/\*

The height of a node is the number of edges in

its longest chain of descendants.

Implement computeHeight to compute the height

of the subtree rooted at the node n. Note that

this function does not return a value. You should

store the calculated height in that node's own

height member variable. Your function should also

do the same for EVERY node in the subtree rooted

at the current node. (This naturally lends itself

to a recursive solution!)

Assume that the following includes have already been

provided. You should not need any other includes

than these.

#include <cstdio>

#include <cstdlib>

#include <iostream>

#include <string>

You have also the following class Node already defined.

You cannot change this class definition, so it is

shown here in a comment for your reference only:

class Node {

public:

int height; // to be set by computeHeight()

Node \*left, \*right;

Node() { height = -1; left = right = nullptr; }

~Node() {

delete left;

left = nullptr;

delete right;

right = nullptr;

}

};

\*/

void computeHeight(Node \*n) {

if (n == nullptr) {

return;

}

computeHeight(n->left);

computeHeight(n->right);

int leftHeight = -1;

int rightHeight = -1;

if (n->left != nullptr) {

leftHeight = n->left->height;

}

if (n->right != nullptr) {

rightHeight = n->right->height;

}

n->height = 1 + std::max(leftHeight, rightHeight);

// Implement computeHeight() here.

}

// This function prints the tree in a nested linear format.

void printTree(const Node \*n) {

if (!n) return;

std::cout << n->height << "(";

printTree(n->left);

std::cout << ")(";

printTree(n->right);

std::cout << ")";

}

// The printTreeVertical function gives you a verbose,

// vertical printout of the tree, where the leftmost nodes

// are displayed highest. This function has already been

// defined in some hidden code.

// It has this function prototype: void printTreeVertical(const Node\* n);

// This main() function is for your personal testing with

// the Run button. When you're ready, click Submit to have

// your work tested and graded.

int main() {

Node \*n = new Node();

n->left = new Node();

n->right = new Node();

n->right->left = new Node();

n->right->right = new Node();

n->right->right->right = new Node();

computeHeight(n);

printTree(n);

std::cout << std::endl << std::endl;

printTreeVertical(n);

// The Node destructor will recursively

// delete its children nodes.

delete n;

n = nullptr;

return 0;

}