**Left View of Binary Tree**

Given a Binary Tree, return Left view of it. Left view of a Binary Tree is set of nodes visible when tree is visited from Left side. The task is to complete the function **leftView()**, which accepts root of the tree as argument.

Left view of following tree is 1 2 4 8.

          1  
       /     \  
     2        3  
   /     \    /    \  
  4     5   6    7  
   \  
     8

//{ Driver Code Starts

import java.util.LinkedList;

import java.util.Queue;

import java.io.\*;

import java.util.\*;

class Node{

int data;

Node left;

Node right;

Node(int data){

this.data = data;

left=null;

right=null;

}

}

class GfG {

static Node buildTree(String str){

if(str.length()==0 || str.charAt(0)=='N'){

return null;

}

String ip[] = str.split(" ");

// Create the root of the tree

Node root = new Node(Integer.parseInt(ip[0]));

// Push the root to the queue

Queue<Node> queue = new LinkedList<>();

queue.add(root);

// Starting from the second element

int i = 1;

while(queue.size()>0 && i < ip.length) {

// Get and remove the front of the queue

Node currNode = queue.peek();

queue.remove();

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

String currVal = ip[i];

// If the left child is not null

if(!currVal.equals("N")) {

// Create the left child for the current node

currNode.left = new Node(Integer.parseInt(currVal));

// Push it to the queue

queue.add(currNode.left);

}

// For the right child

i++;

if(i >= ip.length)

break;

currVal = ip[i];

// If the right child is not null

if(!currVal.equals("N")) {

// Create the right child for the current node

currNode.right = new Node(Integer.parseInt(currVal));

// Push it to the queue

queue.add(currNode.right);

}

i++;

}

return root;

}

static void printInorder(Node root)

{

if(root == null)

return;

printInorder(root.left);

System.out.print(root.data+" ");

printInorder(root.right);

}

public static void main (String[] args) throws IOException{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int t=Integer.parseInt(br.readLine());

while(t > 0){

String s = br.readLine();

Node root = buildTree(s);

Tree g = new Tree();

ArrayList<Integer> result = g.leftView(root);

for(int value : result){

System.out.print(value + " ");

}

System.out.println();

t--;

}

}

}

// } Driver Code Ends

//User function Template for Java

/\* A Binary Tree node

class Node

{

int data;

Node left, right;

Node(int item)

{

data = item;

left = right = null;

}

}\*/

class Tree

{

//Function to return list containing elements of left view of binary tree.

int height(Node node)

{

// code here

if(node==null)

return 0;

return (int)Math.max(height(node.left)+1,height(node.right)+1);

}

void solve(Node node,ArrayList<Integer> sol,int level){

if(node==null){

return;

}

else{

if(sol.get(level)==0){

sol.add(level,node.data);

}

solve(node.left,sol,level+1);

solve(node.right,sol,level+1);

}

}

ArrayList<Integer> leftView(Node root)

{

// Your code here

ArrayList<Integer> solution=new ArrayList<>();

int length=height(root);

for(int i=0;i<length;i++){

solution.add(0);

}

if(root!=null)

solve(root,solution,0);

ArrayList<Integer> result=new ArrayList<>();

for(int i=0;i<length && solution.get(i)!=0;i++)

result.add(solution.get(i));

return result;

}

}