Subsequences in Substrings

Kattis - subsequencesinsubstrings

You are given two strings s, and t. Count the number of substrings of s that contain t as a subsequence at least once.

Note that a *substring* and a *subsequence* both consist of characters from the original string, in order. In a *substring*, the characters must be contiguous in the original string, but in a *subsequence*, they are not required to be contiguous. In the string **abcde**, **ace** is a *subsequence* but not a *substring*.

If s is **aa** and t is **a**, then the answer is 3: **[a]a**, **[aa]**, and **a[a]**.

Input

Each test case will consist of exactly two lines.

The first line will contain string s ($1 \leq |s| \leq 10^5$, $s \in [a-z]^*$), with no other characters.

The second line will contain string t ($1 \leq |t| \leq 100$, $|t| \leq |s|$, $t \in [a-z]^*$), with no other characters.

Output

Output a single integer, which is the number of substrings of s that contain t as a subsequence at least once.

Sample 1

Input	copy	Output	copy
abcdefghijklmnopqrstuvwxyz a		26	

Sample 2

Input	copy	Output	copy
abcdefghijklmnopqrstuvwxyz m		182	

Sample 3

Input	copy	Output	copy
penpineappleapplepen ppap		68	

Palindromes [Kattis - palindromes]

Sam is starting a new company, and has just chosen its name. The name is a string $s = s_1 s_2 \ldots s_n$ of length n. The name is so long that customers won't remember the full name. Rather, each customer is going to remember only a substring t of the full name, and different customers may remember different substrings.

Sam is curious how memorable the substring t will be to a customer. Sam believes that the more palindromes t contains, the more memorable it is. Recall that a string w is a palindrome if w reads the same forwards and backwards (for example, the strings `radar` and `level` are palindromes, while the string `ever` is not). After all, customers are sensitive to symmetries in the substring t that he or she remembers, and palindromes are full of symmetries. As such, Sam wants to find out the number of substrings w of t such that w is a palindrome.

The challenge is that Sam's company will have tons of customers. It is difficult for Sam to know how memorable his company name is to all these customers. Can you help Sam?

Input

The first line of input contains a string $s = s_1 s_2 \ldots s_n$ consisting of n lowercase letters ($1 \leq n \leq 100\,000$). The second line contains an integer m ($1 \leq m \leq 100\,000$), denoting the number of customers. Then m lines follow, each containing a pair of integers ℓ and r ($1 \leq \ell \leq r \leq n$), indicating that a customer remembers precisely the substring $s_\ell s_{\ell+1} \ldots s_r$ between the ℓth and rth characters inclusive.

Output

For each customer, output a line containing an integer that indicates how memorable the company name is to the customer. More specifically, if this customer remembers the substring $s_\ell s_{\ell+1} \ldots s_r$, output the number of pairs of indices (i, j) such that $\ell \leq i \leq j \leq r$ and $s_i s_{i+1} \ldots s_j$ is a palindrome.

Sample 1

Input	copy	Output	copy
aabacab 5 1 7 1 4 3 7 2 5 5 7		11 6 7 5 3	

Neal is very curious about combinatorial problems, and now here comes a problem about words. Knowing that Ray has a photographic memory and this may not trouble him, Neal gives it to Jiejie.

Since Jiejie can't remember numbers clearly, he just uses sticks to help himself. Allowing for Jiejie's only 20071027 sticks, he can only record the remainders of the numbers divided by total amount of sticks.

The problem is as follows: a word needs to be divided into small pieces in such a way that each piece is from some given set of words. Given a word and the set of words, Jiejie should calculate the number of ways the given word can be divided, using the words in the set.

Input

The input file contains multiple test cases. For each test case: the first line contains the given word whose length is no more than 300 000.

The second line contains an integer S, $1 \leq S \leq 4000$.

Each of the following S lines contains one word from the set. Each word will be at most 100 characters long. There will be no two identical words and all letters in the words will be lowercase.

There is a blank line between consecutive test cases.

You should proceed to the end of file.

Output

For each test case, output the number, as described above, from the task description modulo 20071027.

Sample Input

```
abcd
4
a
b
cd
ab
```

Sample Output

```
Case 1: 2
```

Anthem of Berland CodeForces - 808G

Berland has a long and glorious history. To increase awareness about it among younger citizens, King of Berland decided to compose an anthem.

Though there are lots and lots of victories in history of Berland, there is the one that stand out the most. King wants to mention it in the anthem as many times as possible.

He has already composed major part of the anthem and now just needs to fill in some letters. King asked you to help him with this work.

The anthem is the string s of no more than 10^5 small Latin letters and question marks. The most glorious victory is the string t of no more than 10^5 small Latin letters. You should replace all the question marks with small Latin letters in such a way that the number of occurrences of string t in string s is maximal.

Note that the occurrences of string t in s can overlap. Check the third example for clarification.

Input

The first line contains string of small Latin letters and question marks s ($1 \le |s| \le 10^5$).

The second line contains string of small Latin letters t ($1 \le |t| \le 10^5$).

Product of lengths of strings $|s|\cdot|t|$ won't exceed 10^7.

Output

Output the maximum number of occurrences of string t you can achieve by replacing all the question marks in string s with small Latin letters.

Sample 1

Input	copy	Output	copy
winlose???winl???w?? win		5	

Sample 2

Input	copy	Output	copy
glo?yto?e??an? or		3	

Sample 3

Input	copy	Output	copy
??c????? abcab		2	

Note

In the first example the resulting string *s* is "winlose**win**winl**win**w**in**"

In the second example the resulting string *s* is "glo**r**yto**re**e**or**an**d**". The last letter of the string can be arbitrary.

In the third example occurrences of string *t* are overlapping. String *s* with maximal number of occurrences of *t* is "**ab**c**abcab**".

DNA repair HDU - 2457

Biologists finally invent techniques of repairing DNA that contains segments causing kinds of inherited diseases. For the sake of simplicity, a DNA is represented as a string containing characters 'A', 'G' , 'C' and 'T'. The repairing techniques are simply to change some characters to eliminate all segments causing diseases. For example, we can repair a DNA "AAGCAG" to "AGGCAC" to eliminate the initial causing disease segments "AAG", "AGC" and "CAG" by changing two characters. Note that the repaired DNA can still contain only characters 'A', 'G', 'C' and 'T'.

You are to help the biologists to repair a DNA by changing least number of characters.

Input

The input consists of multiple test cases. Each test case starts with a line containing one integers N (1 ≤ N ≤ 50), which is the number of DNA segments causing inherited diseases.
The following N lines gives N non-empty strings of length not greater than 20 containing only characters in "AGCT", which are the DNA segments causing inherited disease.
The last line of the test case is a non-empty string of length not greater than 1000 containing only characters in "AGCT", which is the DNA to be repaired.

The last test case is followed by a line containing one zeros.

Output

For each test case, print a line containing the test case number(beginning with 1) followed by the
number of characters which need to be changed. If it's impossible to repair the given DNA, print -1.

Sample

Input	copy	Output	copy
2 AAA AAG AAAG 2 A TG TGAATG 4 A G C T AGT 0		Case 1: 1 Case 2: 4 Case 3: -1	