Count BST subtrees that lie in given range

Given a Binary Search Tree (BST) of integer values and a range [low, high], return count of nodes where all the nodes under that node (or subtree rooted with that node) lie in the given range.

Examples:

```
Input:
        10
      /    \
    5       50
   /       /  \
 1       40   100
Range: [5, 45]
Output:  1
There is only 1 node whose subtree is in the given range.
The node is 40


Input:
        10
      /    \
    5       50
   /       /  \
 1       40   100
Range: [1, 45]
Output:  3
There are three nodes whose subtree is in the given range.
The nodes are 1, 5 and 40
```

We strongly recommend you to minimize your browser and try this yourself first.

The idea is to traverse the given Binary Search Tree (BST) in bottom up manner. For every node, recur for its subtrees, if subtrees are in range and the nodes is also in range, then increment count and return true (to tell the parent about its status). Count is passed as a pointer so that it can be incremented across all function calls.

Below is C++ implementation of the above idea.

```
// C++ program to count subtrees that lie in a given range
#include<bits/stdc++.h>
using namespace std;

// A BST node
struct node
{
    int data;
    struct node* left, *right;
};

// A utility function to check if data of root is
// in range from low to high
bool inRange(node *root, int low, int high)
{
    return root->data >= low && root->data <= high;
}

// A recursive function to get count of nodes whose subtree
// is in range from low to hgih.  This function returns true
// if nodes in subtree rooted under 'root' are in range.
bool getCountUtil(node *root, int low, int high, int *count)
{
    // Base case
    if (root == NULL)
       return true;

    // Recur for left and right subtrees
    bool l = (root->left) ?  getCountUtil(root->left, low, high, count) : true;
    bool r = (root->right) ? getCountUtil(root->right, low, high, count) : true;


    // If both left and right subtrees are in range and current node
    // is also in range, then increment count and return true
    if (l && r && inRange(root, low, high))
    {
       ++*count;
       return true;
    }

    return false;
}

// A wrapper over getCountUtil(). This function initializes count as 0
//  and calls getCountUtil()
int getCount(node *root, int low, int high)
{
    int count = 0;
    getCountUtil(root, low, high, &count);
    return count;
}

// Utility function to create new node
node *newNode(int data)
{
    node *temp = new node;
    temp->data  = data;
    temp->left  = temp->right = NULL;
    return (temp);
}

// Driver program
int main()
{
    // Let us construct the BST shown in the above figure
    node *root        = newNode(10);
    root->left        = newNode(5);
    root->right       = newNode(50);
    root->left->left  = newNode(1);
    root->right->left = newNode(40);
    root->right->right = newNode(100);
    /* Let us constructed BST shown in above example
          10
        /    \
      5       50
     /       /  \
    1       40   100   */
    int l = 5;
    int h = 45;
    cout << "Count of subtrees in [" << l << ", "
         << h << "] is " << getCount(root, l, h);
    return 0;
}
```

Output:

```
Count of subtrees in [5, 45] is 1
```