Date 04/03/2024

7.Aim: Write the python program to implement Minimax algorithm for gaming

Program:

```
import math

# Function to print the current state of the board
def print_board(board):
    for row in board:
        print(" ".join(row))

# Function to check if the current player has won
def is_winner(board, player):
    # Check rows, columns, and diagonals for a win
    for i in range(3):
        if all(cell == player for cell in board[i]) or all(board[j][i] == player for j in range(3)):
            return True
    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 check if the board is full
def is_board_full(board):
    return all(cell != '-' for row in board for cell in row)

# Function to evaluate the current state of the board
def evaluate(board):
    if is_winner(board, 'X'):
        return 10
    elif is_winner(board, 'O'):
```

```
        return -10
    else:
        return 0

# Minimax function to recursively search for the best move
def minimax(board, depth, is_maximizing):
    score = evaluate(board)

    # Base cases
    if score == 10:
        return score - depth
    elif score == -10:
        return score + depth
    elif is_board_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] == '-':
                    board[i][j] = 'X'
                    score = minimax(board, depth + 1, False)
                    board[i][j] = '-'
                    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] == '-':
                    board[i][j] = 'O'
                    score = minimax(board, depth + 1, True)
                    board[i][j] = '-'
                    best_score = min(score, best_score)
        return best_score

# Function to find and make the best move
def make_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] == '-':
                board[i][j] = 'X'
                score = minimax(board, 0, False)
                board[i][j] = '-'
                if score > best_score:
                    best_score = score
                    best_move = (i, j)
    board[best_move[0]][best_move[1]] = 'X'

# Main function to run the game
def main():
    board = [['-' for _ in range(3)] for _ in range(3)]
    print("Welcome to Tic Tac Toe! You are O and the computer is X.")
    print("Enter your moves by specifying the row and column (e.g., '1 2')")
    print_board(board)
```

```
    while True:
        row, col = map(int, input("Enter your move: ").split())
        if board[row-1][col-1] == '-':
            board[row-1][col-1] = 'O'
            print_board(board)
            if is_winner(board, 'O'):
                pri nt("Congratulations! You win!")
                break
            if is_board_full(board):
                print("It's a tie!")
                break
            make_best_move(board)
            print("Computer's move:")
            print_board(board)
            if is_winner(board, 'X'):
                print("Computer wins! Better luck next time.")
                break
                if is_board_full(board):
                print("It's a tie!")
                break
        else:
            print("Invalid move! Try again.")


if __name__ == "__main__":
    main()
```

Output:

Result: The given program has been executed successfully