```python
import math

# Sample game tree as a nested list (each list represents children of a node)
# Leaves are integers representing heuristic values
# This is a simple static tree example to visualize pruning
game_tree = [
    [3, 5, 6],  # First child has leaves 3,5,6
    [9, 1, 2],  # Second child has leaves 9,1,2
    [0, -1, 4]  # Third child has leaves 0,-1,4
]

# Global variable to count nodes evaluated
nodes_evaluated = 0

def alpha_beta(node, depth, alpha, beta, maximizing_player):
    global nodes_evaluated

    # If leaf node (depth == 0), return its value and count evaluation
    if depth == 0 or not isinstance(node, list):
        nodes_evaluated += 1
        return node

    if maximizing_player:
        value = -math.inf
        for child in node:
            value = max(value, alpha_beta(child, depth-1, alpha, beta, False))
            alpha = max(alpha, value)
            if beta <= alpha:
                # Beta cut-off
                break
        return value
    else:
        value = math.inf
        for child in node:
            value = min(value, alpha_beta(child, depth-1, alpha, beta, True))
            beta = min(beta, value)
            if beta <= alpha:
                # Alpha cut-off
                break
        return value


def minimax(node, depth, maximizing_player):
    global nodes_evaluated

    if depth == 0 or not isinstance(node, list):
        nodes_evaluated += 1
        return node

    if maximizing_player:
        value = -math.inf
        for child in node:
            value = max(value, minimax(child, depth-1, False))
        return value
    else:
        value = math.inf
        for child in node:
            value = min(value, minimax(child, depth-1, True))
        return value

def main():
    global nodes_evaluated

    depth = 2  # Since our sample tree has 3 levels: root -> children -> leaves

    # Minimax
    nodes_evaluated = 0
    optimal_value_minimax = minimax(game_tree, depth, True)
    minimax_nodes = nodes_evaluated

    # Alpha-Beta
    nodes_evaluated = 0
    optimal_value_alphabeta = alpha_beta(game_tree, depth, -math.inf, math.inf, True)
    alphabeta_nodes = nodes_evaluated

    print(f"Optimal value (Minimax): {optimal_value_minimax}")
    print(f"Nodes evaluated (Minimax): {minimax_nodes}")
    print(f"Optimal value (Alpha-Beta): {optimal_value_alphabeta}")
    print(f"Nodes evaluated (Alpha-Beta): {alphabeta_nodes}")

if __name__ == "__main__
```