# Alpha-Beta Pruning Implementation

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

# Base case: If it's a leaf node, return its value (simulating evaluation of the node) if type(node) is int:

return node

# If not a leaf node, explore the children if maximizing\_player:

max\_eval = -float('inf')

for child in node: # Iterate over children of the maximizer node eval = alpha\_beta\_pruning(child, alpha, beta, False)

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

alpha = max(alpha, eval) # Maximize alpha if beta <= alpha: # Prune the branch

break

return max\_eval else:

min\_eval = float('inf')

for child in node: # Iterate over children of the minimizer node eval = alpha\_beta\_pruning(child, alpha, beta, True)

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

beta = min(beta, eval) # Minimize beta if beta <= alpha: # Prune the branch

1

break

return min\_eval

# Function to build the tree from a list of numbers def build\_tree(numbers):

# We need to build a tree with alternating levels of maximizers and minimizers # Start from the leaf nodes and work up

current\_level = [[n] for n in numbers] while len(current\_level) > 1:

next\_level = []

for i in range(0, len(current\_level), 2):

if i + 1 < len(current\_level):

next\_level.append(current\_level[i] + current\_level[i + 1]) # Combine two nodes else:

next\_level.append(current\_level[i]) # Odd number of elements, just carry forward current\_level = next\_level

return current\_level[0] # Return the root node, which is a maximizer # Main function to run alpha-beta pruning

def main():

# Input: User provides a list of numbers

numbers = list(map(int, input("Enter numbers for the game tree (space-separated): ").split())) 2

# Build the tree with the given numbers tree = build\_tree(numbers)

# Parameters: Tree, initial alpha, beta, and the root node is a maximizing player alpha = -float('inf')

beta = float('inf')

maximizing\_player = True # The root node is a maximizing player # Perform alpha-beta pruning and get the final result

result = alpha\_beta\_pruning(tree, alpha, beta, maximizing\_player) print("Final Result of Alpha-Beta Pruning:", result)

if name == " main ": main()