Practical 9

```
def minimax(node, depth, player, game_tree, values, path):
    if depth == 0 or node not in game_tree:
        return values[node], [node]
    if player == "MAX":
        best = float(-100000)
        move = []
        for child in game_tree[node]:
            val, child_path = minimax(child, depth - 1, "MIN", game_tree, values, path)
            if val > best:
                best = val
                move = [node] + child_path
        return best, move
    else:  # MIN player
        best = float(100000)
        move = []
        for child in game_tree[node]:
            val, child_path = minimax(child, depth - 1, "MAX", game_tree, values, path)
            if val < best:
                best = val
                move = [node] + child_path
        return best, move
if __name__ == "__main__":
    game_tree = {
        'A': ['B', 'C'],
        'B': ['D', 'E'],

        'C': ['F', 'G']

}

 values = {

        'D': 2,

        'E': 5,

        'F': 1,

        'G': 9

    }

    result, path = minimax('A', 2, "MAX", game_tree, values, [])

    print("Optimal value of the tree is:", result)

    print("Decision path followed:", " -> ".join(path))
```