**Tic-Tac-Toe with Minimax AI**

**Input:**

# Board setup

board = [' '] \* 10  # index 0 is ignored

player = 1  # Player 1 (AI) starts

Win = 1

Draw = -1

Running = 0

Game = Running

def draw\_board():

    #Display the current board.

    print(f" {board[1]} | {board[2]} | {board[3]} ")

    print("---|---|---")

    print(f" {board[4]} | {board[5]} | {board[6]} ")

    print("---|---|---")

    print(f" {board[7]} | {board[8]} | {board[9]} ")

    print("   |   |   \n")

def check\_position(x):

    #Check if the position is free.

    return board[x] == ' '

def check\_win():

    #Check if the game is won or drawn.

    global Game

    # Horizontal wins

    if (board[1] == board[2] == board[3] != ' ' or

        board[4] == board[5] == board[6] != ' ' or

        board[7] == board[8] == board[9] != ' '):

        Game = Win

    # Vertical wins

    elif (board[1] == board[4] == board[7] != ' ' or

          board[2] == board[5] == board[8] != ' ' or

          board[3] == board[6] == board[9] != ' '):

        Game = Win

    # Diagonal wins

    elif (board[1] == board[5] == board[9] != ' ' or

          board[3] == board[5] == board[7] != ' '):

        Game = Win

    # Draw condition

    elif all(space != ' ' for space in board[1:]):

        Game = Draw

    else:

        Game = Running

def evaluate(board):

    # Evaluate board and return score.

    # Horizontal

    if board[1] == board[2] == board[3]:

        return 10 if board[1] == 'X' else -10 if board[1] == 'O' else 0

    if board[4] == board[5] == board[6]:

        return 10 if board[4] == 'X' else -10 if board[1] == 'O' else 0

    if board[7] == board[8] == board[9]:

        return 10 if board[1] == 'X' else -10 if board[1] == 'O' else 0

    # Vertical

    if board[1] == board[4] == board[7]:

        return 10 if board[1] == 'X' else -10 if board[1] == 'O' else 0

    if board[2] == board[5] == board[8]:

        return 10 if board[2] == 'X' else -10 if board[2] == 'O' else 0

    if board[3] == board[6] == board[9]:

        return 10 if board[3] == 'X' else -10 if board[3] == 'O' else 0

    # Diagonal

    if board[1] == board[5] == board[9]:

        return 10 if board[1] == 'X' else -10 if board[1] == 'O' else 0

    if board[3] == board[5] == board[7]:

        return 10 if board[3] == 'X' else -10 if board[3] == 'O' else 0

    return 0

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

    #Recursive function to find the best move by predicting future moves.

    score = evaluate(board)

    if score == 10 or score == -10:

        return score

    if all(space != ' ' for space in board[1:]):

        return 0  # Draw

    #isMaximizing:

    #True → AI's turn (maximize score)

    #False → Opponent's turn (minimize score)

    if is\_maximizing:

        best = -1000

        for i in range(1, 10):

            if board[i] == ' ':

                board[i] = 'X'

                best = max(best, minimax(board, depth + 1, False))

                board[i] = ' ' #Undo the move so that board stays clean for other possibilities.

        return best

    else:

        best = 1000

        for i in range(1, 10):

            if board[i] == ' ':

                board[i] = 'O'

                best = min(best, minimax(board, depth + 1, True))

                board[i] = ' '

        return best

def find\_best\_move(board):

    #Find the best move for AI ('X').

    best\_val = -1000

    best\_move = -1

    for i in range(1, 10):

        if board[i] == ' ':

            board[i] = 'X'

            move\_val = minimax(board, 0, False)

            board[i] = ' '

            if move\_val > best\_val:

                best\_move = i

                best\_val = move\_val

    return best\_move

# --- Main Game ---

print("Welcome to Tic-Tac-Toe!")

print("Player 1 [X] (AI) vs Player 2 [O] (Human)\n")

print("Please Wait...")

while Game == Running:

    draw\_board()

    if player % 2 != 0:

