16) COUNT BST NODES THAT LIE IN THE GIVEN RANGE

METHOD 1:
In this method, we use the morris traversal to traverse through all the elements and while traversing the we simultaneously check whether the element lies in the given range or not.
Time Complexity : O(N)
Space Complexity : O(1)

METHOD 2:
The same sum can be done using the level order Traversal or InOrder Traversal and While traversing, we check whether the element lies in the given range or not.
(This can be done using the global variables also or by passing the count variable by address.)
Time Complexity : O(N)
Space Complexity : O(log N)

CODE OF METHOD 1:

```
int getCount(Node *root, int l, int h)
{
   int count=0;
   Node *current=root;
   while(current!=NULL){
      if(current->left==NULL){
         if(current->data>=l && current->data<=h){
            count++;
         }
         current=current->right;
      }
      else{
         Node *previous=current->left;
         while(previous->right!=NULL && previous->right!=current){
            previous=previous->right;
         }
         if(previous->right==NULL){
            previous->right=current;
            current=current->left;
         }
         else{
            previous->right=NULL;
            if(current->data>=l && current->data<=h){
               count++;
            }
            current=current->right;
         }
      }
   }
   return count;
}
```

CODE OF METHOD 2:

```
void funct(Node *root,int low,int high,int &count){
    if(root!=NULL){
        funct(root->left,low,high,count);
        if(root->data>=low && root->data<=high){
            count++;
        }
        funct(root->right,low,high,count);
    }
}

int getCount(Node *root, int l, int h)
{
    int count=0;
    funct(root,l,h,count);
    return count;
}
```