Alpha Beta Pruning

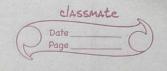
Code:

```
# 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
               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):</pre>
               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()))
   # 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()
```

Output:

Enter numbers for the game tree (space-separated): 10 9 14 18 5 4 50 3 Final Result of Alpha-Beta Pruning: 50



Problem:

Apply the Alpha - Beta search algorithm to find value of root mode to both to root node (MAX mode). Edentity the baths which are pruned (or cutoff) for exploration

