628 Passwords

Having several accounts on several servers one has to remember many passwords. You can imagine a situation when someone forgets one of them. He/she remembers only that it consisted of words `x`, `y` and `z` as well as two digits: one at the very beginning and the other one at the end of the password.

Your task is to write a program which will generate all possible password on the basis of given dictionary and set of rules. For the example given above the dictionary contains three words: `x`, `y`, `z`, and the rule is given as `0#0` what stands for `<digit><word_from_the_dictionary><digit>`.

Input

First line contains a number of words in the dictionary (n). The words themselves are given in n consecutive lines. The next line contains number of rules (m). Similarly consecutive m lines contain rules. Each rule consists of characters '`#`' and '`0`' given in arbitrary order. The character '`#`' stands for word from the dictionary whilst the character '`0`' stands for a digit.

Input data may contain many sets of dictionaries with rules attached two them.

Output

For each set 'dictionary + rules' you should output two hyphens followed by a linebreak and all matching passwords given in consecutive lines. Passwords should be sorted by rules what means that first all passwords matching the first rule and all words must be given, followed by passwords matching the second rule and all words, etc. Within set of passwords matching a word and a rule an ascending digit order must be preserved.

Assumptions: A number of words in the dictionary is greater than 0 and smaller or equal to 100 ($0 < n \leq 100$). Length of the word is greater than 0 and smaller than 256. A word may contain characters '`A`'..'`Z`','`a`'..'`z`','`0`'..'`9`'.A number of rules is smaller that 1000, and a rule is shorter that 256 characters. A character '`0`' may occur in the rule no more than 9 times, but it has to occur at least once. The character '`#`' is not mandatory meaning there can be so such characters in the rule.

Sample Input

```
2
root
2super
1
#0
1
admin
1
#0#
```

Sample Output

```
--
root0
root1
root2
root3
root4
root5
root6
root7
root8
root9
2super0
2super1
2super2
2super3
2super4
2super5
2super6
2super7
2super8
2super9
--
admin0admin
admin1admin
admin2admin
admin3admin
admin4admin
admin5admin
admin6admin
admin7admin
admin8admin
admin9admin
```

536 Tree Recovery

Little Valentine liked playing with binary trees very much. Her favorite game was constructing randomly looking binary trees with capital letters in the nodes.

This is an example of one of her creations:

```
      D
     / \
    /   \
   B     E
  / \     \
 /   \     \
A     C     G
           /
          /
         F
```

To record her trees for future generations, she wrote down two strings for each tree: a preorder traversal (root, left subtree, right subtree) and an inorder traversal (left subtree, root, right subtree).

For the tree drawn above the preorder traversal is `DBACEGF` and the inorder traversal is `ABCDEFG`.

She thought that such a pair of strings would give enough information to reconstruct the tree later (but she never tried it).

Now, years later, looking again at the strings, she realized that reconstructing the trees was indeed possible, but only because she never had used the same letter twice in the same tree.

However, doing the reconstruction by hand, soon turned out to be tedious.

So now she asks you to write a program that does the job for her!

Input Specification

The input file will contain one or more test cases. Each test case consists of one line containing two strings preord and inord, representing the preorder traversal and inorder traversal of a binary tree. Both strings consist of unique capital letters. (Thus they are not longer than 26 characters.)

Input is terminated by end of file.

Output Specification

For each test case, recover Valentine's binary tree and print one line containing the tree's postorder traversal (left subtree, right subtree, root).

Sample Input

```
DBACEGF ABCDEFG
BCAD CBAD
```

Sample Output

```
ACBFGED
CDAB
```