C++ Fundamentals: Judge Assignment 3 (JA3)

The following tasks should be submitted to the SoftUni Judge system, which will be open starting **Saturday, 23 December 2017, 10:00** (in the morning) and will close on **Friday, 12 January 2018, 23:59**. Submit your solutions here: https://judge.softuni.bg/Contests/Compete/Index/893.

After the system closes, you will be able to "Practice" on the tasks – however the "Practice" results are NOT considered in the homework evaluation.

Tasks 1 and 2 of this assignment will require submitting **compressed archive (.zip) files**, containing **a single .cpp file**. Both tasks will have a **.h** file included and you can use that file directly in your code by **#include**-ing it (the Judge system will have a copy of the file) – or you can just copy the file's contents into the **.cpp** file you submit.

For **tasks 3 and 4** of this assignment, the code for each task should be a single C++ file, the contents of which you **copy-paste** into the Judge system (like with previous judge assignments).

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.

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`**.

NOTE: the tasks here are NOT ordered by difficulty level.

Task 4 – Transmission (JA3-Task-4-Message)

You are part of a SETI (*that thing that's supposed to find aliens but never does*) team and you have just detected messages from a distant star system. The signal has been verified to indeed come from a star system, not from an Earth-orbiting satellite or any other sort of interference, and your team is certain the source is artificial. Your team wants to analyze the message, and for that they need to first identify the most-commonly encountered signals in the message.

The message has been converted to a sequence of English words (containing characters **a-z** and digits **0-9**), separated by spaces. The message ends with a dot, preceded by a space (**" ."**).

Your task is to write a program, which answers queries about the message in the form **occurrenceCount index** by finding all words which appear an **occurenceCount** number of times in the transmission, and printing out the word at position **index** in the lexicographical order of the (unique) words. If there are no words with that **occurenceCount**, print the dot character (**"."**).

For example, if we have the message string:

"chug a mug of mead and another mug mead chug another mug of mead warrior ."

(*What? Nobody said the aliens can't be from Skyrim…*)

and type in **3 0**, the program should output **mead** – the words **mead** and **mug** each appear **3** times in the message, and ordered lexicographically they form the array **{"mead", "mug"}**, in which the element at the **0** index is the word **mead**.

If we instead type in **2 1**, the program should output **chug** – the words that appear **2** times in the message are **chug** and **another**. If we sort them lexicographically we get **{"another", "chug"}**, and index **1** in that array is **chug**.

If we instead type in **4 0**, the program should output **.** (a single dot character) – there is no word that appears **4** times in the text.

Write a program which reads in a message in the above-mentioned format, and queries in the above-mentioned format, and prints out the results for those queries

Input

The first line of the input will contain the message – a string containing English characters **a-z**, digits **0-9**, spaces, and ending with a space and a dot (**" ."**). Words in that string are considered sequences of characters and/or digits separated by spaces.

The next line will contain a single integer **N** – the number of queries.

Each of the following lines will contain two positive integer numbers, separated by a single space – the **occurrenceCount** and **index** values of the query.

Output

N lines, each containing a single word, representing the answers to the queries in the same order the queries were given.

Restrictions

0 < N < 500;

The total number of words will be at most **1000**. Of those, no more than **50** will be unique. Each word will be at most **20** symbols.

The total running time of your program should be no more than **`0.05s`**

The total memory allowed for use by your program is **`5MB`**

Example I/O

Example Input	Expected Output

`chug a mug of mead and another mug mead chug another mug of mead warrior .`
`2`
`3 0`
`2 1`
	`mead`
	`chug`