* Dynamic Matrix

    int N; cin >> N;

    int \*\*matriz = new int\*[N];

    for(int i = 0; i < N; i++)

    {

        matriz[i] = new int[N];

    }

* Nodos y LCA:

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

class TreeNode

{

public:

    int key;

    TreeNode \*left;

    TreeNode \*right;

*// constructor*

    TreeNode(int \_key = 0)

    {

        key = \_key;

        left = NULL;

        right = NULL;

    }

};

void printNodesV1(TreeNode \*head, int n)

{

*//cout << "Entra funcion" << endl;*

    if(head == NULL)

    {

        return;

    }

    printNodesV1(head->right, n+1);

    for(int i = 0; i < n; i++)

    {

        cout << "     ";

    }

*//cout << "a" << endl;*

    cout << head->key << endl;

    printNodesV1(head->left, n+1);

}

int lca(TreeNode \*root, int n1, int n2)

{

    if(root == NULL){ *//cout << "LLegaste a una hoja" << endl;*

        return NULL;

    }

*//cout << "puntero a: " << root->key << endl;*

    if(root->key == n1 || root->key == n2){

*//cout << "Encontro nodo" << endl;*

        return root->key;

    }

    int left1 = lca(root->left, n1, n2);

*//cout << "left: " << left1 << endl;*

    int rigth1 = lca(root->right, n1, n2);

*//cout << "rigth: " << rigth1 << endl;*

    if(left1 != NULL && rigth1 != NULL)

    { *//cout << "Key: " << root->key << endl;*

        return root->key;

    }

    if( left1 == NULL && rigth1 != NULL )

    {

        return rigth1;

    }

    if( left1 != NULL && rigth1 == NULL )

    {

        return left1;

    }

    if(left1 == NULL && rigth1 == NULL)

    { *//cout << "Si ambos retornan null" << endl;*

        return NULL;

    }

}

int main()

{

    TreeNode \*root = new TreeNode(3);

    root->left = new TreeNode(6);

    root->right = new TreeNode(8);

    root->left->left = new TreeNode(2);

    root->left->right = new TreeNode(11);

    root->left->right->left = new TreeNode(9);

    root->left->right->right = new TreeNode(5);

    root->right->right = new TreeNode(13);

    root->right->right->left = new TreeNode(7);

    printNodesV1(root,0);

    int n1 = 7;

    int n2 = 9;

    int res = lca(root, n1, n2);

    cout << "LCA(" << n1 << "," << n2 << "): " << res << endl;;

    return 0;

}