We all use search everyday; to find a file in a directory, to find an email in the inbox, to find a song in a playlist. Search is more than just a linear scan through a list of texts in a dictionary; It's binary search, it's indexing, it's using your full text search algorithms to solve one of the hardest problems we know.

– adapted from NUMB3RS

Given a dictionary containing less than $N = 20000$ words labeled from 1 to N. Each word consists of lowercase characters (from 'a' to 'z') with arbitrary length. The total number of characters in the dictionary is at most 100,000. Your task is to answer at most $Q = 100000$ queries. Each query q_i is also a word (as defined above). For each query, you have to print the "Top 10" words in the dictionary with the following rules:

- All the words in the "Top 10" have to contain the substring q_i.
- All the words in the "Top 10" have to be sorted in this order:
  1. The words with shorter length come first, if they have equal length then
  2. The lexicographically smaller words come first, otherwise
  3. The words with smaller label come first.
- If the number of words in the dictionary that contains the substring q_i is less than 10 then print all the words otherwise, print only the top-10 words (note: the words are printed using their labels).
- If there is no word in the dictionary that contains the substring q_i then print '`-1`' (without the quotes).

Input

The first line contains the number N. The next N lines contains the N words in the dictionary (the i-th line is the word with label i). The next line contains the number Q followed by the Q lines containing the queries.

Output

For each query, print one line containing the labels of the "Top 10" words (separated by a space) in the dictionary using the rules defined above.

Sample Input

```
17
acm
icpc
regional
asia
jakarta
two
thousand
and
nine
arranged
by
universitas
bina
nusantara
especially
for
you
5
a
an
win
b
z
```

Sample Output

```
1 8 4 13 5 10 3 7 14 15
8 10 7 14
-1
11 13
-1
```