EXPERIMENT 9

Krishna Thakkar C075 60004220250

Code:

```
import math

# Initialize the board
board = [' ' for _ in range(9)]

# Function to print the Tic-Tac-Toe board
def print_board():
    for row in [board[i * 3:(i + 1) * 3] for i in range(3)]:
        print('| ' + ' | '.join(row) + ' |')

# Function to check for a winner
def check_winner(player):
    # Check rows, columns, and diagonals
    win_conditions = [(0, 1, 2), (3, 4, 5), (6, 7, 8),  # Rows
                      (0, 3, 6), (1, 4, 7), (2, 5, 8),  # Columns
                      (0, 4, 8), (2, 4, 6)]             # Diagonals
    for condition in win_conditions:
        if board[condition[0]] == board[condition[1]] == board[condition[2]]
== player:
            return True
    return False

# Function to check if the board is full (i.e., a tie)
def is_board_full():
    return ' ' not in board

# Function to get available spots on the board
def available_moves():
    return [i for i in range(9) if board[i] == ' ']

# Minimax algorithm to determine the best move for the AI
def minimax(is_maximizing):
    if check_winner('O'):  # AI is 'O'
        return 1  # AI wins
    elif check_winner('X'):  # Human is 'X'
        return -1  # Human wins
    elif is_board_full():
        return 0  # Tie

    if is_maximizing:
        best_score = -math.inf
        for move in available_moves():
            board[move] = 'O'
            score = minimax(False)
            board[move] = ' '
            best_score = max(score, best_score)
        return best_score
```

EXPERIMENT 9

Krishna Thakkar C075 60004220250

```
    else:
        best_score = math.inf
        for move in available_moves():
            board[move] = 'X'
            score = minimax(True)
            board[move] = ' '
            best_score = min(score, best_score)
        return best_score

# Function for the AI to choose the best move
def ai_move():
    best_score = -math.inf
    best_move = None
    for move in available_moves():
        board[move] = 'O'
        score = minimax(False)
        board[move] = ' '
        if score > best_score:
            best_score = score
            best_move = move
    board[best_move] = 'O'

# Function to handle the player's move
def player_move():
    move = int(input("Enter your move (1-9): ")) - 1
    if board[move] == ' ':
        board[move] = 'X'
    else:
        print("Invalid move! Try again.")
        player_move()

# Function to check if the game is over
def is_game_over():
    return check_winner('X') or check_winner('O') or is_board_full()

# Main game loop
def play_game():
    print("Welcome to Tic-Tac-Toe!")
    print_board()

    while not is_game_over():
        # Player move
        player_move()
        print_board()

        if is_game_over():
            break
```

EXPERIMENT 9

Krishna Thakkar C075 60004220250

```
        # AI move
        print("AI is making a move...")
        ai_move()
        print_board()

    if check_winner('X'):
        print("Congratulations! You won!")
    elif check_winner('O'):
        print("AI wins! Better luck next time.")
    else:
        print("It's a tie!")

# Start the game
if __name__ == "__main__":
    play_game()
```

Output:

```
Welcome to Tic-Tac-Toe!
|   |   |   |
|   |   |   |
|   |   |   |
Enter your move (1-9): 3
|   |   | X |
|   |   |   |
|   |   |   |
AI is making a move...
|   |   | X |
|   | O |   |
|   |   |   |
Enter your move (1-9): 6
|   |   | X |
|   | O | X |
|   |   |   |
AI is making a move...
|   |   | X |
|   | O | X |
|   |   | O |
Enter your move (1-9): 2
|   | X | X |
|   | O | X |
|   |   | O |
AI is making a move...
| O | X | X |
|   | O | X |
|   |   | O |
AI wins! Better luck next time.
```