Assignment 2 - Adversarial Search with Alpha-Beta Pruning

Problem Statement:
You are tasked with implementing the Minimax algorithm with Alpha-Beta Pruning for a two-player game. The game involves a simple game tree where:

- Player A (MAX) tries to maximize the score.
- Player B (MIN) tries to minimize the score.

The game tree is provided as input in a specific format, and your algorithm should compute the optimal value for Player A using Alpha-Beta Pruning. Additionally, it should output the nodes pruned during the process.

Requirements:
1. Implement the Minimax algorithm with Alpha-Beta Pruning.
2. Ensure that your algorithm:
    - Minimizes the number of nodes evaluated.
    - Correctly tracks and reports pruned nodes.
3. Test the implementation using the sample input provided.

Input Format:
- A tree represented as an adjacency list. Each node specifies its children and value (if it's a leaf).
- Input will consist of:
    - A root node ID.
    - A dictionary representing the tree structure.

Output Format:
- The optimal value for Player A (MAX).
- A list of pruned nodes (if any).

Sample Input	Sample Output

root = "A"
tree = {
"A": ["B", "C"],
"B": ["D", "E"],
"C": ["F", "G"],
"D": [5],
"E": [6],
"F": [3],
"G": [9]
}
	Optimal Value: 5
	Pruned Nodes: ["G"]

Submission Guidelines:

- Deadline: 14 September, Sunday 02:00 pm
- There will be a viva on the assignment.

Pseudocode:

```
function minimax(node, depth, isMaximizingPlayer, alpha, beta):

    if node is a leaf node :
        return value of the node

    if isMaximizingPlayer :
        bestVal = -INFINITY
        for each child node :
            value = minimax(node, depth+1, false, alpha, beta)
            bestVal = max( bestVal, value)
            alpha = max( alpha, bestVal)
            if beta <= alpha:
                break
        return bestVal

    else :
        bestVal = +INFINITY
        for each child node :
            value = minimax(node, depth+1, true, alpha, beta)
            bestVal = min( bestVal, value)
            beta = min( beta, bestVal)
            if beta <= alpha:
                break
        return bestVal
```