```python
import math

# Alpha-Beta Pruning implementation
def alpha_beta(depth, node_index, maximizing_player, values, alpha, beta):
    # Example tree depth = 3
    if depth == 3:
        return values[node_index]

    if maximizing_player:
        best = -math.inf

        # Left and right child of current node
        for i in range(2):
            val = alpha_beta(depth + 1, node_index * 2 + i, False, values, alpha, beta)
            best = max(best, val)
            alpha = max(alpha, best)

            # Alpha-Beta Pruning condition
            if beta <= alpha:
                break
        return best

    else:
        best = math.inf

        # Left and right child of current node
        for i in range(2):
            val = alpha_beta(depth + 1, node_index * 2 + i, True, values, alpha, beta)
            best = min(best, val)
            beta = min(beta, best)

            # Alpha-Beta Pruning condition
            if beta <= alpha:
                break
        return best


# Example leaf node values (Game possible outcomes)
values = [3, 5, 6, 9, 1, 2, 0, -1]

print("Leaf Node Values:", values)
optimal_value = alpha_beta(0, 0, True, values, -math.inf, math.inf)

print("Optimal Value (Best Outcome for Maximizer):", optimal_value)
```

```
Leaf Node Values: [3, 5, 6, 9, 1, 2, 0, -1]
Optimal Value (Best Outcome for Maximizer): 5
```