

## K-th Level

Given a binary tree with N nodes. Your task is to print its Kth level.

### **Input Format**

In the function a pointer to the root node of the binary tree is passed.

### **Output Format**

Return a vector containing nodes at Kth level

Input:   
 --- Level 1            /   \        60            --- Level 0            /   \            50        30  
 --- Level 2            /   \        80        10    40            --- Level 2 K = 10    output: 30 50

**Solution:** kthLevel.cpp

## Sum of Nodes

Given a binary tree with N nodes. Your task is to return the sum of all N nodes.

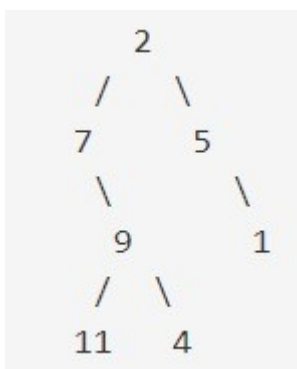
### **Input Format**

In the function a pointer to the root node of the binary tree is passed.

### **Output Format**

Return a integer representing sum of all nodes

### **Sample Input**



### **Sample Output**

39

**Solution:** nodeSum.cpp

## Min Depth

Given a binary tree with N nodes. Your task is to find the minimum depth.

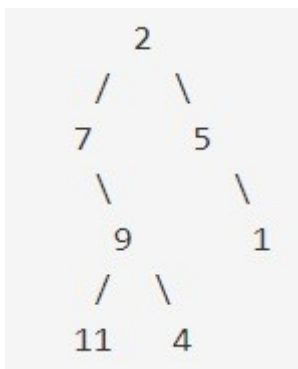
### **Input Format**

In the function a pointer to the root node of the binary tree is passed.

### **Output Format**

Return an integer representing minimum depth

### **Sample Input**



### **Sample Output**

3

### **Explanation**

Minimum depth is between nodes 2 and 1 since minimum depth is defined as the number of nodes along the shortest path from the root node down to the nearest leaf node.

**Solution:** minDepth.cpp

## Symmetric Tree

Given a binary tree with N nodes. Your task is to check whether it is a mirror of itself (i.e., symmetric around its center).

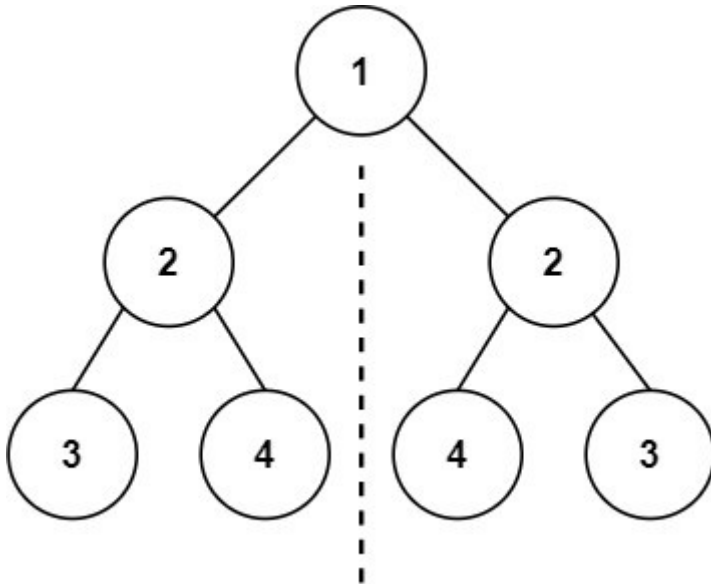
### **Input Format**

In the function a pointer to the root node of the binary tree is passed.

### **Output Format**

Return **true** if symmetric otherwise return **false**.

### **Sample Input**



### Sample Output

True

**Solution:** symmetricTree.cpp

## Expression Tree

Given a full binary expression tree consisting of basic binary operators (+, −, \*, /) and some integers in the form of **string**, Your task is to evaluate the expression tree.

Note: All the nodes' data are in the form of string.

