BST 6

```
//lowest common ancestor
#include <bits/stdc++.h>
using namespace std;
struct node
{
	struct node* left;
	struct node* right;
	int data;
};
struct node* insert(struct node* root,int x)
{
	if(root == NULL)
	{
		struct node* temp = (struct node*)malloc(sizeof(struct node));
		temp->left = NULL;
		temp->right = NULL;
		temp->data = x;
		return temp;
	}
	if(x > root->data)
		root->right = insert(root->right,x);
	else
		root->left = insert(root->left,x);
	return root;
}
void inorder(struct node *root)
{
   if (root != NULL)
   {
      inorder(root->left);
         cout<<root->data<<" ";
      inorder(root->right);
   }
}
struct node* common_ancestor(struct node* root,int a,int b)
{
	if(min(a,b) <= root->data && max(a,b) > root->data)
		return root;
	else if(max(a,b) < root->data)
		return common_ancestor(root->left,a,b);
	else if(min(a,b) > root->data)
		return common_ancestor(root->right,a,b);
	else
		return NULL;
}
int main()
{
	struct node* root = NULL;
```

```
        struct node* temp = NULL;
        int a,b,x,y;
        while(1)
        {
                cin>>x;
                if(x == 1)
                {
                        cin>>y;
                        root = insert(root,y);
                }
                if(x == 2)
                {
                        cin>>a>>b;
                        temp = common_ancestor(root,a,b);
                        if(temp == NULL)
                                cout<<"NO common_ancestor"<<endl;
                        else
                                cout<<temp->data<<endl;
                }
                if(x == 3)
                {
                        inorder(root);
                        cout<<endl;
                }
                if(x == 4)
                        return 0;
        }
}
```