class Node:

def \_\_init\_\_(self, value):

self.value = value

self.children = []

def alpha\_beta(node, depth, alpha, beta, maximizing\_player):

if depth == 0 or not node.children:

return node.value

if maximizing\_player:

max\_eval = float('-inf')

for child in node.children:

eval = alpha\_beta(child, depth - 1, alpha, beta, False)

max\_eval = max(max\_eval, eval)

alpha = max(alpha, eval)

if beta <= alpha:

break

return max\_eval

else:

min\_eval = float('inf')

for child in node.children:

eval = alpha\_beta(child, depth - 1, alpha, beta, True)

min\_eval = min(min\_eval, eval)

beta = min(beta, eval)

if beta <= alpha:

break

return min\_eval

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

root = Node(0)

child1 = Node(3)

child2 = Node(5)

child3 = Node(6)

child4 = Node(9)

child5 = Node(1)

child6 = Node(2)

root.children = [child1, child2, child3]

child1.children = [child4, child5]

child2.children = [child6]

depth = 3

result = alpha\_beta(root, depth, float('-inf'), float('inf'), True)

print("Optimal value:", result)

