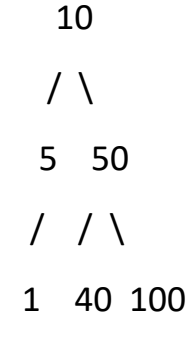
Count BST nodes that lie in a given range

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:



l = 5, h = 45

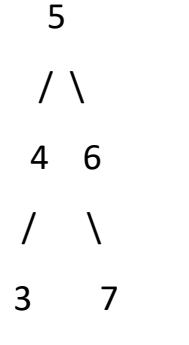
Output: 3

Explanation: 5 10 40 are the node in the

range

Example 2:

Input:



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 < 10^3

Try more examples

```
#User function Template for python3


#Function to count number of nodes in BST that lie in the
given range.
class Solution:
    def getCount(self,root,low,high):
        ##Your code here
        if not root:
            return 0
        if low<=root.data<=high:
            return 1 + self.getCount(root.left,low,high) +
self.getCount(root.right,low,high)

        if root.data < low:
            return self.getCount(root.right, low, high)
        else:
            return self.getCount(root.left, low, high)

#{
 # Driver Code Starts
```

```
#Initial Template for Python 3

from collections import deque
# Tree Node
class Node:
    def __init__(self, val):
        self.right = None
        self.data = val
        self.left = None

# Function to Build Tree
def buildTree(s):
    #Corner Case
    if(len(s)==0 or s[0]=="N"):
        return None

    # Creating list of strings from input
    # string after spliting by space
    ip=list(map(str,s.split()))

    # Create the root of the tree
    root=Node(int(ip[0]))
    size=0
    q=deque()

    # Push the root to the queue
    q.append(root)
    size=size+1

    # Starting from the second element
    i=1
    while(size>0 and i<len(ip)):
        # Get and remove the front of the queue
        currNode=q[0]
        q.popleft()
        size=size-1

        # Get the current node's value from the string
        currVal=ip[i]

        # If the left child is not null
        if(currVal!="N"):
```

```
            # Create the left child for the current node
            currNode.left=Node(int(currVal))

            # Push it to the queue
            q.append(currNode.left)
            size=size+1
        # For the right child
        i=i+1
        if(i>=len(ip)):
            break
        currVal=ip[i]

        # If the right child is not null
        if(currVal!="N"):

            # Create the right child for the current node
            currNode.right=Node(int(currVal))

            # Push it to the queue
            q.append(currNode.right)
            size=size+1
        i=i+1
    return root


if __name__=="__main__":
    t=int(input())
    for _ in range(0,t):
        s=input()
        root=buildTree(s)
        l, r=map(int, input().split())
        obj=Solution()
        print(obj.getCount(root,l,r))
# } Driver Code Ends
```