13)KTH SMALLEST ELEMENT IN THE BST:(THIS SUM IS ALMOST SAME AS THE PREVIOUS SUM)

METHOD 1:
The first method is that, We take the InOrder Traversal of the Binary Tree into the Array and then return the kth element from the start as the answer.
Time Complexity : O(N)
Space Complexity : O(N)

METHOD 2:
This method is by using the global variable, same as that of the previous sum.
But This is not the preferred method as we are using the global variable.
Space Complexity : O(1)
Time Complexity : O(N)

METHOD 3:
This Method is same as that of the Method 2 But without using global variable**(Preferred and efficient).**

CODE FOR METHOD 2:

```
class Solution {
 public:
   int answer=-1;
   // Return the Kth smallest element in the given BST
   void InOrderTraversal(Node *root,int &count,int k){
      if(root!=NULL){
         InOrderTraversal(root->left,count,k);
         count++;
         if(count==k){
            answer=root->data;
            return;
         }
         InOrderTraversal(root->right,count,k);
      }
   }
   int KthSmallestElement(Node *root, int k) {
      int count=0;
      answer=-1;
      InOrderTraversal(root,count,k);
      return answer;
   }
};
```

CODE FOR METHOD 3:

```
class Solution {
 public:
   // Return the Kth smallest element in the given BST
   int InOrderTraversal(Node *root,int &count,int k){
       if(root!=NULL){
           int element=InOrderTraversal(root->left,count,k);
           if(count==k){
               return element;
           }
           else{
               count++;
               if(count==k){
                   return root->data;
               }
           }
           return InOrderTraversal(root->right,count,k);
       }
       return -1;    //Most Important Thing in question that the value of k can be higher than the
number of nodes in the BST so in such case the output will be -1.
   }
   int KthSmallestElement(Node *root, int k) {
       int count=0;
       int answer=InOrderTraversal(root,count,k);
       return answer;
   }
};
```