FOLLOWING Grado 11

Order is an important concept in mathematics and in computer science. For example, Zorn’s Lemma states: “a partially ordered set in which every chain has an upper bound contains a maximal element.” Order is also important in reasoning about the fix-point semantics of programs. This problem involves neither Zorn’s Lemma nor fix-point semantics, but does involve order. Given a list of variable constraints of the form x < y, you are to write a program that prints all orderings of the variables that are consistent with the constraints. For example, given the constraints x < y and x < z there are two orderings of the variables x, y, and z that are consistent with these constraints: xyz and xzy.

Input

The input consists of a sequence of constraint specifications. A specification consists of two lines: a list of variables on one line, followed by a list of constraints on the next line.

A constraint is given by a pair of variables, where ‘x y’ indicates that x < y.

All variables are single character, lower-case letters. There will be at least two variables, and no more than 20 variables in a specification. There will be at least one constraint, and no more than 50 constraints in a specification. There will be at least one, and no more than 300 orderings consistent with the constraints in a specification.

Input is terminated by end-of-file.

Output

The output should consist of the final state of the blocks world. Each original block position numbered i (0 ≤ i < n where n is the number of blocks) should appear followed immediately by a colon. If there is at least a block on it, the colon must be followed by one space, followed by a list of blocks that appear stacked in that position with each block number separated from other block numbers by a space. Don’t put any trailing spaces on a line. There should be one line of output for each block position (i.e., n lines of output where n is the integer on the first line of input).

Example

Input:

```
10
move 9 onto 1
move 8 over 1
move 7 over 1
move 6 over 1
pile 8 over 6
pile 8 over 5
move 2 over 1
move 4 over 9
quit
```

Output:

```
0: 0
1: 1 9 2 4
2:
3: 3
4:
5: 5 8 7 6
6:
7:
8:
9:
```