**Artificial Intelligent (Lab)**

**Task # 05**

**A purple circle with dots and lines

AI-generated content may be incorrect.**

**Submitted To: Sir Rasikh Ali**

**Submitted By: Shumaila Maryam**

**Roll no: SU92-BSDSM-F24-062**

**Submitted On: 25-September-25**

**Section: BSDS-3A**

**Department of Software Engineering, Superior University, Lahore**

**Question # 01:**

**Min Max Algorithim**

**Introduction:**The Minimax Algorithm Program is a Python tool designed to simulate decision-making in a two-player, turn-based game scenario. It determines the optimal value achievable by a player (MAX or MIN) assuming that both play optimally. The algorithm explores all possible moves to a certain depth, evaluating the leaf node values, and then propagates the optimal choice back up the tree using recursive logic. The program allows users to input leaf node values, select the starting player (MAX or MIN), and view the decision-making process step-by-step in the console.

**Why I Made This Program:**The main purpose behind developing this program was to understand and practice the Minimax algorithm, a fundamental concept in artificial intelligence and game theory. This project helps to:

* Understand recursive algorithms and decision trees.
* Learn how computer players make strategic decisions in games like Tic-Tac-Toe, Chess, or Checkers.
* Gain experience in Python programming with functions, recursion, user input, and logical conditions.
* Observe how the algorithm evaluates each node and chooses the optimal move using the maximization and minimization principles.
* Visualize the backtracking process where leaf values determine the final decision at the root.

This program serves both as an educational exercise and as a foundational demonstration of how AI decision-making works in competitive game environments.

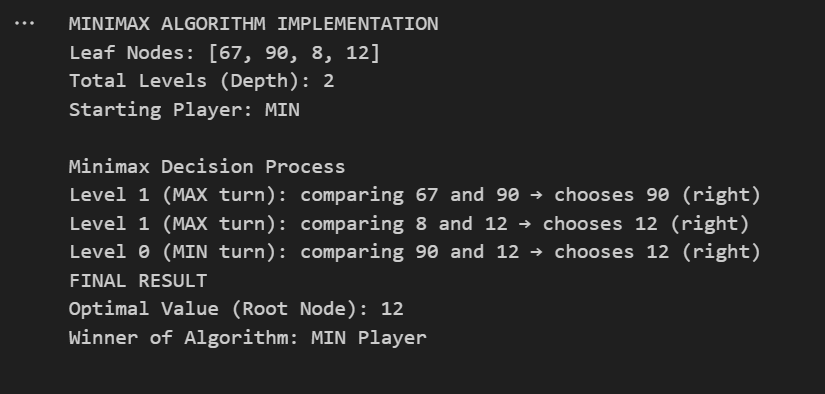
**How It Works:**

1. Starting with Input Data
   * The user enters the number of leaf nodes
   * The program then asks for the value of each leaf node, representing possible outcomes of the game.
   * The user specifies who starts first either the MAX player (who tries to maximize the score) or the MIN player (who tries to minimize it).
2. **Tree Construction Concept**
   * The algorithm assumes a complete binary game tree structure, where:
     + Each non-leaf node has two children.
     + Each leaf node represents a final score.
   * The total depth of the tree is calculated using log₂(n) where *n* is the number of leaf nodes.
3. **Recursive Minimax Evaluation**
   * The function minimaxalgo() recursively evaluates each level:
     + If the current depth equals the total depth, it returns the leaf node value.
     + If it’s the MAX player’s turn, it picks the maximum of the two child values.
     + If it’s the MIN player’s turn, it picks the minimum of the two child values.
   * At each comparison, the program prints a message like:
   * Level 1 (MAX turn): comparing 23 and 12 → chooses 23 (left)

This provides a detailed trace of the decision-making process**.**

1. **Displaying the Decision Process**
   * After processing all nodes, the program prints:
     + The leaf nodes and total levels (depth).
     + The chosen starting player (MAX or MIN).
     + Step-by-step comparisons for each recursive call.
     + The final optimal value at the root node.
2. **Result Interpretation**
   * The final output displays:
     + Optimal Value (Root Node): X
     + Winner of Algorithm: MAX Player / MIN Player
   * This result indicates which player secured the optimal value based on perfect play from both sides.

**Summary:**

1. The Minimax Algorithm Program is a Python project that simulates decision-making in two-player games, determining the optimal move sequence using recursion.
2. It allows dynamic user input for leaf node values, starting player choice, and provides a detailed trace of the minimax process.
3. This project was created to practice algorithmic thinking, recursion, and Python programming concepts.
4. ****It clearly demonstrates how AI evaluates all possible outcomes before making a decision, ensuring the best possible move under competitive conditions.