# 3650 Gap

There are many different ways of saying things and also of writing things. Sometimes there are much more different ways of saying things than things that are worth to be said. Sometimes people invent funny ways of telling things, sometimes people find complicated ways of not telling anything. This is true not only of people professionally engaged in the communication business, but also of everyone else. It is hard to communicate (do you see what we mean?).

To help you correlate what some people may want to say (or not) you decided to workout a tool that may be used to compare two texts with gaps, and tell whether the patches may be filled so that the resulting texts will become the same. Gaps in the text are named by strings and have a length attached. For example, the sequence of characters <$firstname$:10> identifies a gap named "$firstname$" of length 10. Here is an example of texts with gaps (just to clarify, we identify here blank spaces in the input with the symbol ␣), but this symbol doesn't exist in the input file, that matches exactly the Sample Input below's format).

```
<name:3>'s␣boat␣is␣no␣longer␣than␣Anne's.␣If␣Joe␣likes
␣<food:7>␣then␣so␣do␣I.␣Usually,␣<food:7>␣are␣yellow.

<thing:10>␣is␣no␣longer␣than␣Anne's.␣If␣<name:3>␣likes
␣bananas␣then␣so␣do␣I.␣Usually,␣bananas␣are␣<color:6>.
```

The texts may be matched by consistently replacing each named gap by some string of the appropriate length. The same named gap may appear in either text, always with the same associated length. In the example above, we may set:

```
color yellow
food bananas
name Joe
thing Joe's boat
```

Your goal is to write a program that given two texts with gaps will determine if the texts can be matched, in which case it must list how to fill the gaps, or not.

## Input

The input will contain several test cases, each of them as described below. Consecutive test cases are separated by a single blank line.

The input consists of two texts, in sequence. A text is given as a sequence of printable characters, for convenience split into several lines. Each text is specified by an integer $N$, in a single line, indicating how many lines the text has, followed by precisely $N$ lines. To obtain the text from its lines, you should just concatenate the contents of all the lines, in order. The text may contain the usual alphabetic and punctuation characters, and also sequences of the form '<$identifier$:$integer$>' indicating a gap in the text. The $identifier$ is a sequence of alphabetic characters, with the name of the gap, and $integer$ is an integer (between 0 and 32), with the length of the gap. The number of lines in each test does not exceed 100 lines, and each line does not exceed 400 characters.

## Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

If the texts can be matched, the output will contain '`yes`' in the first line, followed by the strings that have been chosen to fill the gaps. For each gap, you should list the gap identifier, followed by a single space, and the text selected to fill the given gap. This list should appear by alphabetic order of the gap identifiers. If your matching is not able to identify some character precisely, then such character must be printed as '`*`' in the output.

If the texts cannot be matched, the output should contain '`no`' in a single line.

## Sample Input

```
2
<name:3>'s boat is no longer than Anne's. If Joe likes
 <food:7> then so do I. Usually, <food:7> are yellow.
2
<thing:10> is no longer than Anne's. If <name:3> likes
 bananas then so do I. Usually, bananas are <color:6>.

1
potato<bingo:6>.
1
po<bobo:6> ppp.
```

## Sample Output

```
yes
color yellow
food bananas
name Joe
thing Joe's boat

yes
bingo ** ppp
bobo tato**
```