Problem Description

Given a binary tree and a sum, find the number of root-to-leaf paths where each path's sum equals the given sum.
Note: A leaf is a node with no children.

Input format

Line1: Number of Test cases
Line2 to X: First Test Case details with first set of lines giving the binary tree structure (refer section below for the format) and last line giving the Sum.
LineX+1 to Y: Second Test case details and so on.

Output format

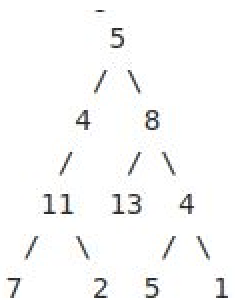
Number of paths from root to leaf whose path-sum is equal to the given Sum.
One line for each Test case specified.

Constraints

1<= value of nodes <=10000
1<= number of nodes <=100000

Sample Input 1

Tree given with root as 5.



```
        5
       / \
      4   8
     /   / \
    11  13  4
   /  \    / \
  7    2  5   1
```

Sum is 22

Sample Output 1

2

Explanation 1

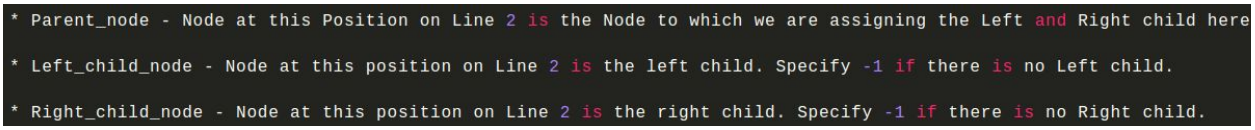
There are two paths from root to leaf whose sum is equal to 22:
[5,4,11,2]
[5,8,4,5]

Instructions to create custom input for a Binary Tree

In order to specify a binary tree that can be used as custom input to the problem, you'll need to follow this convention.

- Line 1: Number of nodes in the Binary Tree (N)
- Line 2: N space separated node values. The position of the Nodes on this line will be used to refer to them in the below lines, starting from 1.
- Line 3 to N+2: Lines specifying the child nodes for each of the N nodes

Format of each line (space separated): Parent_node Left_child_node Right_child_node

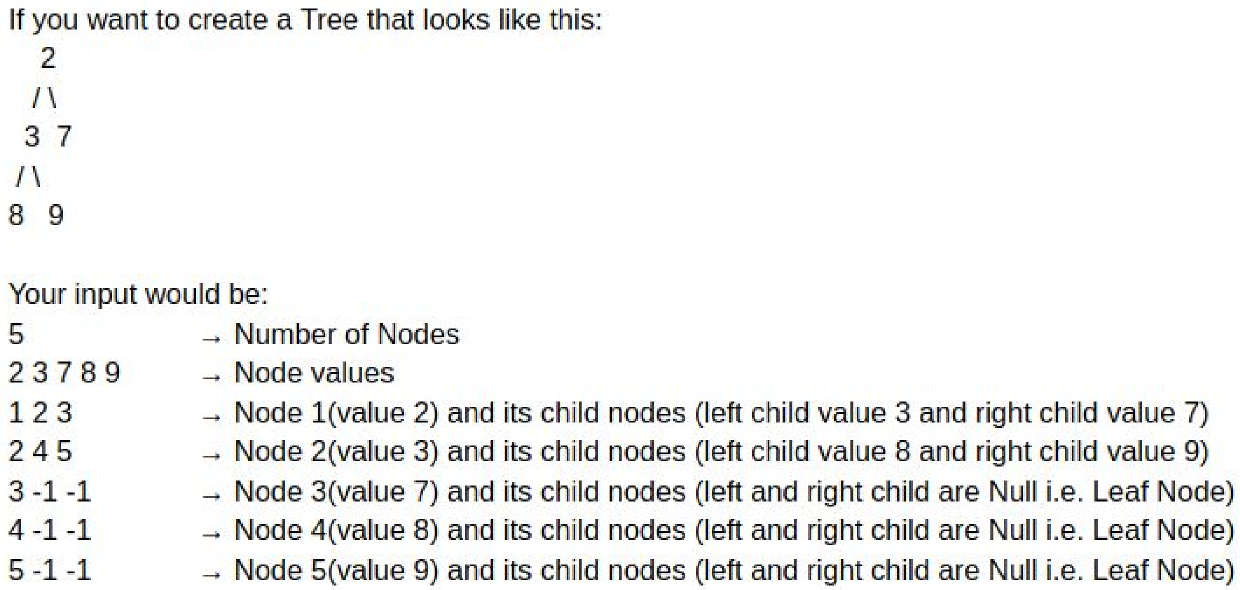


```
* Parent_node - Node at this Position on Line 2 is the Node to which we are assigning the Left and Right child here

* Left_child_node - Node at this position on Line 2 is the left child. Specify -1 if there is no Left child.

* Right_child_node - Node at this position on Line 2 is the right child. Specify -1 if there is no Right child.
```

Example1



If you want to create a Tree that looks like this:

```
   2
  / \
 3   7
/ \
8   9
```

Your input would be:

5 → Number of Nodes
2 3 7 8 9 → Node values
1 2 3 → Node 1(value 2) and its child nodes (left child value 3 and right child value 7)
2 4 5 → Node 2(value 3) and its child nodes (left child value 8 and right child value 9)
3 -1 -1 → Node 3(value 7) and its child nodes (left and right child are Null i.e. Leaf Node)
4 -1 -1 → Node 4(value 8) and its child nodes (left and right child are Null i.e. Leaf Node)
5 -1 -1 → Node 5(value 9) and its child nodes (left and right child are Null i.e. Leaf Node)