Mind your Vocabulary

The autocomplete dictionary application found on mobile phones takes the advantage of a retrieval data structure called Trie, which can be used to perform typical operations of finding a word present in a given dictionary and inserting a new word in the dictionary. These operations can be performed in time complexity linear in length of a word, which is not possible in data structures like Binary Search Tree. The purpose of this assignment is to make you familiar with this retrieval data structure. You are given a dictionary of words which you are required to maintain as a Trie data structure, by representing each Trie node as an object (make use of OOP!). You are given another set of strings (which may be a **valid word** already present in the dictionary) or may be just a **prefix of a valid word** present in the dictionary. You are required to sequentially process this list of given strings in the given order, and every time you encounter a **valid word**, print it along with its frequency obtained so far. In case of a **prefix**, print all the corresponding valid words in lexicographic order.

Input Format

The first line contains two space separated integers N and M, respectively denoting the number of words in the dictionary, and the number of strings/ prefixes to process.

Following N lines contain the words in the dictionary (all lower case ascii strings). Format of each line is:

$< \mathbf{word}_i >$

where $i = 1 \ldots N$. The subsequent M lines including the $(N+2)^{th}$ line contain exactly one string each (that needs to be processed), which may be a **valid word** (a word present in the dictionary) or may be a **prefix of a valid word**. Format of each line is:

$< \mathbf{string}_j >$

where $j = 1 \ldots M$.

Constraints

$5 \leq N \leq 10^5$

$1 \leq M \leq 10^7$

Output Format

There should be exactly M lines of output corresponding to the last M lines of input.

For each of $< \mathbf{string}_j >$ (where $j = 1 \ldots M$) corresponding to the last M lines of the input, process $< \mathbf{string}_j >$ to produce a line of output by the following rule:

1. If $< \mathbf{string}_j >$ is a **valid word** (present in given dictionary), output line will be in the format (with single space separation):

$< \mathbf{string}_j \ \mathbf{frequency}_j >$

where $\mathbf{frequency}_j$ is the number of times $< \mathbf{string}_j >$ has been encountered in the input **so far**.

2. If $< \mathbf{string}_j >$ is a **prefix of a valid word**, the output should first print the **prefix** followed by the corresponding **valid words** (in lexicographical order) possible for it by virtue of the dictionary given. The format is (with single space separation):

$< \mathbf{string}_j \ \mathbf{valid}_1^j \ \mathbf{valid}_2^j \ \ldots \ \mathbf{valid}_{k_j}^j >$

$\mathbf{valid}_1^j\ \mathbf{valid}_2^j\ \ldots\ \mathbf{valid}_{k_j}^j$ refer to the **valid words** (in lexicographical order) corresponding to $< \mathbf{string}_j >$. k_j denotes the number of **valid words** in the dictionary that contain $< \mathbf{string}_j >$ as a prefix.

Sample Input 0

```
10 2
damnedest
fouler
fulsome
incorporeal
judiciously
overcoats
preppies
sulfurs
taoists
yeps
overco
sulfurs
```

Sample Output 0

```
overco overcoats
sulfurs 1
```

Sample Input 1

```
10 5
damnedest
fouler
fulsome
incorporeal
judiciously
overcoats
preppies
sulfurs
taoists
yeps
taoists
ye
taoists
judicio
taoists
```

Sample Output 1

```
taoists 1
ye yeps
taoists 2
judicio judiciously
taoists 3
```

Sample Input 2

```
10 11
damnedest
fouler
fulsome
incorporeal
judiciously
overcoats
preppies
sulfurs
taoists
yeps
overcoats
overcoats
yeps
overcoats
```

```
damnedest
sulfurs
damnedest
preppies
damnedest
sulfurs
judicio
```

Sample Output 2

```
overcoats 1
overcoats 2
yeps 1
overcoats 3
damnedest 1
sulfurs 1
damnedest 2
preppies 1
damnedest 3
sulfurs 2
judicio judiciously
```

Sample Input 3

```
10 8
dunbar
gloriously
jawing
remain
remainder
remaindered
remainders
remained
remaining
remains
jawing
gloriously
dunb
jawing
jawing
gloriously
remai
gloriously
```

Sample Output 3

```
jawing 1
gloriously 1
dunb dunbar
jawing 2
jawing 3
gloriously 2
remai remain remainder remaindered remainders remained remaining remains
gloriously 3
```

Explanation 3

1. 10 and 8 in the first line of input gives the number of words in the dictionary and the number of subsequent strings to process respectively. Lines 2 to 11 contain the dictionary words. Lines 12 to 19 contain the strings to process.
2. **`jawing`** is encountered first in line 12 of input, so print **`jawing 1`** in first line of output. It is again encountered in lines 15 and 16 of input, so print **`jawing 2`** and **`jawing 3`** respectively in output. Here 2 and 3 correspond to the frequency obtained till that line of input.
3. **`dunb`** in line 14 of input is not a valid word of the dictionary but is a prefix of **`dunbar`**. So print both the prefix and the corresponding valid word in line 3 of output. Note that there is only a single valid word, but in case of the prefix **`remai`** in line 18 of input, there are 7 valid words namely **`remain remainder remaindered remainders remained remaining remains`**, which are printed in the output in lexicographic order.