```
import math

def alpha_beta_pruning(depth, node_index, is_maximizing_player, values, alpha, beta, max_depth):
    # Base case: when the maximum depth is reached
    if depth == max_depth:
        return values[node_index]

    if is_maximizing_player:
        best = -math.inf

        # Recur for left and right children
        for i in range(2):
            val = alpha_beta_pruning(depth + 1, node_index * 2 + i, False, values, alpha, beta, max_depth)
            best = max(best, val)
            alpha = max(alpha, best)

            # Prune the remaining nodes
            if beta <= alpha:
                break
        return best

    else:
        best = math.inf

        # Recur for left and right children
        for i in range(2):
            val = alpha_beta_pruning(depth + 1, node_index * 2 + i, True, values, alpha, beta, max_depth)
            best = min(best, val)
            beta = min(beta, best)

            # Prune the remaining nodes
            if beta <= alpha:
                break
        return best
print("SWAPNIL SAHIL(1BM22CS300):")

# Example usage
if __name__ == "__main__":
    # Example tree represented as a list of leaf node values
    values = [3, 5, 6, 9, 1, 2, 0, -1]
    max_depth = 3  # Height of the tree

    result = alpha_beta_pruning(0, 0, True, values, -math.inf, math.inf, max_depth)
    print("The optimal value is:", result)
```

```
SWAPNIL SAHIL(1BM22CS300):
The optimal value is: 5
```