1)FIND A VALUE IN BST (STANDARD SUM)

```
#include<iostream>
#include<string>
#include<vector>
#include<queue>
using namespace std;

class node{
    public:
    int data;
    node *left=NULL;
    node *right=NULL;
};

void Traversal(node *head){
    if(head!=NULL){
        Traversal(head->left);
        cout<<head->data<<" ";
        Traversal(head->right);
    }
}

class BinarySearchTree{
    public:
    node *head=NULL;
    public:
    BinarySearchTree(vector<string> &initialisation){
        queue<node*> q;
        node *root=new node;
        root->data=stoi(initialisation[0]);
        q.push(root);
        int k=1;
        this->head=root;
        while(q.size()!=0 && k+1<initialisation.size()){
            if(initialisation[k]=="N" && initialisation[k+1]=="N"){
                q.pop();
            }
            else if(initialisation[k]=="N" && initialisation[k+1]!="N"){
                node *temp=new node;
```

```
                temp->data=stoi(initialisation[k+1]);
                q.front()->right=temp;
                q.push(temp);
                q.pop();
            }
            else if(initialisation[k]!="N" && initialisation[k+1]=="N"){
                node *temp=new node;
                temp->data=stoi(initialisation[k]);
                q.front()->left=temp;
                q.push(temp);
                q.pop();
            }
            else{
                node *temp1=new node;
                node *temp2=new node;
                temp1->data=stoi(initialisation[k]);
                temp2->data=stoi(initialisation[k+1]);
                q.front()->left=temp1;
                q.front()->right=temp2;
                q.push(temp1);
                q.push(temp2);
                q.pop();
            }
            k=k+2;
        }
        if(k+1==initialisation.size() && q.size()!=0 &&
initialisation[k]!="N"){
            node *temp=new node;
            temp->data=stoi(initialisation[k]);
            q.front()->left=temp;
        }
    }
    void InOrderTraversal(){
        Traversal(head);
    }
};

bool Search(node *head,int search_element){
    if(head->data==search_element){
```

```
        return true;
    }
    else{
        if(head->data<search_element){
            if(head->right==NULL){
                return false;
            }
            else{
                return Search(head->right,search_element);
            }
        }
        else{
            if(head->left==NULL){
                return false;
            }
            else{
                return Search(head->left,search_element);
            }
        }
    }
}

int main(){
    int n,search_element;
    cout<<"\n Enter the number of nodes present in the Binary Search
Tree:";
    cin>>n;
    vector<string> initialisation(n);
    cout<<"\n Enter the node values of the Binary Search Tree:";
    for(int i=0;i<n;i++){
        cin>>initialisation[i];
    }
    BinarySearchTree bst(initialisation);
    cout<<"\n The InOrder Traversal of the Binaary Search Tree:";
    bst.InOrderTraversal();
    cout<<"\n Enter the element to be searched in the Binary Search
Tree:";
    cin>>search_element;
    bool answer=Search(bst.head,search_element);
```

```
    if(answer==false){
        cout<<"\n The Given element is not present in the Binary Search
Tree.";
    }
    else{
        cout<<"\n The Given element is present in the binary Search
Tree.";
    }
    return 0;
}
/*
Sample Input:
nodes:16
node Values:20 10 30 N 15 25 40 7 18 N N N N N N 17
*/
```