opponent:
            return -10
    return 0

def minimax(board, depth, isMax):
    score = evaluate(board)
    if score == 10:
        return score
    if score == -10:
        return score
    if not isMovesLeft(board):
        return 0

    if isMax:
        best = -1000
        for i in range(3):
            for j in range(3):
                if board[i][j] == '_':
                    board[i][j] = player
                    best = max(best, minimax(board, depth + 1, not
```

```
isMax))
                    board[i][j] = '_'
        return best
    else:
        best = 1000
        for i in range(3):
            for j in range(3):
                if board[i][j] == '_':
                    board[i][j] = opponent
                    best = min(best, minimax(board, depth + 1, not
isMax))
                    board[i][j] = '_'
        return best

def findBestMove(board):
    bestVal = -1000
    bestMove = (-1, -1)
    for i in range(3):
        for j in range(3):
            if board[i][j] == '_':
                board[i][j] = player
                moveVal = minimax(board, 0, False)
                board[i][j] = '_'
                if moveVal > bestVal:
                    bestMove = (i, j)
                    bestVal = moveVal
    return bestMove

player, opponent = 'x', 'o'

board = [
    ['x', 'o', 'o'],
    ['x', 'x', 'o'],
    ['_', '_', '_']
]
bestMove = findBestMove(board)
print("The Optimal Move is :")
print("ROW:", bestMove[0], " COL:", bestMove[1])
```

```
The Optimal Move is :
ROW: 2  COL: 0
```