21) THE SIZE OF THE LARGEST BST IN THE BINARY TREE.

METHOD:

This sum is very important and one of the best sum.
In this sum we first recursively traverse to the bottom of the binary tree and start returning the vector of size 4 from the bottom.
Index 0 : true or false.
Index 1 : Size of the largest BST below it.
Index 2: The minimum element on that subtree.
Index 3: The maximum element on that subtree.

It's like a bottom Up Approach and the value returned from both the side is compared.
If both forms the BST within itself and also forms BST with the current node then the returned value is size of left +size of right+1.
Else the value returned is the {false,max size between left And right,0,0}

LINK OF EXPLANATION: Largest BST in a Binary Tree | BST | Love Babbar DSA Sheet | A…

CODE FOR THE METHOD:

```
class Solution{
    public:
    /*You are required to complete this method */
    // Return the size of the largest sub-tree which is also a BST

    vector<int> fun(Node *root){
        if(root==NULL){
            return {1,0,INT_MAX,INT_MIN};
        }
        else{
            vector<int> left=fun(root->left);
            vector<int> right=fun(root->right);
            if(left[0]==1 && right[0]==1 && left[3]<root->data && right[2]>root->data){
                return {1,1+left[1]+right[1],min(left[2],root->data),max(root->data,right[3])};
            }
            else{
                return {0,max(left[1],right[1]),0,0};
            }
        }
    }

    int largestBst(Node *root){
        vector<int> answer=fun(root);
        return answer[1];
    }
};
```