/\*

Problem Statement:

-------------------------

Bheemla is working with Binary Trees. The elements of the tree is given in the level order format.

Bheemla likes to print the nodes of the tree line by line.

You will be given the root of the binary tree.Your task is to help Bheemla to print the nodes according to his wish. Look at the hint for understanding.

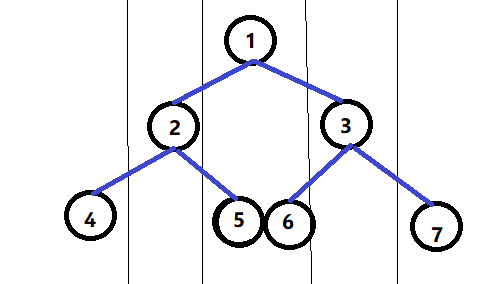
Your task is to implement the class Solution:

- public List<List<Integer>> printTheLines(BinaryTreeNode root):

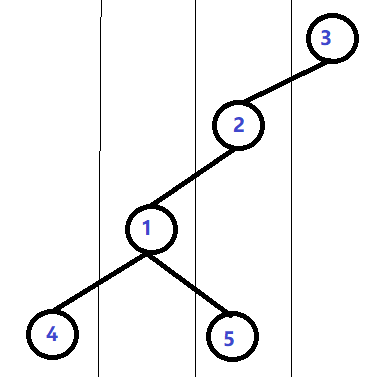
return the list of node values.

NOTE: Please do consider the node with data '-1' as null node in the given trees.

Example-1:



Example-2:



Input Format:

------------------

Space separated integers, elements of the tree.

Output Format:

--------------------

Print the list of list of integers

Sample Input-1:

--------------------

1 2 3 4 5 6 7

Sample Output-1:

----------------

[[4],[2],[1,5,6],[3],[7]]

Sample Input-2:

---------------

3 2 -1 1 -1 -1 -1 4 5

Sample Output-2:

----------------

[[4],[1],[2,5],[3]]

Test Cases:

---------------

case =1

input =1 2 3 4 5 6 7

output =[[4],[2],[1,5,6],[3],[7]]

case =2

input =3 2 -1 1 -1 -1 -1 4 5

output =[[4],[1],[2,5],[3]]

case =3

input =1 2 3 4 5 -1 7 8 9 10 11 -1 -1 -1 -1 16 17 18 19 20 21 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 33 -1 35 36

output =[[32],[16],[8,33,36],[4,17,18,20],[2,9,10,35],[1,5,19,21],[3,11],[7]]

case =4

input =7 4 3 5 6 2 3 8 4 5 6 7 5 5 6

output =[[8],[5],[4,4,5,7],[7,6,2],[3,6,5,5],[3],[6]]

case =5

input =7 4 3 5 6 2 -1 8 4 5 6 7

output =[[8],[5],[4,4,5,7],[7,6,2],[3,6]]

case =6

input =1 2 3 4 5 6 7 8 9 -1 -1 -1 -1 10 11

output =[[8],[4],[2,9],[1,5,6],[3,10],[7],[11]]

case =7

input =1 2 3 4 5 6 -1 12 14 13 -1 -1 15

output =[[12],[4],[2,14,13],[1,5,6],[3,15]]

case =8

input =1 2 3 4 5 6 7 15 14 13 12 10 11 9 7

output =[[15],[4],[2,14,13,10],[1,5,6],[3,12,11,9],[7],[7]]

\*/

Solution:

------------

import java.util.\*;

import java.util.\*;

class BinaryTreeNode{

public int data;

public BinaryTreeNode left, right;

public BinaryTreeNode(int data){

this.data = data;

left = null;

right = null;

}

}

public class RightSideTree {

static BinaryTreeNode root;

static BinaryTreeNode temp = root;

void insert(BinaryTreeNode temp, int key)

{

if (temp == null) {

root = new BinaryTreeNode(key);

return;

}

Queue<BinaryTreeNode> q = new LinkedList<BinaryTreeNode>();

q.add(temp);

// Do level order traversal until we find

// an empty place.

while (!q.isEmpty()) {

temp = q.peek();

q.remove();

if (temp.left == null) {

temp.left = new BinaryTreeNode(key);

break;

}

else

q.add(temp.left);

if (temp.right == null) {

temp.right = new BinaryTreeNode(key);

break;

}

else

q.add(temp.right);

}

}

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

String str[]=sc.nextLine().split(" ");

RightSideTree bt=new RightSideTree();

root=new BinaryTreeNode(Integer.parseInt(str[0]));

for(int i=1; i<str.length; i++)

bt.insert(root,Integer.parseInt(str[i]));

Solution sol= new Solution();

System.out.println(sol.printTheLines(root));

}

}

class Solution {

public List<List<Integer>> printTheLines(BinaryTreeNode root) {

Map<Integer,List<Integer>> mp = new TreeMap<>();

helper(root, mp,0);

List<List<Integer>> ans = new ArrayList<>();

for(int x: mp.keySet()){

// Collections.sort(mp.get(x));

ans.add(mp.get(x));

}

return ans;

}

static void helper(BinaryTreeNode current, Map<Integer,List<Integer>> mp, int pos){

LinkedList<BinaryTreeNode> x = new LinkedList<>();

LinkedList<Integer> y = new LinkedList<>();

x.addLast(current);

y.addLast(0);

while(!x.isEmpty()){

BinaryTreeNode c = x.getFirst();

int p = y.getFirst();

x.removeFirst();

y.removeFirst();

if(c==null || c.data==-1)continue;

if(!mp.containsKey(p)){

mp.put(p, new LinkedList<>());

}

mp.get(p).add(c.data);

x.addLast(c.left);

y.addLast(p-1);

x.addLast(c.right);

y.addLast(p+1);

}

}

}