        # AI's turn

        print("Player 1's (AI) turn:")

        mark = 'X'

        choice = find\_best\_move(board)

    else:

        # Human's turn

        print("Player 2's (Human) turn:")

        mark = 'O'

        choice = int(input("Enter the position (1-9) where you want to move: "))

    if check\_position(choice):

        board[choice] = mark

        player += 1

        check\_win()

    else:

        print("Invalid move! Try again.")

draw\_board()

if Game == Draw:

    print("Game Draw!")

elif Game == Win:

    player -= 1

    if player % 2 != 0:

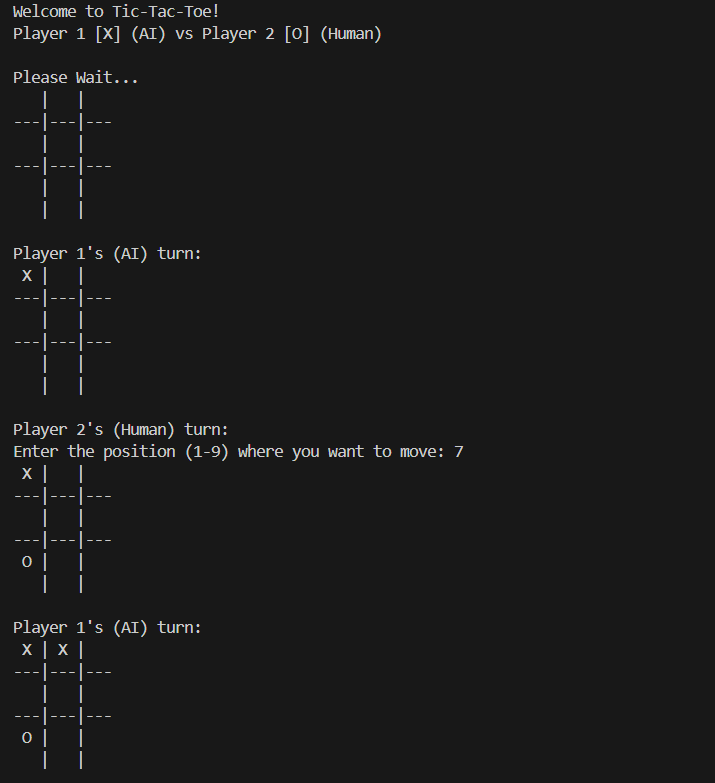
        print("Player 1 (AI) Wins!")

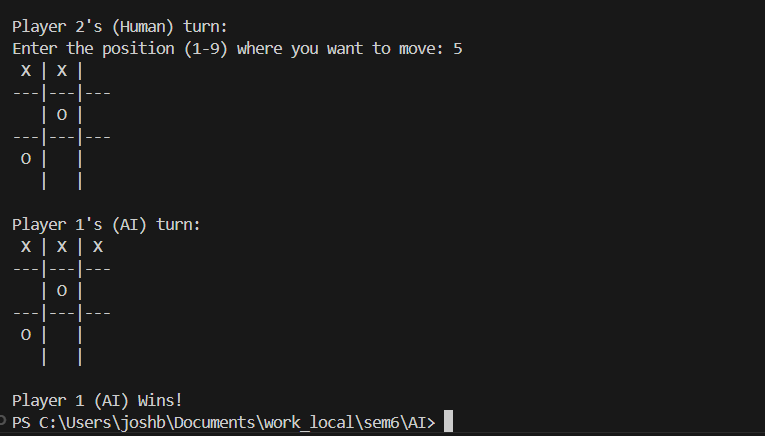
    else:

        print("Player 2 (Human) Wins!")

