#include <iostream>

#include <algorithm>

#include <climits>

struct TreeNode {

int val;

TreeNode \*left;

TreeNode \*right;

TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

};

class Solution {

public:

int maxPathSum(TreeNode\* root) {

int max\_sum = INT\_MIN; // Initialize to minimum integer value

dfs(root, max\_sum);

return max\_sum;

}

private:

int dfs(TreeNode\* node, int& max\_sum) {

if (!node) return 0;

// Recursively get maximum path sums from left and right subtrees

int left\_max = std::max(dfs(node->left, max\_sum), 0); // Ignore negative sums

int right\_max = std::max(dfs(node->right, max\_sum), 0);

// Update the maximum path sum found so far

max\_sum = std::max(max\_sum, node->val + left\_max + right\_max);

// Return the maximum path sum extending from this node

return node->val + std::max(left\_max, right\_max);

}

};

// Example usage

int main() {

TreeNode\* root = new TreeNode(-10);

root->left = new TreeNode(9);

root->right = new TreeNode(20);

root->right->left = new TreeNode(15);

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

Solution solution;

std::cout << "Maximum Path Sum: " << solution.maxPathSum(root) << std::endl; // Output: 42

return 0;

}