C++ Programming: Exam Variant 1 (Exam-2017-05-21)

Solutions for each task will be submitted in the form of compressed archive (.zip) files, containing .h and .cpp files.

Please be mindful of the strict input and output requirements for each task, as well as any additional requirements on running time, used memory, etc., as the tasks are evaluated automatically and not following the requirements strictly may result in your program's output being evaluated as incorrect, even if the program's logic is mostly correct.

For some of the tasks in this exam you are provided with files, which the Judge system places in your submitted solution. These files are the so-called "Solution Skeleton" and, depending on the task, may require you to write specific code for your solution to work (e.g. a Solution Skeleton may contain a file with the **`main()`** function defined, in which case your task will usually be to implement a class or function in another file, for the program to work correctly). DO NOT attempt to edit the Solution Skeleton files – the Judge system overwrites any files from the skeleton you submit, so it won't see your changes to them. Some tasks may contain additional files you can use (and edit) if you want – if so, this will be described explicitly in the task.

You can use C++03 and C++11 features in your code.

Unless explicitly stated, any integer **input** fits into **`int`** and any floating-point **input** can be stored in **`double`**.

Task 3 – Emails (E1-Task-3-Emails)

Captain Grant is a Scottish naval officer, and, as you might know, has a lot of paperwork to do. He especially hates reading emails – he gets a lot of irrelevant stuff in his inbox, and he'd much rather spend some quality time with the bottles in his cabinet than by sorting through thousands of emails. He needs a program which, given a search word, prints messages containing that word, ordered by how many times the word appears in them.

Your task is to write a program, which, given a search word **S** (a sequence of lowercase English letters), a page size **P** (a positive integer number) and a set of email messages in the captain's inbox (single-line strings of words consisting lowercase English letters, separated by spaces), prints out at most **P** messages, ordered by the number of times the search word appears in them (don't print out messages which don't contain the search word). **A word is considered to match the search word** only if **all the characters** of both words are equal and the words are of **equal length**.

For example, if the captain is searching for messages containing the search word **`"speed"`**, specifies a page size of **2** (e.g. he only has time to read the 2 most-relevant emails), and has the messages:

```
1. speed give me what I need yeaaaah
2. speedy gonzales is speedier than a speeder from from star wars
3. a crucial part of any naval engagement is speed speed speed
4. the speed of hms daring is lower than the speed of your ship
```

the program should output messages **3** and **4** (in that order). If the page size **P** was **4**, then the program should output messages **3**, **4** and **1** only (the other message doesn't contain the word "speed" at all – message **2** has words which have "speed" as a sub-word, but that doesn't count as a match, as described above).

Input

The first line of the standard input contains the search word **S** (sequence of **a**-**z** characters).

The second line of the standard input contains a single positive integer number – the page size **P**.

Each of the following lines is a message – words (sequences of **a**-**z** characters) separated by spaces.

The last line of the input contains only the dot character (**`"."`**), signaling the end of the input

Output

At most **P** lines, containing the most-relevant messages (those with the most occurrences of the search word), in descending order (i.e. those with more occurrences of the search word come before those with less occurrences of the search word).

Restrictions

The input will not contain more than **`100`** messages containing the search word. No two messages will have an equal number of occurrences of the search word. There will always be at least **1** message containing the search word.

The total number of messages will be **`2500`** or less. **`50%`** of the tests will have up to **`100`** messages.

Each message will contain **`100`** words or less. Each word will be up to **`10`** letters long.

The total running time of your program should be no more than **`0.2s`** (Hint: before reading input, do **`cin.sync_with_stdio(false)`** to speed up reading if you're using **`cin`** in some way)

The total memory allowed for use by your program is **`2.5MB`** (NOTE: "hello world" uses about **`1.7MB`**)

Example I/O

Example Input	Expected Output
to 2 to get to the end you first need to begin ground control to major tom this is major tom to ground control to be or not to be to go to the top you first need to go to the bottom that is the question .	to go to the top you first need to go to the bottom to get to the end you first need to begin
ab 5 ab ab a a a abba a bb aa ab .	ab ab a bb aa ab