NAME : ANKIT KUMAR SAHU

REG NO. 20BAI1005

LAB 7

MINIMAX ALGORITHM

```
PLAYER, OPPONENT = 'x', 'o'

def any_move_exist(board):
    for i in range(4):
        for j in range(4):
            if (board[i][j] == '_'):
                return True
    return False

def evaluation_function(b):
    for row in range(4):
        if (b[row][0] == b[row][1] and b[row][1] == b[row][2] == b[row][3]):
            if (b[row][0] == PLAYER):
                return 1
            elif (b[row][0] == OPPONENT):
                return -1

    for col in range(3):
        if (b[0][col] == b[1][col] and b[1][col] == b[2][col] == b[3][col]):
            if (b[0][col] == PLAYER):
                return 1
            elif (b[0][col] == OPPONENT):
                return -1

    if (b[0][0] == b[1][1] and b[1][1] == b[2][2] == b[3][3]):
        if (b[0][0] == PLAYER):
            return 1
        elif (b[0][0] == OPPONENT):
            return -1
```

```
def evaluation_function(b):
    for row in range(4):
        if (b[row][0] == b[row][1] and b[row][1] == b[row][2] == b[row][3]):
            if (b[row][0] == PLAYER):
                return 1
            elif (b[row][0] == OPPONENT):
                return -1

    for col in range(3):
        if (b[0][col] == b[1][col] and b[1][col] == b[2][col] == b[3][col]):
            if (b[0][col] == PLAYER):
                return 1
            elif (b[0][col] == OPPONENT):
                return -1

    if (b[0][0] == b[1][1] and b[1][1] == b[2][2] == b[3][3]):
        if (b[0][0] == PLAYER):
            return 1
        elif (b[0][0] == OPPONENT):
            return -1

    if (b[0][3] == b[1][2] and b[2][1] == b[1][2] and b[2][1] == b[3][0]):
        if (b[0][3] == PLAYER):
            return 1
        elif (b[0][3] == OPPONENT):
            return -1
    return 0
```

```
def minimax(board, depth, isMax):
    score = evaluation_function(board)
    if (score == 1):
        return score
    if (score == -1):
        return score
    if (any_move_exist(board) == False):
        return 0
    if (isMax):
        best = -100
        for i in range(4):
            for j in range(4):
                if (board[i][j]=='_'):
                    board[i][j] = PLAYER
                    best = max( best, minimax(board,depth + 1,not isMax))
                    board[i][j] = '_'
        return best
    else:
        best = 100
        for i in range(4):
            for j in range(4):
                if (board[i][j] == '_'):
                    board[i][j] = OPPONENT
                    best = min(best, minimax(board, depth + 1, not isMax))
                    board[i][j] = '_'
        return best
```

```
def best_move(board):
    bestVal = -100
    bestMove = (-1, -1)
    for i in range(4):
        for j in range(4):
            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
```

```
board = [
    [ 'x', 'o', 'x','x' ],
    [ 'o', 'o', 'x','o' ],
    [ '_', '_', '_','x' ],
    [ '_', '_', '_','x' ],
]

move = best_move(board)

print("Next best move is")
print("ROW:", move[0], " COL:", move[1])
```

```
Next best move is
ROW: 2  COL: 1
```