Count BST nodes that lie in a given range

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

Examples:

```
Input:
        10
      /    \
    5       50
   /       /  \
 1       40   100
Range: [5, 45]

Output:  3
There are three nodes in range, 5, 10 and 40
```

Source: Google Question

We strongly recommend that you click here and practice it, before moving on to the solution.

The idea is to traverse the given binary search tree starting from root. For every node being visited, check if this node lies in range, if yes, then add 1 to result and recur for both of its children. If current node is smaller than low value of range, then recur for right child, else recur for left child.

Below is C++ implementation of above idea.

```
// C++ program to count BST nodes withing a given range
#include<bits/stdc++.h>
using namespace std;

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

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

// Returns count of nodes in BST in range [low, high]
int getCount(node *root, int low, int high)
{
    // Base case
    if (!root) return 0;

    // Special Optional case for improving efficiency
    if (root->data == high && root->data == low)
        return 1;

    // If current node is in range, then include it in count and
    // recur for left and right children of it
    if (root->data <= high && root->data >= low)
         return 1 + getCount(root->left, low, high) +
                    getCount(root->right, low, high);

    // If current node is smaller than low, then recur for right
    // child
    else if (root->data < low)
         return getCount(root->right, low, high);

    // Else recur for left child
    else return getCount(root->left, low, high);
}

// 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 nodes between [" << l << ", " << h
         << "] is " << getCount(root, l, h);
    return 0;
}
```

Output:

```
Count of nodes between [5, 45] is 3
```

Time complexity of the above program is O(h + k) where h is height of BST and k is number of nodes in given range.