Set A Solution:

| def count\_upward\_paths\_utility(cur\_node, prev\_node):  prev\_elem = prev\_node.elem  cur\_elem = cur\_node.elem    if prev\_elem>=cur\_elem:  return 0    if cur\_node.left is None and cur\_node.right is None:  return 1    ans = 0  if cur\_node.left is not None:  ans+=count\_upward\_paths\_utility(cur\_node.left, cur\_node)    if cur\_node.right is not None:  ans+=count\_upward\_paths\_utility(cur\_node.right, cur\_node)    return ans   def count\_upward\_paths(root):  ans = 0  if root.left is not None:  ans+=count\_upward\_paths\_utility(root.left, root)  if root.right is not None:  ans+=count\_upward\_paths\_utility(root.right, root)  return ans |
| --- |
|  |

Rubric:

| Portion | Marks |
| --- | --- |
| Writing the base case correctly (reached a leaf) | 3 |
| Checking whether current node is greater than the previous node | 4 |
| Calculating answer for left branch only if left child exists, similarly for right child | 2 + 2 |
| Adding both answers to final result | 2 + 2 |

Alternate approaches exist, and should be marked according to the faculties’ discretion.

Set B Solution:

| def count\_interesting\_paths\_utility(cur\_node, prev\_node):  prev\_parity = prev\_node.elem % 2  cur\_parity = cur\_node.elem % 2    if prev\_parity==cur\_parity:  return 0    if cur\_node.left is None and cur\_node.right is None:  return 1    ans = 0  if cur\_node.left is not None:  ans+=count\_interesting\_paths\_utility(cur\_node.left, cur\_node)    if cur\_node.right is not None:  ans+=count\_interesting\_paths\_utility(cur\_node.right, cur\_node)    return ans    def count\_interesting\_paths(root):  ans = 0  if root.left is not None:  ans+=count\_interesting\_paths\_utility(root.left, root)  if root.right is not None:  ans+=count\_interesting\_paths\_utility(root.right, root)  return ans |
| --- |

Rubric:

| Portion | Marks |
| --- | --- |
| Writing the base case correctly (reached a leaf) | 3 |
| Checking parity of consecutive nodes correctly | 4 |
| Calculating answer for left branch only if left child exists, similarly for right child | 2 + 2 |
| Adding both answers to final result | 2 + 2 |

Alternate approaches exist, and should be marked according to the faculties’ discretion.   
0000000