PRACTICAL10.2

```
def alpha_beta(node, depth, alpha, beta, maximizingPlayer, game_tree, values):
    if depth == 0 or node not in game_tree:
        return values[node]

    if maximizingPlayer:
        value = float("-inf")
        for child in game_tree[node]:
            value = max(value, alpha_beta(child, depth - 1, alpha, beta, False, game_tree, values))
            alpha = max(alpha, value)
            if beta <= alpha:
                break
        return value
    else:
        value = float("inf")
        for child in game_tree[node]:
            value = min(value, alpha_beta(child, depth - 1, alpha, beta, True, game_tree, values))
            beta = min(beta, value)
            if beta <= alpha:
                break
        return value


if __name__ == "__main__":
    game_tree = {
        'A': ['B', 'C', 'D'],
        'B': ['M', 'E', 'F'],
        'C': ['H', 'G', 'I'],
        'D': ['J', 'K', 'L']
    }
    values = {
        'M': 3,
        'E': 12,
        'F': 8,
        'G': 2,
        'H': 4,
        'I': 6,
        'J': 14,
        'K': 5,
        'L': 2
    }

    best_value = alpha_beta('A', 3, float("-inf"), float("inf"), True, game_tree, values)
    print("Best achievable value for root A:", best_value)
```