**SET A**

**Python**

| **def product\_greater\_than\_k(root, k):**  if root is None:  return 1  if root.val <= k:  return product\_greater\_than\_k(root.right, k)  else:  return root.val \* product\_greater\_than\_k(root.left, k) \* product\_greater\_than\_k(root.right, k) |
| --- |

**Java**

| **public static int productGreaterThanK(Node root, int k)** {  if (root == null) return 1;  if (root.val <= k)  return productGreaterThanK(root.right, k);  else  return root.val \* productGreaterThanK(root.left, k) \* productGreaterThanK(root.right, k);  } |
| --- |

**SET B**

**Python**

| **def count\_divisible\_nodes(root, target):**  if root is None:  return 0  if root.val % target == 0:  count = 1  else:  count = 0  count += count\_divisible\_nodes(root.left, target)  count += count\_divisible\_nodes(root.right, target)  return count |
| --- |

**Java**

| **public static int countDivisibleNodes(Node root, int target)** {  if (root == null) return 0;  int count;  if (root.val % target == 0)  count = 1;  else  count = 0;  count += countDivisibleNodes(root.left, target);  count += countDivisibleNodes(root.right, target);  return count;  } |
| --- |

**Rubric**

| **Category** | **Marks** |
| --- | --- |
| Proper Method/Function Definition with Proper Parameters | 2 |
| Base case and recursion | 6 |
| Logical condition | 5 |
| Correct result returned | 2 |