Problem A: Nenokku

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

The very famous author of a not less famous book decided to continue his work and write a new one. He was writing his book on a computer with internet connection. Knowing this the boy Nenokku could get access to the not yet published book. Every evening the boy has been downloading new entries of the book to his computer. One time after downloading another chapter he was interested if the author used the word "book" in his book. But he is too lazy to read the book and wants you to write the program for checking if a specified word occurs in the text.

Input

Each line of input has a query of one of the following two types:

- `? <word>` ($< word >$ is a sequence of not more than 50 characters)
- `A <text>` ($< text >$ is a sequence of not more than 1024 characters)

A query of the first type asks you to look for an occurence of the word $< word >$ in the book. A query of the second type asks you to add the text $< text >$ to the book. There are no more than 30 lines of input.

Output

For every query of type 1 output `"YES"`, if the requested word appears in the book and `"NO"` otherwise. Your search should not be case sensitive.

Sample input and output

Input	Output
? love ? is A Loveis ? love ? WHO A Whoareyou ? is	NO NO YES NO YES

Problem B: Given a string

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

Given two strings α and β, you are to find all positions at which β occurs as a substring in α.

Input

The first line of input contains α and the second β. Both strings consists of lower case characters $a - z$, at most 10^5 in each one.

Output

On the first line output the number of occurrences of β in α. On the second line output the position in string α where β starts for each of those occurrences. The numbers in the second line should be printed in ascending order.

Sample input and output

Input	**Output**
abacaba aba	2 1 5

Problem C: Basis

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

A string S was written many times sequentially. After that a substring of that long string was given to you. You are to find out the minimal possible length of the basis string S.

Input

The input contains a string formed only by letters. The length of the string doesn't exceed $5 \cdot 10^4$ characters.

Output

Output the minimal length of the basis string S.

Sample input and output

Input	**Output**
zzz	1
bcabcab	3

Problem D: Substrings

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

A substring of a string $S = s_1 s_2 ... s_n$ is a sequence $s_i s_{i+1} s_{i+2} ... s_{j-1} s_j$ of consecutive characters of S. Given a string, find out how many unique substrings it contains, except the empty one.

Input

The input contains a string S ($1 \leq S \leq 10^4$), formed only by lowercase letters.

Output

Output the amount of unique substrings of S, except the empty one.

Sample input and output

Input	**Output**
aab	5
dabyx	15

Problem E: Cyclic suffixes

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

Consider a string $S = s_1 s_2 s_3 ... s_{n-1} s_n$ over alphabet Σ. We will call the string $s_1 s_2 s_3 ... s_{n-1} s_n s_1 s_2 ...$ *cyclic expansion* of length m of string S, i.e. we are adding string S to the end while the total length is not more than m.

We will call $\tilde{S}$, the infinite cyclic expansion of the string S, *cyclic string*.

Let's consider the suffixes of the string $\tilde{S}$. Obviously it has not more than $|S|$ different suffixes: $(n+1)$-th suffixes is the same as the 1-st, $(n+2)$-th with the 2-nd and so on. Moreover the number of different suffixes could be even less. For instance, if $S =$ `abab`, first four suffixes of the cyclic string $\tilde{S}$ would be:

$$\tilde{S}_1 = \texttt{abababab...}$$
$$\tilde{S}_2 = \texttt{bababababa...}$$
$$\tilde{S}_3 = \texttt{abababab...}$$
$$\tilde{S}_4 = \texttt{bababababa...}$$

Here we can see only two different suffixes, although $|S| = 4$.

Sort the first $|S|$ suffixes of $\tilde{S}$ lexicographically. If two suffixes are same, their relative positions should remain unchanged. We then want to know which position $\tilde{S}$ is in after sorting.

For example with the string $S =$ `cabcab` we have:

$$\tilde{S}_2 = \texttt{abcabcabca...} \quad (1)$$
$$\tilde{S}_5 = \texttt{abcabcabca...} \quad (2)$$
$$\tilde{S}_3 = \texttt{bcabcabcab...} \quad (3)$$
$$\tilde{S}_6 = \texttt{bcabcabcab...} \quad (4)$$
$$\tilde{S}_1 = \texttt{cabcabcabc...} \quad (5)$$
$$\tilde{S}_4 = \texttt{cabcabcabc...} \quad (6)$$

Here the string $\tilde{S} = \tilde{S}_1$ ends up in position 5.

Given a string S, you are to find it's position in the list specified above.

Input

The input contains the string S ($1 \leq |S| \leq 10^6$), formed only by lowercase letters.

Output

Output the position of $\tilde{S}$ in the sorted list of the first $|S|$ suffixes of $\tilde{S}$.

Sample input and output

Input	**Output**
abracadabra	3
cabcab	5