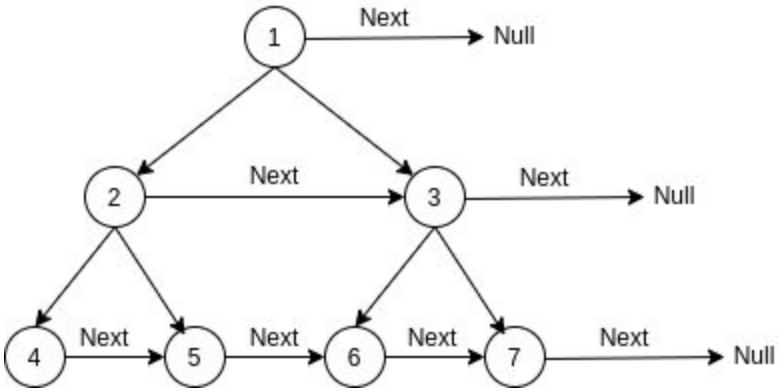
Problem Description

You are given a perfect binary tree where all leaves are on the same level, and every parent has two children. The binary tree nodes have the following definition:

```
public class TreeNode {
public long val;
public TreeNode left;
public TreeNode right;
public TreeNode next;
}
```

Populate each next pointer to point to its next right node. If there is no next right node, the next pointer should be set to NULL.
Initially, all next pointers are set to NULL.



Input format

Line 1: Number of Test cases (T)
Line 2 to X: First Test Case binary tree structure (refer section below for the format)
Line X+1 to Y: Second Test Case details and so on.

Output format

For each test case, print N (number of nodes in the tree) space separated values of next nodes on a separate line. If nothing exists in the next print 'null' without quotes.

Constraints

1 <= T <= 100
1 <= N <= 100000
0 <= Value of nodes <= 10^9

It is guaranteed that the sum of the number of tree nodes across all test cases will be less than 500000.

Sample Input 1

1
7
1 2 3 4 5 6 7
1 2 3
2 4 5
3 6 7
4 -1 -1
5 -1 -1
6 -1 -1
7 -1 -1

Sample Output 1

null 3 null 5 6 7 null

Explanation 1

Refer to the diagram above.
1 does not have any nodes on its right so it is pointing to null.
2 has 3 as its next node.
3 does not have any nodes on its right so it is pointing to null.
4 has 5 as its next node.
5 has 6 as its next node.
6 has 7 as its next node.
7 does not have any nodes on its right so it is pointing to null

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

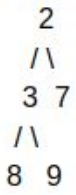
```
* Parent_node - Node at this Position on Line 2 is the Node to which we are assigning the Left and Right child here

* 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:



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)