Binary Search Tree : Lowest Common Ancestor

```
#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:
        Node* insert(Node* root, int data) {
            if(root == NULL) {
                return new Node(data);
            } else {
                Node* cur;
                if(data <= root->data) {
                    cur = insert(root->left, data);
                    root->left = cur;
                } else {
                    cur = insert(root->right, data);
                    root->right = cur;
                }

                return root;
            }
        }

/*The tree node has data, left child and right child
class Node {
    int data;
    Node* left;
    Node* right;
};
```

```
*/

    Node *lca(Node *root, int v1,int v2) {
        // Write your code here.
        if(v1 < root->data && v2 < root->data)
        return lca(root->left, v1, v2);
        if(v1 > root->data && v2 > root->data)
        return lca(root->right, v1, v2);
        return root;
    }

}; //End of Solution

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);
    }

    int v1, v2;
    std::cin >> v1 >> v2;

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

    std::cout << ans->data;

    return 0;
}
```