# Assignment 5: Implement Min-Max Algorithm for Game Playing

## Theory

The Min-Max algorithm is a fundamental technique used in Artificial Intelligence (AI) for decision-making and game-playing. It is primarily applied in two-player games such as Tic-Tac-Toe, Chess, or Checkers, where one player tries to maximize their chances of winning, while the other tries to minimize the opponent’s chances.  
  
The Min-Max algorithm simulates all possible future moves by both players and evaluates the resulting board states using a heuristic function. The Maximizer aims to achieve the maximum score, while the Minimizer aims to achieve the minimum score. By recursively exploring all possibilities, the algorithm selects the optimal move assuming both players play perfectly.  
  
This algorithm represents adversarial search, which is a branch of game theory in AI. It ensures that the agent (computer) makes rational and optimal decisions in competitive environments.

## Definition

The Min-Max Algorithm is a recursive decision-making process used in game theory to minimize the possible loss in a worst-case scenario.  
  
It consists of:  
• Game States: Representation of the game board at any point.  
• Utility Function (Evaluation): Assigns a numerical value to each terminal state (win, lose, draw).  
• Players:  
 - Maximizer (Player X): Tries to maximize the score.  
 - Minimizer (Player O): Tries to minimize the score.  
• Recursion: The algorithm explores all possible future moves and selects the one that leads to the best guaranteed outcome.

## Explanation of Implementation

The system implements the Min-Max algorithm for the Tic-Tac-Toe game in C++, allowing a human player (X) to compete against a computer (O).

### Step 1: Game Representation

The Tic-Tac-Toe board is represented using a 3×3 2D vector. Each cell contains:  
• 'X' → Human Player (Maximizer)  
• 'O' → Computer Player (Minimizer)  
• '.' → Empty Cell

*const char PLAYER\_X = 'X';  
const char PLAYER\_O = 'O';  
const char EMPTY = '.';  
const int BOARD\_SIZE = 3;*

### Step 2: Board Evaluation Function

This function checks if a player has won or if the game is a draw.  
• Returns +10 if X wins.  
• Returns −10 if O wins.  
• Returns 0 for a draw or ongoing game.  
It checks all rows, columns, and diagonals for winning conditions.

### Step 3: Game Over Condition

This function checks whether the game has ended due to a win or a full board (draw).

### Step 4: Min-Max Algorithm

The core recursive function explores all possible moves and applies the Min-Max logic.  
  
• Maximizing phase: The AI assumes player X tries to maximize the score.  
• Minimizing phase: The AI assumes player O tries to minimize the score.  
• Uses backtracking to undo moves after simulation.

Pseudo Flow:  
*If terminal state (win/lose/draw): return score  
If maximizing:  
 best = -∞  
 for each empty cell:  
 make move  
 best = max(best, minimax(..., false))  
 undo move  
 return best  
If minimizing:  
 best = +∞  
 for each empty cell:  
 make move  
 best = min(best, minimax(..., true))  
 undo move  
 return best*

### Step 5: Finding the Best Move

The computer (Player O) uses this function to determine its optimal move by calling Min-Max on each possible move and choosing the one with the minimum score.

### Step 6: Game Play Loop

The main function runs an interactive console-based game where:  
1. The human (X) enters their move (row, column).  
2. The computer (O) calculates its move using Min-Max.  
3. The board is printed after every move.  
4. The program checks for win, lose, or draw.

### Step 7: Output and Inference

Example Gameplay Output:  
  
*Tic-Tac-Toe: Player X (you) vs. Player O (computer)  
  
. . .  
. . .  
. . .  
  
Enter your move row (0, 1, or 2): 0  
Enter your move column (0, 1, or 2): 0  
Computer's turn...  
X . .  
. O .  
. . .  
  
You win! (or) Computer wins! (or) It’s a draw!*

## Conclusion

The implementation of the Min-Max algorithm successfully demonstrates the concept of AI game-playing using adversarial search. By simulating all possible moves and choosing the optimal one, the computer makes intelligent and rational decisions in the Tic-Tac-Toe game.  
  
This experiment highlights how rule-based logic, recursive reasoning, and heuristic evaluation form the core of decision-making in Artificial Intelligence. It establishes a foundation for understanding AI planning, game strategy, and search-based problem solving.