Problem Description

Given a Binary Search Tree, generate all the sequences which would create the same binary search tree.

Input format

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

Output format

On Line 1, print X, the number of valid sequences
On the next X lines print N space separated integers, where N is the number of nodes in the input tree.

Constraints

1 <= T <= 100
1 <= N <= 15
0 <= Node values <= 10^9

Sample Input 1

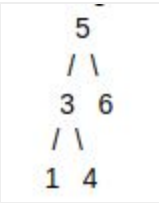
```
1
5
5 3 6 1 4
1 2 3
2 4 5
3 -1 -1
4 -1 -1
5 -1 -1
```

Sample Output 1

```
8
5 3 1 4 6
5 3 1 6 4
5 3 6 1 4
5 6 3 1 4
5 3 4 1 6
5 3 4 6 1
5 3 6 4 1
5 6 3 4 1
```

Explanation 1

The tree is given below and it can be constructed from any of the sequences given in the output



Different number sequences can lead to the same BST. For each number sequence, we start inserting them into an empty BST one by one in order. The first number becomes the root node and the next element becomes the root's right or left child depending on whether it is greater or lesser than the value at root. Each subsequent number gets inserted into the BST depending on the previous nodes inserted into the BST. Here we see that all these sequences produce the same BST when we create a BST from scratch.

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

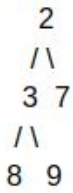
```
* Parent_node - Node at this Position on Line 2 is the Node to which we are assigning the Left and Right child here

* 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)