A - Alien DNA

Source file name: `alien.py`
Time limit: 1 second

Alien DNA is more complicated than human DNA. For example, rather than viewing an alien DNA strand as a sequence of base pairs, it can be viewed as a sequence of base sets. A base set is, simply put, a string of lowercase letters where each letter represents a different base. Note that human DNA is also a sequence of base sets, except a base set is simply a pairing of *a* with *t* or *c* with *g*.

Since alien DNA is more complicated than human DNA, the algorithms used in alien bioinformatics are also much more complicated. However, you have only been working at AlienBioTechNextGenCorp for three days and are not yet trusted enough to handle the complicated tasks.

Your first job is simple. You must develop software to cut a strand of alien DNA into as few segments as possible. The only constraint is that base sets of a given segment must share at least one common base.

Input

Input begins with an integer $t \geq 0$, the number of test cases to be processed. Each test case begins with a single value n ($1 \leq n \leq 10\,000$) representing the length of the alien DNA strand. Each of the n following lines consists of a single string of lowercase letters representing a base set. Each string will contain at least one character, and no repeated characters. The base sets are given in the input in the same order they appear on the DNA strand.

The input must be read from standard input.

Output

For each test case, output the minimum number of cuts required to partition the strand into segments that share a common base.

The output must be written to standard output.

Sample Input	**Sample Output**
2 5 as sd df fg gh 3 plum orange plum	2 2

B - Bad Code

Source file name: `code.py`
Time limit: 2 seconds

In his endless attempt to keep his secret documents safe, Bob has designed yet another coding system. In this system, every character is replaced by a unique positive integer value – which we call the code for a character. However, this system – as expected – is not the best in the world. So more than one plain text can result in the same encrypted string. Here are a few more notes:

- The document consists exclusively of lower-case letters.
- The code for a letter is no more than 99.
- The encrypted string does not contain more than 100 characters.
- A code may or may not be preceded by a 0 in the encrypted string (note the 2nd sample test case).

Given the codes for each character and the encrypted string, you have to find all the plain text strings in alphabetical order that produces that encrypted string.

Input

There will be several test cases. Each test case starts with an integer N, which gives the number of unique characters in the document. Each of next N lines contains a character in the letter and its code. The last line in the test case gives the encrypted string. The last test case will have $N = 0$, which will need not be processed.

The input must be read from standard input.

Output

For each test case, print the test case number and the possible plain texts for the given encrypted string. If there is more than 100 possible strings report only the first 100 strings. Print a blank line after the output for each test case.

The output must be written to standard output.

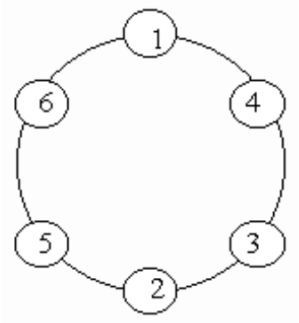
Sample Input	**Sample Output**
5 a 12 b 1 c 2 d 3 e 23 123 2 o 10 x 1 1010101 0	Case #1 ad bcd be Case #2 ooox ooxx oxox oxxx xoox xoxx xxox xxxx

C - Prime Ring Problem

Source file name: `prime.py`
Time limit: 3 seconds

A ring is composed of n (even number) circles as shown in diagram. Put natural numbers $1, 2, \ldots, n$ into each circle separately and the sum of numbers in two adjacent circles should be a prime. The number of first circle is always 1.

You are to write a program that generates the rings of length n.



Input

The input consists of several test cases. Each test case consists of a line containing an even integer number n ($0 < n \leq 16$).

The input must be read from standard input.

Output

For each test case, output all the rings of length n that satisfy the conditions above, each in a single line. Two consecutive numbers are to be separated by a single blank. Order the rings in ascending lexicographical order.

See the sample output for details.

The output must be written to standard output.

Sample Input	**Sample Output**
6 8	Case 1: 1 4 3 2 5 6 1 6 5 2 3 4 Case 2: 1 2 3 8 5 6 7 4 1 2 5 8 3 4 7 6 1 4 7 6 5 8 3 2 1 6 7 4 3 8 5 2

D - Marbles on a Tree

Source file name: `tree.py`
Time limit: 1 second

n boxes are placed on the vertices of a rooted tree, which are numbered from 1 to n, $1 \leq n \leq 10000$. Each box is either empty or contains a number of marbles; the total number of marbles is n.

The task is to move the marbles such that each box contains exactly one marble. This is to be accomplished by a sequence of moves; each move consists of moving one marble to a box at an adjacent vertex. What is the minimum number of moves required to achieve the goal?

Input

The input contains a number of cases. Each case starts with the number n followed by n lines. Each line contains at least three numbers which are: v the number of a vertex, followed by the number of marbles originally placed at vertex v followed by a number d which is the number of children of v, followed by d numbers giving the identities of the children of v.

The input is terminated by a case where $n = 0$ and this case should not be processed.

The input must be read from standard input.

Output

For each case in the input, output the smallest number of moves of marbles resulting in one marble at each vertex of the tree.

The output must be written to standard output.

Sample Input	**Sample Output**
9 1 2 3 2 3 4 2 1 0 3 0 2 5 6 4 1 3 7 8 9 5 3 0 6 0 0 7 0 0 8 2 0 9 0 0 9 1 0 3 2 3 4 2 0 0 3 0 2 5 6 4 9 3 7 8 9 5 0 0 6 0 0 7 0 0 8 0 0 9 0 0 9 1 0 3 2 3 4 2 9 0 3 0 2 5 6 4 0 3 7 8 9 5 0 0 6 0 0 7 0 0 8 0 0 9 0 0 0	7 14 20