Given a Binary Search Tree (BST) and a range l-h(inclusive), count the number of nodes in the BST that lie in the given range.

- **The values smaller than root go to the left side**
- **The values greater and equal to the root go to the right side**

Example 1:

Input:

```
     10
    /  \
   5    50
  /    /  \
 1    40  100
```

l = 5, h = 45 Output: 3

Explanation: 5 10 40 are the node in the range

Example 2:

Input:

```
    5
   /  \
  4    6
 /      \
3        7
```

l = 2, h = 8 Output: 5

Explanation: All the nodes are in the given range.

Your Task:

This is a function problem. You don't have to take input. You are required to complete the function getCountOfNode() that takes root, l ,h as parameters and returns the count.

Expected Time Complexity: O(N)

Expected Auxiliary Space: O(Height of the BST).

Constraints:

1 <= Number of nodes <= 100

1 <= l < h < 103

Solution:

```
class Solution

{

    public void getinorder(int[] arr, Node root, int l, int h){

        if(root == null){

            return;

        }

        getinorder(arr, root.left, l, h);

        if(root.data >= l && root.data <= h){

            arr[0]++;

        }

        getinorder(arr, root.right, l, h);  // Change root.left to root.right here

    }

     int getCount(Node root, int l, int h)

    {

        int[] arr=new int[1];

        getinorder(arr, root, l, h);

        return arr[0];

    }

}
```