/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

You may assume that the following Node class has already

been defined for you previously:

class Node {

public:

Node \*left, \*right;

Node() { left = right = nullptr; }

~Node() {

delete left;

left = nullptr;

delete right;

right = nullptr;

}

};

You may also assume that iostream has already been included.

Implement the function "int count(Node \*n)" below to return

an integer representing the number of nodes in the subtree

of Node n (including Node n itself).

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int count(Node \*n) {

if (n==nullptr){

return 0;

}

else{

int num=1;

num+=count(n->left);

num+=count(n->right);

return num;

}

// Implement count() here.

}

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();

// This should print a count of six nodes

std::cout << count(n) << std::endl;

// Deleting n is sufficient to delete the entire tree

// because this will trigger the recursively-defined

// destructor of the Node class.

delete n;

n = nullptr;

return 0;

}