Animal Finding

Objective
Give practice with reading and writing to standard input in C.
Give practice with loops and conditionals in C.
Give practice with strings in C.

Story
It's a *Cat*-tastrophy! You were just put in charge of running **The Wild Experience**, a safari theme park, and it appears to be an *otter* mess. The park is divided into several sections, each of which contain cages. Each cage can contain multiple animals. Sadly, *owl* the sections and cages have seemingly random animals contained within them.

You want to work on organizing them, but first you need to do some *cattle*-logging. You have an advanced imaging system that can scan sections of the park and produce a simplified image format. It wasn't *sheep*, but it will be worth it.

The output of your imaging system will be a sequence of letters, where capitalization will denote differences in the image. Currently, *ewe* want to know how many of each type of animal might exist within the park. You know for a particular animal what sequence of letters represents it (called an animal pattern). For now we will count the number of occurrences of these particular letter sequences. You can assume that nothing will block the animal pattern from the imaging system

Problem
Given a list of newline-separated park sections and an animal pattern (as a letter sequence), find out how many times the pattern occurs in each park section.

Input
Input will begin with a line containing 2 tokens, $\boldsymbol{S}$ and $\boldsymbol{N}$ ($1 \leq |\boldsymbol{S}| \leq 20$; $1 \leq \boldsymbol{N} \leq 100{,}000$), representing the target animal pattern and the number of sections to process respectively. All characters in $\boldsymbol{S}$ will be either upper or lower case latin letters. The following $\boldsymbol{N}$ lines will contain a sequence of characters representing a scanned section of the park. Each park section will contain at most 100,000 characters (letters and spaces; not including the new line at the end of the line).

Output
Output should contain $\boldsymbol{N}$ lines. Each line will contain a single integer representing the number of occurrences of the letter sequence for the given section. NOTE: You do not need to wait to print out the numbers. You can print the number after each line of input.

Sample Input	Sample Output
qq 4 qxq qqpqqppqpq q q abc qqq	0 2 0 2
hello 2 HELLO hello Hello WORLD world World olleh	1 0

Explanation

Case 1

The first line says that we are looking for the pattern "qq" and that there are 4 sections (lines) in our theme park

In the first section ("qxq") there is no contiguous occurrence of the animal pattern (i.e. letters "qq").

In the second section ("qqpqqppqpq") there are two locations of qq, which are highlighted for your convenience. There are 2 q's at the end, but they are not consecutive.

In the third section ("q q") the q's in the beginning are separated by spaces, so it could not be the animal we are looking for.

In the fourth and final section ("abc qqq") the end has qqq, which gives us 2 qq's. The first 2 letters are the first qq and the last 2 letters are the second qq. **Note** that this means that a letter can be part of multiple animals.

Case 2

There are only two sections. "HELLO hello Hello WORLD world World" is the first section. Although, there are several "hello"s of any capitalization, there is exactly one that matches in terms of case.

The second section "olleh" looks to contain the animal pattern backwards, but we won't count it, since it does not match the given pattern as given. Thus there are 0 instances of the animal pattern in the second section.

Helpful Hints/Recommendations

Reading First Line: I recommend reading in the first number using traditional scanf. If you want to you could instead read the first line using fgets or gets, but then you need to use a method to parse the integer from the string you extract.

Multiple Following Lines: After reading the first line, you need to process all the remaining lines. There are multiple lines to process, and each processing step will be identical. For this reason I recommend using a loop to process the remaining input. I recommend using an existing method to read a line (e.g. fgets or gets).

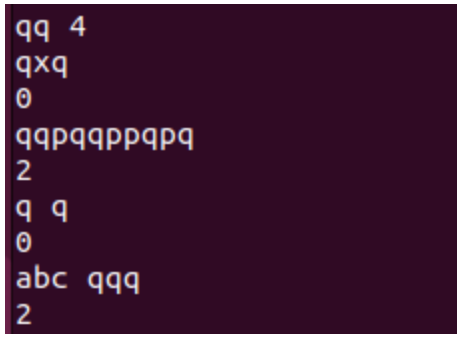
Mixing Token and Line Reading: IMPORTANT If you read the first line using scanf (or fscanf) the first call of fgets or gets will read the remainder of that first line rather than following line. For this reason you might need to use an extra fgets or gets to read (and ignore) the remnants of the first line.

Line Processing: The name processing should involve reading in each character of the name until a new line character is reached, but be careful, since there might be a newline character at the end of the line with the number of names.

fgets: Keep in mind when using fgets you should specify the size of the buffer

Null Terminator: Keep in mind that if you are using character arrays as strings in C you need to make enough space for the Null Terminator at the end of the string.

Storage: DO NOT store the full input. I recommend storing only the current line that is being processed. DO NOT store the output. You can print as you go. Below is an example of my program running on the sample. I have labeled the output



Output =>

Output =>

Output =>

Output =>

Section Handling: You should check every valid spot in the section as a location for an animal pattern. This can be done using a loop. To check if a spot matches the animal pattern, I recommend using another loop.

Grading Criteria

- Read/Write from/to **standard** input/output (e.g. scanf/printf and no FILE *)
    - 10 points
- Good comments, whitespace, and variable names
    - 15 points
- No extra input output (e.g. input prompts, "Please enter the number of friends:")
    - 10 points
- Read in all the input
    - 5 points
- Loop over all the sections
    - 10 points
- Programs will be tested on 10 cases
    - 5 points each

No points will be awarded to programs that do not compile using "`gcc -std=gnu11 -lm`*".*

Sometime a requested technique will be given, and solutions without the requested technique will have their maximum points total reduced. For this problem use a loop and conditional. ***Without this programs will earn at most 50 points!***

Any case that causes a program to return a non-zero return code will be treated as wrong. Additionally, any case that takes longer than the maximum allowed time (the max of {5 times my solutions time, 10 seconds}) will also be treated as wrong.

No partial credit will be awarded for an incorrect case.