A - Rank the Languages

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

You might have noticed that English and Spanish are spoken in many areas all over the world. Now it would be nice to rank all languages according to the number of states where they are spoken. You're given a map which shows the states and the languages where they are spoken. Look at the following map:

```
ttuuttdd
ttuuttdd
uuttuudd
uuttuudd
```

Every letter stands for a language and states are defined as connected areas with the same letter. Two letters are *connected* if one is at left, at right, above or below the other one. In the above map, there are three states where the language `t` is spoken, three where `u` is spoken, and one where people speak `d`.

Your job is to determine the number of states for each language and print the results in a certain order.

Input

The first line contains the number of test cases N. Each test case consists of a line with two numbers H and W, which are the height and the width of the map. Then follow H lines with a string of W letters. Those letters will only be lowercase letters from '`a`' to '`z`'.

The input must be read from standard input.

Output

For each test case print '`World #n`', where n is the number of the test case. After that print a line for each language that appears in the test case, which contains the language, a colon, a space and the number of states, where that language is spoken. These lines have to be ordered decreasingly by the number of states. If two languages are spoken in the same number of states, they have to appear alphabetically.

The output must be written to standard output.

Sample Input	**Sample Output**
2 4 8 ttuuttdd ttuuttdd uuttuudd uuttuudd 9 9 bbbbbbbbb aaaaaaaab bbbbbbbab baaaaacab baccccab bacbbbcab baccccab baaaaaaab bbbbbbbbb	World #1 t: 3 u: 3 d: 1 World #2 b: 2 a: 1 c: 1

B - Vertex

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

A directed graph is represented by n vertices where $1 \leq n \leq 100$, numbered consecutively $1, 2, \ldots, n$, and a series of edges $p \rightarrow q$ which connect the pair of nodes p and q in one direction only.

A vertex r is *reachable* from a vertex p if there is an edge $p \rightarrow r$, or if there exists some vertex q for which q is reachable from p and r is reachable from q. A vertex r is *inaccessible* from a vertex p if r is not reachable from p.

Write a program that searches a directed graph for vertices which are inaccessible from a given starting vertex.

Input

The input data for this program consists of several directed graphs and starting nodes. For each graph, there is first one line containing a single integer n, the number of vertices in the graph. Following, there will be a group of lines, each containing a set of integers. The group is terminated by a line which contains only the integer 0. Each set represent a collection of edges. The first integer in the set, i, is the starting vertex, while the next group of integers, $j \cdots k$, define the series of edges $i \rightarrow j, \cdots, i \rightarrow k$, and the last integer on the line is always 0. Each possible start vertex i, $1 \leq i \leq n$ will appear once or not at all. Following each graph definition, there will be a one line containing list of integers. The first integer on the line will specify how many integers follow. Each of the following integers represents a start vertex to be investigated by your program. The next graph then follows. If there are no more graphs, the next line of the input will contain only the integer 0.

The input must be read from standard input.

Output

For each start vertex to be investigated, your program should identify all the vertices which are inaccessible from the given start vertex. Each list should appear on one line, beginning with the count of inaccessible vertices and followed by the inaccessible vertex numbers.

The output must be written to standard output.

Sample Input	**Sample Output**
3 1 2 0 2 2 0 3 1 2 0 0 2 1 2 0	2 1 3 2 1 3

C - Word Transformation

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

A common word puzzle found in many newspapers and magazines is the word transformation. By taking a starting word and successively altering a single letter to make a new word, one can build a sequence of words which changes the original word to a given end word. For instance, the word "`spice`" can be transformed in four steps to the word "`stock`" according to the following sequence: `spice`, `slice`, `slick`, `stick`, `stock`. Each successive word differs from the previous word in only a single character position while the word length remains the same.

Given a dictionary of words from which to make transformations, plus a list of starting and ending words, you are asked to write a program to determine the number of steps in the shortest possible transformation.

Input

The first line of the input is an integer N, indicating the number of test set that your correct program should test followed by a blank line. Each test set will have two sections. The first section will be the dictionary of available words with one word per line, terminated by a line containing an asterisk ($*$) rather than a word. There can be up to 200 words in the dictionary; all words will be alphabetic and in lower case, and no word will be longer than ten characters. Words can appear in the dictionary in any order.

Following the dictionary are pairs of words, one pair per line, with the words in the pair separated by a single space. These pairs represent the starting and ending words in a transformation. All pairs are guaranteed to have a transformation using the dictionary given. The starting and ending words will appear in the dictionary.

Two consecutive input set will separated by a blank line.

The input must be read from standard input.

Output

The output should contain one line per word pair for each test set, and must include the starting word, the ending word, and the number of steps in the shortest possible transformation, separated by single spaces. Two consecutive output set will be separated by a blank line.

The output must be written to standard output.

Sample Input	**Sample Output**
1 dip lip mad map maple may pad pip pod pop sap sip slice slick spice stick stock * spice stock may pod	spice stock 4 may pod 3

D - Demanding Dilemma

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

A *simple undirected graph* is an ordered pair $G = (V, E)$ with V a non-empty set of vertices and E a set of unordered pairs $\{u, v\}$, where u and v are in V and $u \neq v$. If S is a set, define $|S|$ as the size of S. An *incidence matrix* M for G is a $|V| \times |E|$ matrix where $M(i, j)$ is 1 if edge j is incident to vertex i (i.e., there is some u in V such that $j = \{i, u\}$), and 0 otherwise.

Given an $n \times m$ matrix, can it be an incidence matrix of a simple undirected graph $G = (V, E)$ where $|V| = n$ and $|E| = m$?

Input

The first line of the input contains an integer t, the number of test cases. Each test case starts with a line with two integers n ($1 \leq n \leq 8$) and m ($0 \leq m \leq \frac{n(n-1)}{2}$). Then n lines containing m integers (0's or 1's) each follow such that the j-th number on the i-th line is $M(i, j)$.

The input must be read from standard input.

Output

The output must be written to standard output.

Sample Input	**Sample Output**
3 3 3 1 1 0 0 1 1 1 0 1 3 1 1 1 0 3 3 1 1 0 1 1 1 1 0 0	Yes Yes No