**Practical: 14**

**Aim: Implement the Tic-Tac-Toe game using Minimax algorithm for adversarial search.**

**(Note: The first move is to be made by humans. The next move should be made by computer and so on.)**

**Code :**

import math

PLAYER\_X = 'X'

PLAYER\_O = 'O'

EMPTY = ' '

def print\_board(board):

for row in board:

print('|'.join(row))

print('-' \* 5)

def check\_winner(board):

lines = [\*board,

\*zip(\*board),

[board[i][i] for i in range(3)],

[board[i][2-i] for i in range(3)]]

for line in lines:

if line[0] == line[1] == line[2] != EMPTY:

return line[0]

return None

def is\_full(board):

return all(cell != EMPTY for row in board for cell in row)

def minimax(board, depth, is\_maximizing):

winner = check\_winner(board)

if winner == PLAYER\_O:

return 10 - depth

elif winner == PLAYER\_X:

return depth - 10

elif is\_full(board):

return 0

if is\_maximizing:

best\_score = -math.inf

for i in range(3):

for j in range(3):

if board[i][j] == EMPTY:

board[i][j] = PLAYER\_O

score = minimax(board, depth + 1, False)

board[i][j] = EMPTY

best\_score = max(score, best\_score)

return best\_score

else:

best\_score = math.inf

for i in range(3):

for j in range(3):

if board[i][j] == EMPTY:

board[i][j] = PLAYER\_X

score = minimax(board, depth + 1, True)

board[i][j] = EMPTY

best\_score = min(score, best\_score)

return best\_score

def find\_best\_move(board):

best\_score = -math.inf

best\_move = (-1, -1)

for i in range(3):

for j in range(3):

if board[i][j] == EMPTY:

board[i][j] = PLAYER\_O

score = minimax(board, 0, False)

board[i][j] = EMPTY

if score > best\_score:

best\_score = score

best\_move = (i, j)

return best\_move

def play\_game():

board = [[EMPTY for \_ in range(3)] for \_ in range(3)]

while True:

print\_board(board)

try:

x, y = map(int, input("Enter your move (row and column): ").split())

if board[x][y] != EMPTY:

print("Invalid move! Try again.")

continue

board[x][y] = PLAYER\_X

print(f"You placed X at ({x}, {y})")

except (ValueError, IndexError):

print("Invalid input! Please enter row and column as two numbers (0, 1, or 2).")

continue

if check\_winner(board):

print\_board(board)

print("You win!")

break

if is\_full(board):

print\_board(board)

print("It's a draw!")

break

print("Computer's turn...")

move = find\_best\_move(board)

board[move[0]][move[1]] = PLAYER\_O

print(f"Computer placed O at ({move[0]}, {move[1]})")

if check\_winner(board):

print\_board(board)

print("Computer wins!")

break

if is\_full(board):

print\_board(board)

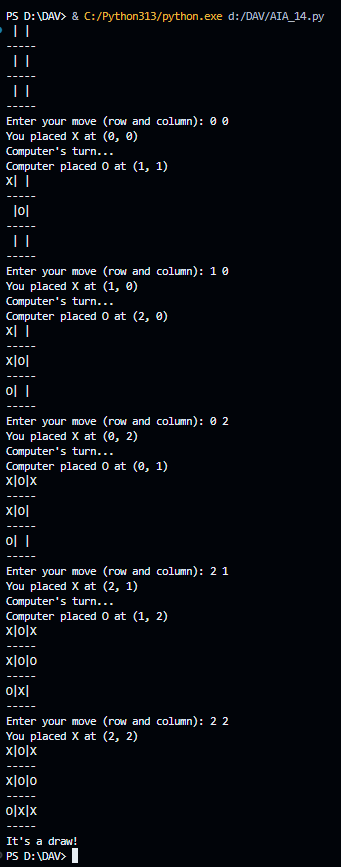
print("It's a draw!")

break

if \_\_name\_\_ == "\_\_main\_\_":

play\_game()

* **Output :**

****