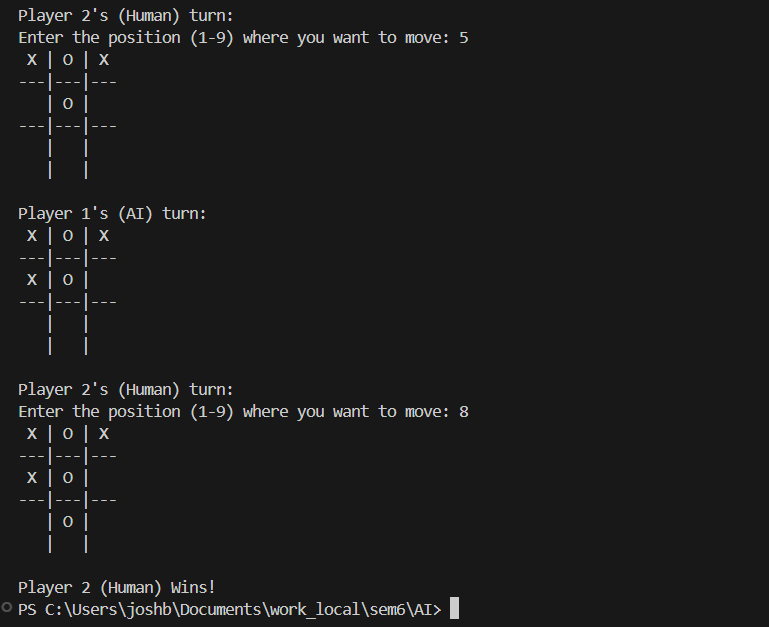
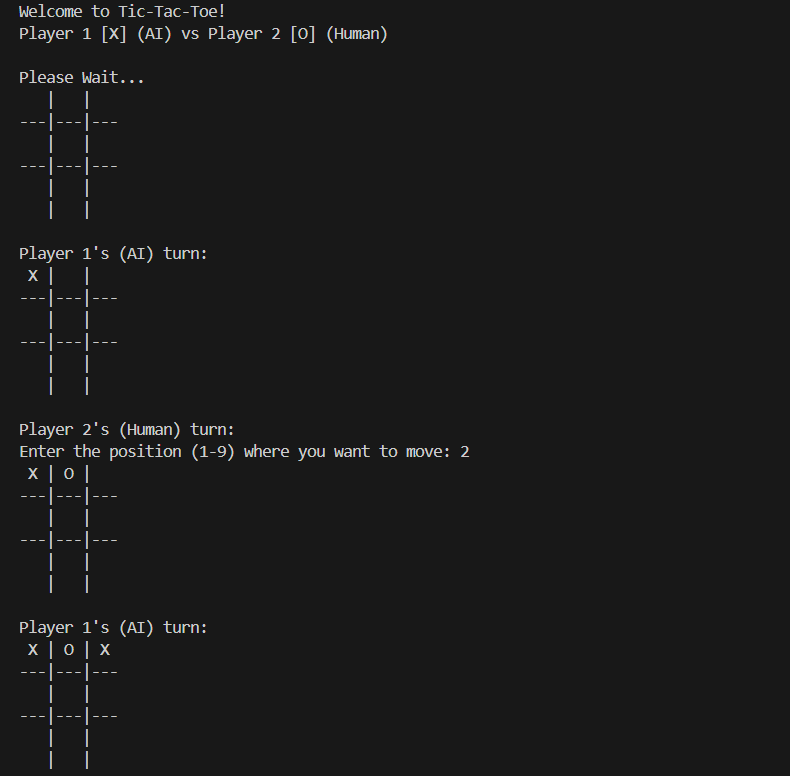
**Output:**

