☆ Sort hotel list

Given a set of hotel identifiers and some number of reviews from guests about a hotel, sort the hotel identifiers based on a list of keywords of interest (these will be given)
The criteria to sort the hotels should be how many times the interesting keywords are mentioned in any reviews about a given hotel.

Input:

There are 3 groups of input: keywords, hotel_ids, and reviews.

The first input contains keywords, as a single line of a space-separated words.
(If you use the pre-generated input code stubs, this will be the first string argument to your function. If you are parsing input yourself, this string of keywords will be the first line of input.)

The second line contains one integer `M`, which means the number of lines in the following group.
Then followed by `M` lines of integers, which mean the list of hotel IDs.

Then followed by one integer `N`, which means the number of lines in the following group.
Then followed by `N` lines of strings, which means the list of reviews. Each review is a single string, on a single line.

The list of hotel IDs and the list of reviews are parallel arrays. These two lists are always the same length (`N == M`). For example, if they are put into 2 arrays called `hotel_ids` and `reviews`, respectively, then `reviews[3]` is a review for hotel `hotel_id[3]`. (A hotel can have more than one review).

Output:
A list of hotel IDs. It should be sorted, in descending order, by how many mentions they have of the keywords across all of the reviews for a given hotel.
(If you are not using the pre-generated output code stubs, you should print each hotel ID on a separate line.)

Notes:
* The words to be find will **always** be single words like 'breakfast' or 'noise'. **Never** double words like 'swimming pool'.
* Hotel ID is a 4-byte integer.
* Matching should be case-insensitive.
* Dots and commas should be ignored.
* If a word appears in a review twice, it should count twice

Notes:

* The words to be find will **always** be single words like 'breakfast' or 'noise'. **Never** double words like 'swimming pool'.
* Hotel ID is a 4-byte integer.
* Matching should be case-insensitive.
* Dots and commas should be ignored.
* If a word appears in a review twice, it should count twice.
* If two hotels have the same number of mentions, they should be sorted in the output based on their ID, smallest ID first.
* In case one or more test cases time out, consider revisiting the runtime complexity of your algorithms.

Sample Input

```
breakfast beach citycenter location metro view staff price
5
1
2
1
1
2
5
This hotel has a nice view of the citycenter. The location is perfect.
The breakfast is ok. Regarding location, it is quite far from citycenter but price is cheap so it is worth.
Location is excellent, 5 minutes from citycenter. There is also a metro station very close to the hotel.
They said I couldn't take my dog and there were other guests with dogs! That is not fair.
Very friendly staff and good cost-benefit ratio. Its location is a bit far from citycenter.
```

Sample Output

```
2
1
```

Explanation

Hotel **2** has 7 mentions of the keywords. Those are: "location" and "citycenter" are mentioned twice each, while "breakfast", "price", and "staff" are mentioned once each. Hotel **1** on the other hand has 6 mentions of the keywords. Those are: "location" and "citycenter" also twice each, and then "view" and "metro" once each.