Oh, My Trees!

Background

Pablito likes trees of all kinds: palm trees, pine trees, orange trees, eggplants, etc. But his favorites are binary search trees.

As you should know, **binary search trees** (BST) are binary trees (i.e., each node of the tree can have zero, one or two subtrees, which are called *left* and *right* subtrees) with the following property:

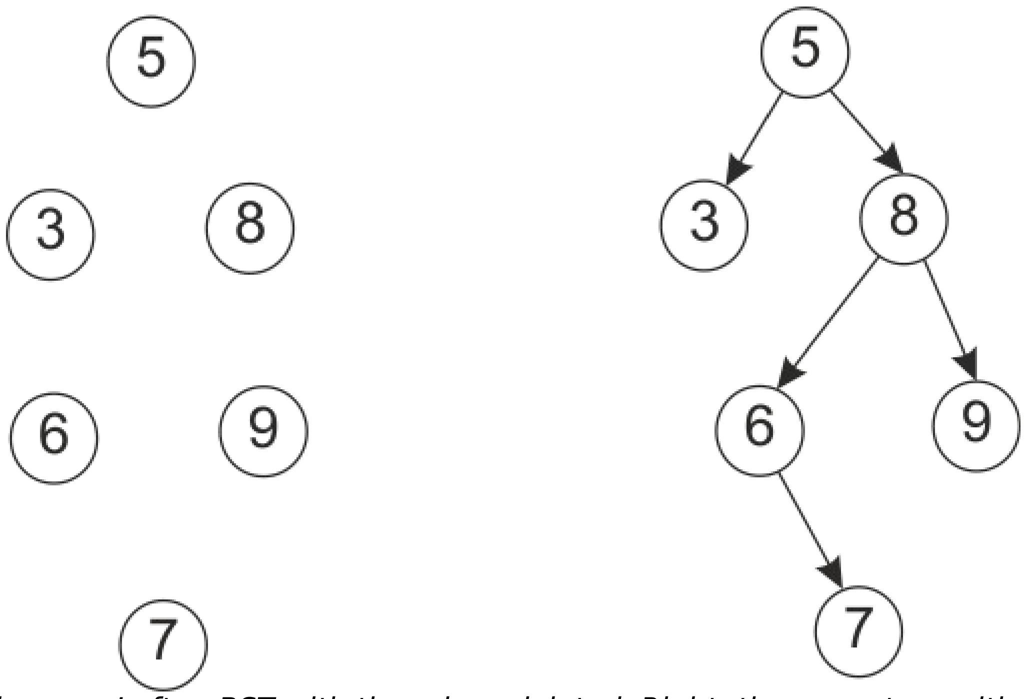
- The left subtree of a node contains only nodes with keys less than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.

Pablito is very proud of his huge collection of BS trees. However, he has a younger brother who loves deleting trees. Poor thing! He is only 2 years old!

Pablito's brother has the ability of deleting only the edges of the trees, but not the nodes. Can you help Pablito to fix up the disaster made by his brother?

The Problem

We have a set of different integer values, which are the keys of a BST. But the edges of the BST have been deleted; we know the level (or *depth*) of each key in the tree, but we do not know the father-child relationships.



A sample case. Left: a BST with the edges deleted. Right: the same tree with the left-child / right-child relationships undeleted.

Your task is to recover the right-child / left-child relationships for all nodes.

Input

The input can contain different test cases. The first line of the input indicates the number of test cases.

For each test case, the first line contains an integer, *H*, which is the total hight of the tree (i.e., the number of levels). Then, there are *H* lines, each of them with a set of integers separated by spaces. Each line contains the keys of each level of the BST. You can assume that the description of the tree is always valid; keys are not repeated.

For example, the previous tree would be represented as:

```
4
5
3 8
6 9
7
```

Output

For each test case, you have to output each key in a line, in the same order as they appear in the input. For eack key, you have to indicate his children in the following format:

```
node:leftChild-rightChild
```

If the node does not have left or right child, no number is printed. For example, in the previous three the result should be:

```
5:3-8
3:-
8:6-9
6:-7
9:-
7:-
```

Sample Input

```
3
4
5
3 8
6 9
7
5
23
59
35
39
36
8
10
4 12
1 6 11 31
0 3 5 8 51
34
49
40
38 45
```

Sample Output

```
5:3-8
3:-
8:6-9
6:-7
9:-
7:-
23:-59
59:35-
35:-39
39:36-
36:-
10:4-12
4:1-6
12:11-31
1:0-3
6:5-8
11:-
31:-51
0:-
3:-
5:-
8:-
51:34-
34:-49
49:40-
40:38-45
38:-
45:-
```