Suffixionary

Antonio's memory isn't so good, and on top of that his way of thinking tends to be backwards. Sometimes he would try to remember a word, but he would remember the *ending* rather than the beginning of that word, so he wouldn't even be able to look it up in a dictionary. The other day he was trying to remember a word. It was probably "procrastination", since that's essentially his way of life, or perhaps it was "explanation", since he definitely likes those, or maybe "indignation", given the way the world is going. Whatever it might have been, he knew that the word ended with "nation". In fact, it might have even been "nation" itself.

You should help Antonio by programming a dictionary application with which he could look up words by their suffix. Perhaps you should call this application a *suffixionary!*

Input

The first line of input contains two positive numbers, the number of words in the suffixionary, $n \leq 10^5$, and the number of queries, $q \leq 10^3$. Each of the following n lines contains a word consisting of a sequence of at most 30 lowercase letters ('a'–'z'). Then another q lines follow, each containing a query string consisting again of at most 30 lowercase letters.

Output

For each query string s in the given order, the output contains zero or more lines containing, in alphabetical order, all the words in the suffixionary that have s as a suffix.

Examples

Sample input 1

```
10 2
rumination
alienation
national
prespiration
convolution
nation
notional
capital
whatever
suffixation
nation
l
```

Sample output 1

```
alienation
nation
rumination
capital
national
notional
```

Limits

Time limit is 3 seconds.

Memory limit is 256 megabytes.