**Height of Binary Tree: -**

**Easy** Accuracy: **78.58%** Submissions: **239K+** Points: **2**

Given a binary tree, find its height.

**Example 1:**

**Input:**

1

/ \

2 3

**Output:** 2

**Example 2:**

**Input:**

2

\

1

/

3

**Output:** 3

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function **height()**which takes root node of the tree as input parameter and returns an integer denoting the height of the tree. If the tree is empty, return 0.

**Expected Time Complexity:** O(N)  
**Expected Auxiliary Space:** O(N)

**Constraints:**  
1 <= Number of nodes <= 105  
1 <= Data of a node <= 109

**Code: -**

**//{ Driver Code Starts**

**//Initial template for C++**

**#include <bits/stdc++.h>**

**using namespace std;**

**struct Node**

**{**

**int data;**

**struct Node \*left;**

**struct Node \*right;**

**Node(int val) {**

**data = val;**

**left = right = NULL;**

**}**

**};**

**// Function to Build Tree**

**Node\* buildTree(string str)**

**{**

**// Corner Case**

**if(str.length() == 0 || str[0] == 'N')**

**return NULL;**

**// Creating vector of strings from input**

**// string after spliting by space**

**vector<string> ip;**

**istringstream iss(str);**

**for(string str; iss >> str; )**

**ip.push\_back(str);**

**// Create the root of the tree**

**Node \*root = new Node(stoi(ip[0]));**

**// Push the root to the queue**

**queue<Node\*> queue;**

**queue.push(root);**

**// Starting from the second element**

**int i = 1;**

**while(!queue.empty() && i < ip.size()) {**

**// Get and remove the front of the queue**

**Node\* currNode = queue.front();**

**queue.pop();**

**// Get the current node's value from the string**

**string currVal = ip[i];**

**// If the left child is not null**

**if(currVal != "N") {**

**// Create the left child for the current Node**

**currNode->left = new Node(stoi(currVal));**

**// Push it to the queue**

**queue.push(currNode->left);**

**}**

**// For the right child**

**i++;**

**if(i >= ip.size())**

**break;**

**currVal = ip[i];**

**// If the right child is not null**

**if(currVal != "N") {**

**// Create the right child for the current node**

**currNode->right = new Node(stoi(currVal));**

**// Push it to the queue**

**queue.push(currNode->right);**

**}**

**i++;**

**}**

**return root;**

**}**

**// } Driver Code Ends**

**//User function template for C++**

**/\***

**struct Node**

**{**

**int data;**

**struct Node\* left;**

**struct Node\* right;**

**Node(int x){**

**data = x;**

**left = right = NULL;**

**}**

**};**

**\*/**

**class Solution{**

**public:**

**//Function to find the height of a binary tree.**

**int height(struct Node\* node){**

**// code here**

**if(!node)**

**return 0;**

**return 1+max(height(node->left), height(node->right));**

**}**

**};**

**//{ Driver Code Starts.**

**int main()**

**{**

**int t;**

**scanf("%d ",&t);**

**while(t--)**

**{**

**string treeString;**

**getline(cin,treeString);**

**Node\* root = buildTree(treeString);**

**Solution ob;**

**cout<<ob.height(root)<<endl;**

**}**

**return 0;**

**}**

**// } Driver Code Ends**

**T.C: - O(N)**

**S.C: - O(N)**