**SET\_A**

**Python Solution**  
  
class Node:

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

self.value = value

self.left = None

self.right = None

def mirror\_node\_sum(root):

def helper(left, right):

if left is None or right is None: # If one of the nodes is missing, no mirror pair exists

return 1

return (

left.value \* right.value \* # Sum current mirror nodes

helper(left.left, right.right) \* # Recur for outer mirrors

helper(left.right, right.left) # Recur for inner mirrors

)

if root is None: # If the tree is empty

return 0

return helper(root.left, root.right)

**SET\_B**

class Node:

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

self.value = value

self.left = None

self.right = None

def mirror\_node\_sum(root):

def helper(left, right):

if left is None or right is None: # If one of the nodes is missing, no mirror pair exists

return 0

return (

left.value + right.value + # Sum current mirror nodes

helper(left.left, right.right) + # Recur for outer mirrors

helper(left.right, right.left) # Recur for inner mirrors

)

if root is None: # If the tree is empty

return 0

return helper(root.left, root.right)

**Rubric**

Rubric:

| Portion | Marks |
| --- | --- |
| Correct Node Class | 3 |
| Correct base case(no mirror pairs exist) | 4 |
| Correct sum or product of mirror nodes | 2 |
| Correct recursive calls | 2 + 2 = 4 |
| Correct return statements | 2 |