#include <iostream>

using namespace std;

class node{

public:

int data;

node \*left;

node \*right;

node(int d){

data=d;

left=NULL;

right=NULL;

}

};

node\* buildTree(){

int d;

cin>>d;

if(d==-1)

return NULL;

node \*root=new node(d);

root->left=buildTree();

root->right=buildTree();

return root;

}

int height(node \*root){

if(root==NULL)

return 0;

int ls=height(root->left);

int rs=height(root->right);

return max(ls,rs)+1;

}

int diameter(node \*root){

if(root==NULL)

return 0;

int h1=height(root->left);

int h2=height(root->right);

int op1=h1+h2;

int op2=diameter(root->left);

int op3=diameter(root->right);

return max(op1,max(op2,op3));

}

int main() {

node \*root=buildTree();

cout<<diameter(root); // it will take O(n^2) time

}

Input-

8 16 6 -1 -1 4 -1 -1 3 2 -1 -1 -1

Output-

4

//O(n) time complexity- using post order

#include <iostream>

using namespace std;

class node{

public:

int data;

node \*left;

node \*right;

node(int d){

data=d;

left=NULL;

right=NULL;

}

};

node\* buildTree(){

int d;

cin>>d;

if(d==-1)

return NULL;

node \*root=new node(d);

root->left=buildTree();

root->right=buildTree();

return root;

}

class Pair{

public:

int height;

int diameter;

};

Pair fastdiameter(node \*root){

Pair p;

if(root==NULL){

p.diameter=p.height=0;

return p;

}

//otherwise

Pair left=fastdiameter(root->left);

Pair right=fastdiameter(root->right);

p.height=max(left.height,right.height)+1;

p.diameter=max(left.height+right.height,max(left.diameter,right.diameter));

return p;

}

int main() {

node \*root=buildTree();

Pair p=fastdiameter(root);

cout<<p.height<<"\n";

cout<<p.diameter;

}