**PRACTICAL 12**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name:** | Harsh Shah | **Semester:** | VI | **Division:** | 6 |
| **Roll No.:** | 21BCP359 | **Date:** |  | **Batch:** | G11 |
| **Aim:** | WAP to design Tic Tac Toe games from O (Opponent) and X (Player) by using minimax algorithm. | | | | |

**Program**

import sys

# Function to print the Tic Tac Toe board

def print\_board(board):

    for row in board:

        print(" | ".join(row))

        print(" - " \* 3)

# Function to check if the board is full

def is\_board\_full(board):

    for row in board:

        for cell in row:

            if cell == " ":

                return False

    return True

# Function to check if a player has won

def check\_winner(board, player):

    # Check rows

    for row in board:

        if all(cell == player for cell in row):

            return True

    # Check columns

    for col in range(3):

        if all(board[row][col] == player for row in range(3)):

            return True

    # Check diagonals

    if all(board[i][i] == player for i in range(3)) or all(

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

    ):

        return True

    return False

# Function to evaluate the current state of the board

def evaluate(board):

    if check\_winner(board, "O"):

        return 1

    elif check\_winner(board, "X"):

        return -1

    elif is\_board\_full(board):

        return 0

    else:

        return None

# Minimax algorithm implementation

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

    score = evaluate(board)

    if score is not None:

        return score

    if is\_maximizing:

        best\_score = -sys.maxsize

        for i in range(3):

            for j in range(3):

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

                    board[i][j] = "O"

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

                    board[i][j] = " "

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

        return best\_score

    else:

        best\_score = sys.maxsize

        for i in range(3):

            for j in range(3):

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

                    board[i][j] = "X"

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

                    board[i][j] = " "

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

        return best\_score

# Function to find the best move for the opponent using Minimax

def find\_best\_move(board):

    best\_score = -sys.maxsize

    best\_move = None

    for i in range(3):

        for j in range(3):

            if board[i][j] == " ":

                board[i][j] = "O"

                score = minimax(board, 0, False)

                board[i][j] = " "

                if score > best\_score:

                    best\_score = score

                    best\_move = (i, j)

    return best\_move

# Main function to play the game

def play\_game():

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

    print("Welcome to Tic Tac Toe!")

    print\_board(board)

    while True:

        # Player's move

        row, col = map(int, input("Enter your move (row col): ").split())

        if board[row][col] != " ":

            print("Invalid move. Try again.")

            continue

        board[row][col] = "X"

        print\_board(board)

        if check\_winner(board, "X"):

            print("Congratulations! You win!")

            break

        if is\_board\_full(board):

            print("It's a draw!")

            break

        # Opponent's move

        print("Opponent is thinking...")

        opponent\_row, opponent\_col = find\_best\_move(board)

        board[opponent\_row][opponent\_col] = "O"

        print\_board(board)

        if check\_winner(board, "O"):

            print("Sorry, you lose!")

            break

        if is\_board\_full(board):

            print("It's a draw!")

            break

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

    play\_game()

**Output**

  