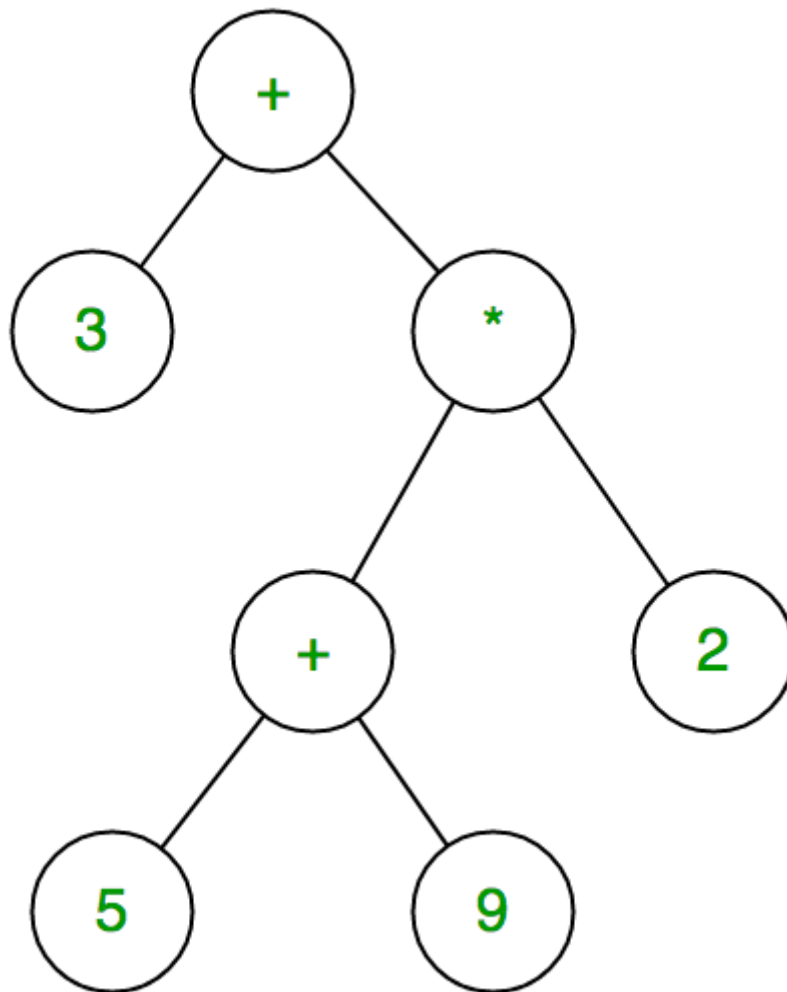
### Input Format

In the function a pointer to the root node of the binary tree is passed.

### Output Format

Return an integer representing the final value of the expression.

### Sample Input



### Sample Output

31

### Explanation

$((5 + 9) * 2) + 3 = 31$

**Solution:** expressionTree.cpp

### Remove Half Nodes

Given A binary Tree. Your task is to remove all the half nodes (which has only one child) and return the INORDER traversal of the updated binary tree.

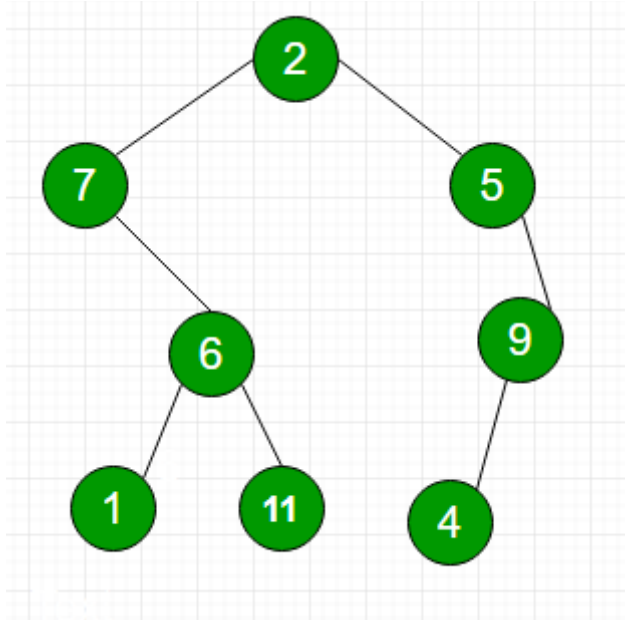
Input Format

In the function a pointer to the root node of the binary tree is passed.

