LAB 7 - (1)Qns - BST 7

1.Given N Nodes of a binary tree Create a Binary Search Tree. 2.Given a Node return value of its Parent and also the height of the parent from root. Consider root as 0.

Input Format

First line has Number of nodes in the BST Second Line has space N integers of tree Third Line has the value of Key

Constraints

N<100 key<100

Output Format

prints the parent value (space) Line prints the height of the key .

Sample Input 0

```
 4
3 2 1 4
4
```

Sample Output 0

```
3 1
```

```
#include<stdio.h>
#include<stdlib.h>
struct Node
{

};

struct Node* createNode(int data){


struct Node* insert(struct Node* root,int data){
    //insert the values in the tree and return the updated root.
}
void findParentAndHeight(struct Node* root,int target){
    //Complete the logic here.
        printf("%d %d",parent->data,height);
```

```
}

int main(){
    struct Node* root=NULL;
    int n;
    scanf("%d",&n);
    for (int i=0;i<n;i++){
        int element;
        scanf("%d",&element);
        root=insert(root,element);
    }
    int target;
    scanf("%d",&target);
    findParentAndHeight(root,target);
    return 0;
}
#include<stdio.h>
#include<stdlib.h>
struct Node
{
    //Complete the node structure.
};

struct Node* createNode(int data){
    //Complete thr createNode function

struct Node* insert(struct Node* root,int data){
    //insert the values in the tree and return the updated root.
}
void findParentAndHeight(struct Node* root,int target){
    //Complete the logic here.
        printf("%d %d",parent->data,height);
    //print the values in the same format as shown above to pass the test cases.

}

int main(){
    struct Node* root=NULL;
    int n;
    scanf("%d",&n);
    for (int i=0;i<n;i++){
        int element;
        scanf("%d",&element);
        root=insert(root,element);
    }
    int target;
    scanf("%d",&target);
    findParentAndHeight(root,target);
    return 0;
}
```