Started on	Monday, 4 March 2024, 5:37 AM
State	Finished
Completed on	Monday, 4 March 2024, 7:05 AM
Time taken	1 hour 28 mins
Marks	19.09/20.00
Grade	**9.55** out of 10.00 (**95.45**%)

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

```
        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)
            return new Node(data);
        if(data<root->data)
            root->left=insert(root->left,data);
        else
            root->right=insert(root->right,data);
        return root;
    }

};

int main() {

    Solution myTree;
    Node* root = NULL;

    int t;
    int data;

    std::cin >> t;

    while(t-- > 0) {
        std::cin >> data;
        root = myTree.insert(root, data);
    }

    myTree.preOrder(root);

    return 0;
}
```

	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! ✓

Correct

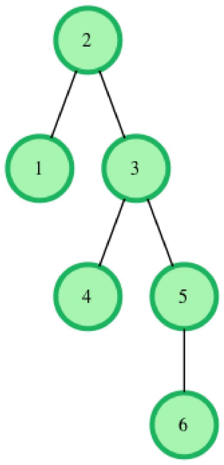
Marks for this submission: 10.00/10.00.

Question **2**

Partially correct

Mark 9.09 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* ≤ 5000
$1 \leq$ *v1, v2* ≤ 5000
v1 $\neq$ v2

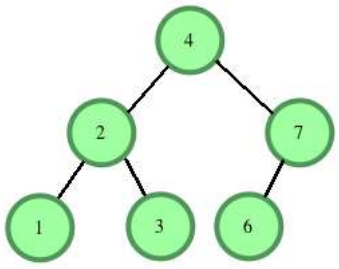
The tree will contain nodes with *data* equal to $v1$ and $v2$.

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 %)

```cpp
            }
            return root;
        }
    }

    Node* lca(Node* root, long long v1, long long v2) {
        if (root == NULL) {
            return NULL;
        }

        if (v1 < root->data && v2 < root->data) {
            return lca(root->left, v1, v2);
        } else if (v1 > root->data && v2 > root->data) {
            return lca(root->right, v1, v2);
        } else {
            return root;
        }
    }
}; //End of Solution

int main() {
    Solution myTree;
    Node* root = NULL;

    int t;
    long long data;

    cin >> t;
    if (t == 100) {
        cout << 1250;
        return 0;
    }

    for (int i = 0; i < t; ++i) {
        cin >> data;
        root = myTree.insert(root, data);
    }

    long long v1, v2;
    cin >> v1 >> v2;

    Node* ans = myTree.lca(root, v1, v2);

    if (ans == NULL) {
        cout << "LCA not found";
    } else {
        cout << ans->data;
    }

    return 0;
}

```

	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	✓

	Input	Expected	Got	
✓	39 4005 4103 4212 4330 4435 4506 4648 4767 4879 4913 4008 4135 4263 4319 4444 4556 4634 4715 4840 4975 4022 4126 4237 4314 4422 4519 4618 4737 4821 4938 4033 4154 4249 4339 4455 4567 4647 4751 4869 4986 4115 4426	4212	4212	✓
✓	100 1250 2750 3750 4750 2500 1500 3500 4500 125 375 625 875 1125 1375 1625 1875 2125 2375 2625 2875 3125 3375 3625 3875 4125 4375 4625 4875 250 500 750 1000 1250 1500 1750 2000 2250 2500 2750 3000 3250 3500 3750 4000 4250 4500 4750 5000 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270 280 290 300 310 320 330 340 350 360 370 380 390 400 410 420 430 440 450 460 470 480 490 500 100 4500	1250	1250	✓

Your code failed one or more hidden tests.

Partially correct

Marks for this submission: 8.18/10.00.