Output Format

Return an integer vector containing INORDER traversal of the updated binary tree.

Sample Input



Sample Output

INORDER traversal of the **new** tree : 1 6 11 2 4

Explanation

Nodes 7, 5 and 9 are half nodes as one of their child is Null.

**Solution:** removeHalfNode.cpp

## Target Path Sum

Given the **root** of a binary tree and an integer **targetSum**, return all root-to-leaf paths where each path's sum equals **targetSum**.

Return an empty vector if no such path exists.

Input Format

In the function a pointer to the root node of the binary tree and target sum is passed.

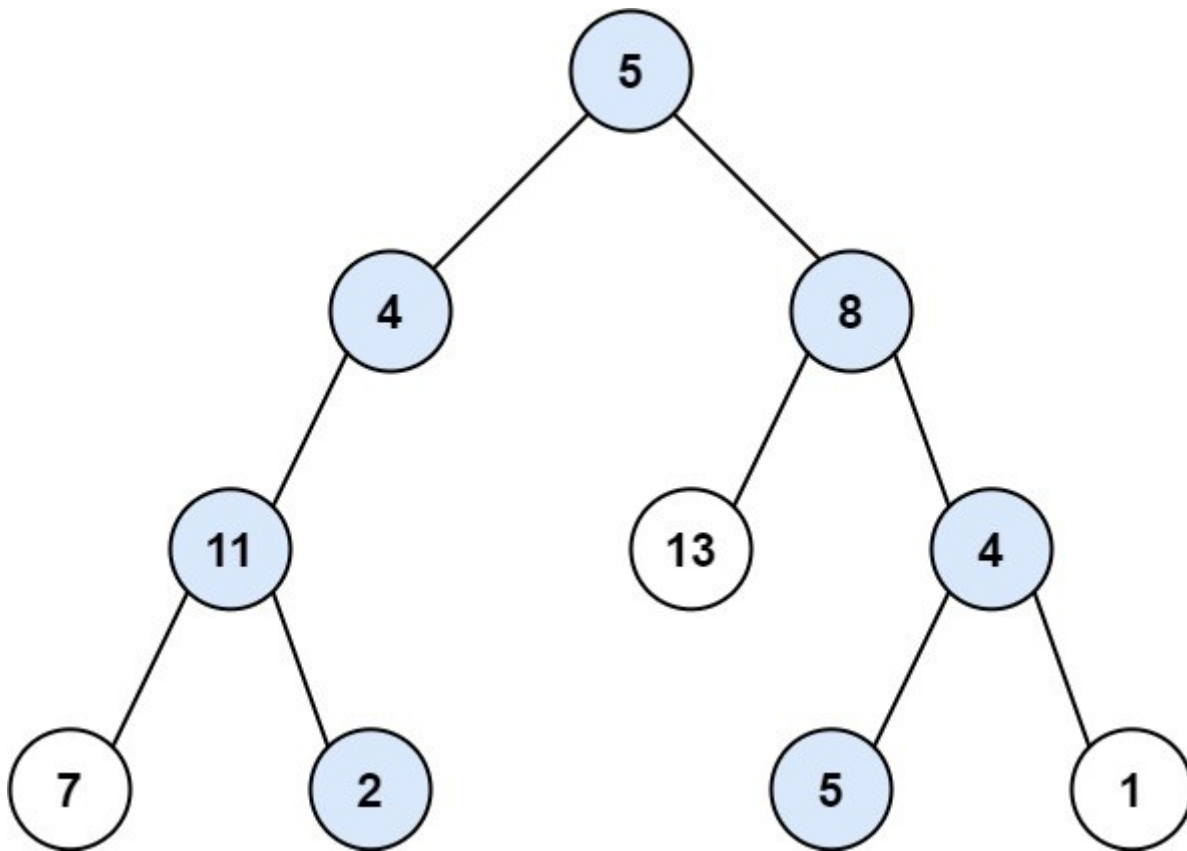
Output Format

Return a vector of vectors containing all those root to leaf paths.

Sample Input

Target sum: 22

Tree:



Sample Output

[ [5, 4, 11, 2] , [5, 8, 4, 5] ]

**Solution:** targetPathSum.cpp