Started on	Monday, 4 March 2024, 7:59 PM
State	Finished
Completed on	Monday, 4 March 2024, 8:49 PM
Time taken	50 mins 28 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

```
16
17 class Solution {
18     public:
19
20     void preOrder(Node *root) {
21
22         if( root == NULL )
23             return;
24
25         std::cout << root->data << " ";
26
27         preOrder(root->left);
28         preOrder(root->right);
29     }
30
```

```
/*
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 the tree is empty, create a new node and return it as the root
    if(root == NULL) {
        return new Node(data);
    }

    // If the data is less than the current node's data, insert into the left subtree
    if(data < root->data) {
        root->left = insert(root->left, data);
    }
    // If the data is greater than or equal to the current node's data, insert into the right subtree
    else {
        root->right = insert(root->right, data);
    }

    return root; // Return the root after insertion
}

};

```

	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

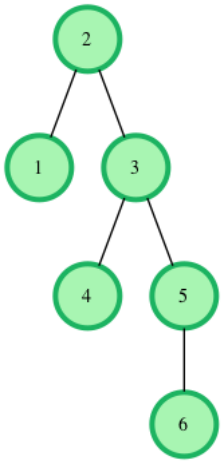
Marks for this submission: 10.00/10.00.

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**

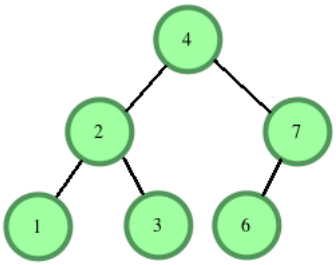
The tree will contain nodes with *data* equal to $v1$ and $v2$.

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 %)

```
    Node *left, *right;
    Node(int x) {
        val = x;
        left = right = NULL;
    }
};

// Function to insert a new value into the binary search tree
Node* insert(Node* root, int x) {
    // If the root is null, create a new node with the given value
    if (root == NULL)
        return new Node(x);
    // If the value is less than the current node's value, insert into the left subtree
    if (x < root->val)
        root->left = insert(root->left, x);
    // If the value is greater than or equal to the current node's value, insert into the right subtree
    else
        root->right = insert(root->right, x);
    return root;
}

// Function to find the Lowest Common Ancestor (LCA) of two values in the binary search tree
Node* lca(Node* root, int v1, int v2) {
    // If the root is null, return null (base case)
    if (root == NULL)
        return NULL;
    // If both values are less than the current node's value, search in the left subtree
    if (v1 < root->val && v2 < root->val)
        return lca(root->left, v1, v2);
    // If both values are greater than the current node's value, search in the right subtree
    else if (v1 > root->val && v2 > root->val)
        return lca(root->right, v1, v2);
    // If one value is less than the current node's value and the other is greater,
    // or if one value matches the current node's value, then the current node is the LCA
    else
        return root;
}

// Main function to test the above functions
int main() {
    Node* root = NULL; // Initialize the root of the binary search tree
    int n;
    cin >> n; // Input the number of elements in the binary search tree
    // Loop to input the elements and construct the binary search tree
    for (int i = 0; i < n; i++) {
        int x;
        cin >> x; // Input the value
        root = insert(root, x); // Insert the value into the binary search tree
    }
    int v1, v2;
    cin >> v1 >> v2; // Input the two values to find their Lowest Common Ancestor (LCA)
    Node* result = lca(root, v1, v2); // Find the LCA
```

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

	Input	Expected	Got	
✔	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.