Problem H: Legacy Code

Once again you lost days refactoring code, which never runs in the first place. Enough is enough – your time is better spent writing a tool that finds unused code!

Your software is divided into packages and executables. A package is a collection of methods. Executables are packages defining among other methods exactly one method with name PROGRAM. This method is executed on the start of the corresponding executable. Ordinary packages have no method named PROGRAM.

Each method is uniquely identified by the combination of package and method names. E.g. the method with the identifier SuperGame::PROGRAM would be the main method of the executable SuperGame.

For every method in your software you are given a list of methods directly invoking it. Thus you can easily identify methods, that are never called from any method. However, your task is more challenging: you have to find unused methods. These are methods that are never reached by the control flow of any executable in your software.

Input

The first line of the input contains an integer N, the number of methods in your software $(1 \le N \le 400)$.

Each method is described by two lines, totaling in $2 \cdot N$ lines. The first line consists of the unique identifier of the method and k_i , the number of methods directly invoking this one $(0 \le k_i \le N)$. The second line consists of a set of k_i identifiers of these calling methods or is empty if there are no such methods, i.e. $k_i = 0$.

Method identifiers consist of a package name followed by two colons and a method name like Packagename: :Methodname. Both strings, the package and the method name, each consist of up to 20 lowercase, uppercase characters or digits (a-z, A-Z, 0-9).

There will be exactly N different method identifiers mentioned in the input.

Output

A line containing the number of unused methods in your software.

Sample Input 1

Sample Output 1

2 0
SuperGame::PROGRAM 0

HelpPackage::HelpFunction 2

HelpPackage::HelpFunction SuperGame::PROGRAM

Sample Input 2

Sample Output 2

2
Loop::CallA 1

Loop::CallB 1
Loop::CallA

Loop::CallB

Sample Input 3

Sample Output 3

0

2

SuperGame::PROGRAM 1
SuperServer42::PROGRAM
SuperServer42::PROGRAM 1
SuperServer42::PROGRAM