Started on	Monday, 4 March 2024, 8:00 PM
State	Finished
Completed on	Monday, 4 March 2024, 8:57 PM
Time taken	57 mins 33 secs
Marks	20.00/20.00
Grade	**10.00** out of 10.00 (**100**%)

Question 1

Correct

Mark 10.00 out of 10.00

You are given a pointer to the root of a binary search tree and values to be inserted into the tree. Insert the values into their appropriate position in the binary search tree and return the root of the updated binary tree. You just have to complete the function.

You are given a function,

```
Node * insert (Node * root ,int data) {

}
```

Input Format

- First line of the input contains *t*, the number of nodes in the tree.
- Second line of the input contains the list of *t* elements to be inserted to the tree.

Constraints

- No. of nodes in the tree, $1 \leq t \leq 5000$
- Value of each node in the tree, $1 \leq t[i] \leq 10000$

Output Format

Return the items in the binary search tree after inserting the values into the tree. Start with the root and follow each element by its left subtree, and then its right subtree.

Sample Input

```
6
4 2 3 1 7 6
```

Sample Output

```
4 2 1 3 7 6
```

Sample Explanation

The binary tree after inserting the 6 elements in the given order will look like this.

```
        4
     /   \
    2     7
   / \   /
  1   3 6
```

For example:

Input	Result
6 4 2 3 1 7 6	4 2 1 3 7 6
19 44 67 91 20 87 20 31 11 19 39 86 65 57 84 10 72 84 15 46	44 20 11 10 19 15 20 31 39 67 65 57 46 91 87 86 84 72 84

Answer: (penalty regime: 0 %)

Reset answer

```
#include <bits/stdc++.h>

using namespace std;

class Node {
    public:
        int data;
        Node *left;
        Node *right;
        Node(int d) {
```

```
            data = d;
            left = NULL;
            right = NULL;
        }
};

class Solution {
    public:

    void preOrder(Node *root) {

        if( root == NULL )
            return;

        std::cout << root->data << " ";

        preOrder(root->left);
        preOrder(root->right);
    }

/*
Node is defined as

class Node {
    public:
        int data;
        Node *left;
        Node *right;
        Node(int d) {
            data = d;
            left = NULL;
            right = NULL;
        }
};

*/

    Node * insert(Node * root, int data) {
        if(root==NULL){//when root is null create a new node
           return new Node(data);
        }
        if (data>=root->data){//when data is greater than root insert node to right
```

	Input	Expected	Got	
✔	6 4 2 3 1 7 6	4 2 1 3 7 6	4 2 1 3 7 6	✔
✔	19 44 67 91 20 87 20 31 11 19 39 86 65 57 84 10 72 84 15 46	44 20 11 10 19 15 20 31 39 67 65 57 46 91 87 86 84 72 84	44 20 11 10 19 15 20 31 39 67 65 57 46 91 87 86 84 72 84	✔

Passed all tests! ✔

▸ Show/hide question author's solution (Cpp)

Correct

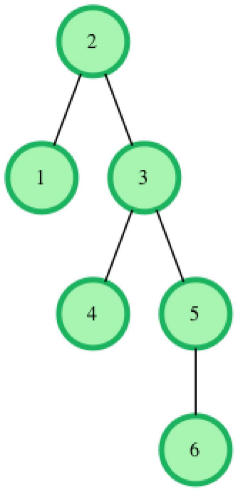
Marks for this submission: 10.00/10.00.

Question 2

Correct

Mark 10.00 out of 10.00

You are given pointer to the root of the binary search tree and two values $v1$ and $v2$. You need to return the lowest common ancestor (LCA) of $v1$ and $v2$ in the binary search tree.



In the diagram above, the lowest common ancestor of the nodes 4 and 6 is the node 3. Node 3 is the lowest node which has nodes 4 and 6 as descendants.

Function Description

Complete the function *lca* in the editor below. It should return a pointer to the lowest common ancestor node of the two values given.

lca has the following parameters:
- root: a pointer to the root node of a binary search tree
- v1: a node.data value
- v2: a node.data value

Input Format

The first line contains an integer, n, the number of nodes in the tree.
The second line contains n space-separated integers representing $node.data$ values.
The third line contains two space-separated integers, $v1$ and $v2$.

To use the test data, you will have to create the binary search tree yourself.

Constraints

$1 \leq n, node.data \leq 5000$
$1 \leq v1, v2 \leq 5000$
$v1 \neq v2$

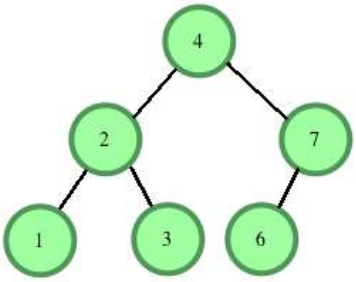
The tree will contain nodes with *data* equal to $v1$ and $v2$.

Output Format

Return the value of the node that is the lowest common ancestor of $v1$ and $v2$.

Sample Input

```
6
4 2 3 1 7 6
1 7
```



$v1 = 1$ and $v2 = 7$.

Sample Output

4

Explanation

LCA of 1 and 7 is 4, the root in this case.
Return a pointer to the node.

For example:

Input	Result
6 4 2 3 1 7 6 1 7	4
2 1 2 1 2	1

Answer: (penalty regime: 0 %)

```
#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node *left, *right;

    // Constructor to initialize the Node with a value
    Node(int x) {
        data = x;
        left = right = nullptr;
    }
};

class BinarySearchTree {
public:
    // Function to insert a new node with a given value into the binary search tree
    Node* insert(Node* root, int value) {
        // If the tree is empty, create a new node and return it as the root
        if (root == nullptr)
            return new Node(value);
        // If the value is less than the root, insert it into the left subtree
        if (value < root->data)
            root->left = insert(root->left, value);
        // If the value is greater than or equal to the root, insert it into the right subtree
        else
            root->right = insert(root->right, value);
        return root;
    }

    // Function to find the Lowest Common Ancestor (LCA) of two nodes in the binary search tree using recurs
    Node* lca(Node* root, int value1, int value2) {
        // If the root is NULL or if both values are less than or greater than the root, return NULL
        if (root == nullptr)
            return nullptr;
        // If both values are less than the root, search in the left subtree
        if (value1 < root->data && value2 < root->data)
            return lca(root->left, value1, value2);
        // If both values are greater than the root, search in the right subtree
        else if (value1 > root->data && value2 > root->data)
            return lca(root->right, value1, value2);
        // If one value is less than the root and the other is greater, or if one value is equal to the root
        // then the root is the LCA
        else
            return root;
    }
};

int main() {
    BinarySearchTree bst;
    Node* root = nullptr;
```

	Input	Expected	Got	
✔	6 4 2 3 1 7 6 1 7	4	4	✔
✔	2 1 2 1 2	1	1	✔
✔	3 5 3 7 3 7	5	5	✔

Passed all tests! ✔

▸ **Show/hide question author's solution (Cpp)**

Correct

Marks for this submission: 10.00/10.00.