**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**