1. **Player 1 (AI) wins:**

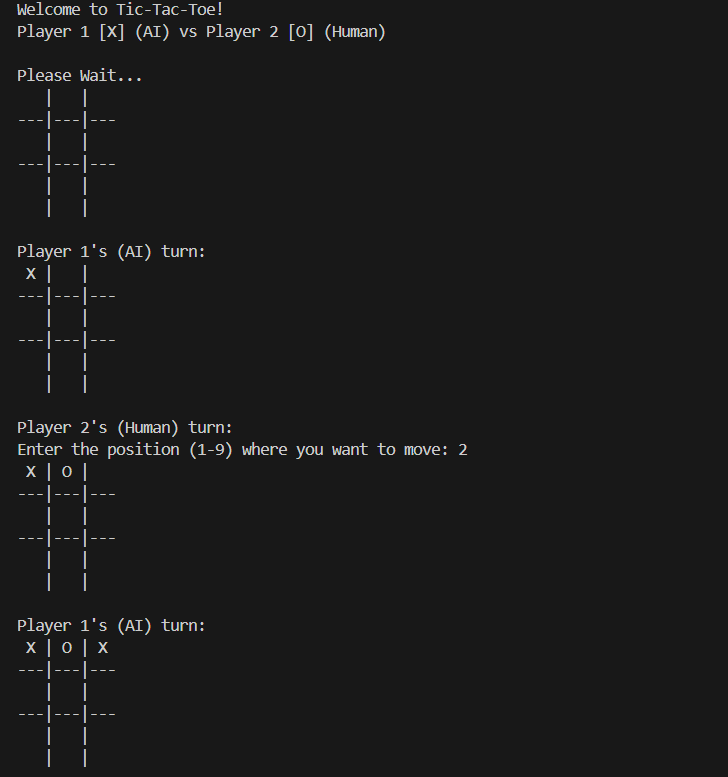
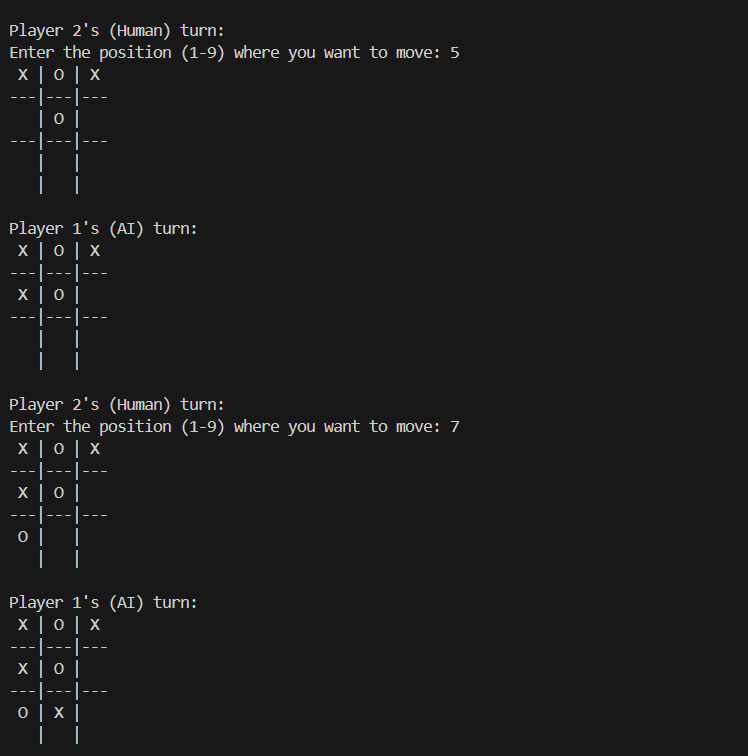
****

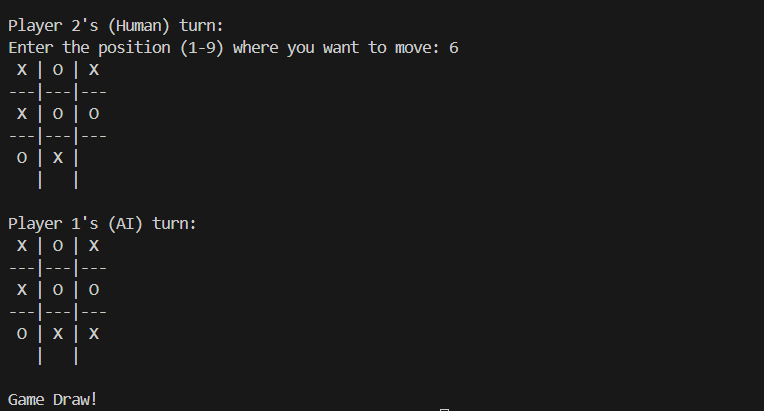
****

1. **Player 2 (human) wins:**

****

1. **Game draw:**





**Conclusion**: Tic-Tac-Toe with Minimax AI was studied and implemented successfully