**LAB 8: MIN\_MAX\_GAME**

This code implements the **Minimax Algorithm**, a fundamental technique in **game theory** and **AI-based decision-making**, particularly for two-player turn-based games like **chess** and **tic-tac-toe**. The function minMaxAlgo is a recursive implementation that traverses a **binary game tree** to determine the optimal move for the **Maximizing** and **Minimizing** players. It takes parameters such as the current depth (currNode), node index (nodeindex), turn (maxturn), leaf node values (leafnodes), and total depth (nodedepth). The base case occurs when the function reaches the leaf node level, where it returns the respective value. If it's the Maximizer's turn (maxturn == True), it chooses the **maximum** value from its child nodes; otherwise, the Minimizer selects the **minimum** value. The recursive structure ensures that the function evaluates all possible moves before returning the optimal decision.

The tree depth is determined using log₂(len(leafnodes)), ensuring a **balanced binary decision tree**. The algorithm starts at the root node with the **Maximizer’s turn**, and the recursion propagates down to the leaf nodes while alternating turns. The tree follows a **bottom-up approach**, where decisions made at deeper levels influence the choices at higher levels. Given the input [2,4,6,4,2,4,6,4], the algorithm constructs a decision tree and determines the best possible move for the Maximizer. The result, in this case, is 4, indicating the **optimal move** based on the Minimax strategy. This approach is crucial in AI for strategic game-playing and can be further optimized using **Alpha-Beta Pruning** to improve efficiency by eliminating unnecessary computations.