Problem K. Missing LDAP

Time Limit 1000 ms
Mem Limit 262144 kB
OS Windows

This is an interactive problem. You have to use a `flush` operation right after printing each line. For example, in C++ you should use the function `fflush(stdout)`, in Java — `System.out.flush()`, in Pascal — `flush(output)` and in Python — `sys.stdout.flush()`.

In this problem, you must assist UNAL's tech team in determining the next unused LDAP given a student's full name. Student names consist of three or four words, with the final two representing the surnames and the preceding word(s) representing the first name(s).

LDAPs are generated sequentially. The initial LDAP is formed by concatenating the first letter of the first given name, the first letter of the second given name (if applicable), the entire first surname, and the first letter of the second surname.

If an LDAP is already taken, the process is repeated by using two letters from the first given name, followed by the first letter of the second given name (if applicable), and the entire first surname. Additional letters from the second surname are then appended until the total number of letters matches the number of letters used from the first given name. This pattern continues, incrementing the number of letters from the first given name by one until all letters from the first given name have been used. Once the first given name is fully used, letters from the second surname can be added as needed until all the second surname is complete.

For example if 'James David Rodriguez Rubiano', had decided to study at UNAL instead of being a football player, his LDAPs progression would have been

- JDRodriguezR
- JaDRodriguez
- JaDRodriguezR
- JaDRodriguezRu
- JamDRodriguez
- JamDRodriguezR
- JamDRodriguezRu
- JamDRodriguezRub
- JameDRodriguez
- ...

- JamesDRodriguezRubi
- JamesDRodriguezRubia
- JamesDRodriguezRubian
- JamesDRodriguezRubiano

Another example, for 'alex de hoz' would be:

- adeh
- alde
- aldeh
- aldeho
- alede
- aledeh
- aledeho
- aledehoz
- alexde
- alexdeh
- alexdeho
- alexdehoz

Each of the four names consists of uppercase and lowercase English letters and has a maximum length of 20 characters. It is guaranteed that the combination of the first letter of the second given name and the entire first surname will not appear as a substring within any other name. And also that there is at least one unused LDAP.

You can make queries to the testing system. Each query is a valid LDAP. Flush output stream after printing each query. There are two different responses the testing program can provide:

- the string "`1`" (without quotes), if the LDAP in your query has been used;
- the string "`0`" (without quotes), if the LDAP in your query has not been used.

When your program guesses the LDAP, print string "`! x`", where x is the answer, and **terminate your program normally** immediately after flushing the output stream.

Your program is allowed to make no more than 20 queries (not including printing the answer) to the testing system.

Input

Use standard input to read the responses to the queries.

The first line contains the full name of the student, 3 or 4 words separated by space.

Following lines will contain responses to your queries — strings "0' or "1". The i-th line is a response to your i-th query. When your program will guess the LDAP print "`! x`", where x is the answer and terminates your program.

The testing system will allow you to read the response on the query only after your program prints the query for the system and performs `flush` operation.

Output

To make the queries your program must use standard output.

Your program must print the queries — guessed LDAP x_i, one query per line (do not forget "*end of line*" after each x_i). After printing each line your program must perform operation `flush`.

Each of the values x_i mean the query to the testing system. The response to the query will be given in the input file after you flush output. In case your program guessed the LDAP x, print string "`! x`", where x — is the answer, and terminate your program.

Examples

Input	Output

`james rodriguez rubio`
`0`
 	
`0`
 	
`1`
	`jamrodriguez`
	
	`jarodriguezr`
	
	`jrodriguezr`
	
	`! jarodriguez`

Note

LDAPs preserve the capitalization of the names. (If a letter is uppercase in the name should be uppercase in the LDAP, same for lowercase)

This is (possibly) not the actual LDAP creation algorithm. It was designed just for the sake of the problem.

The first part should be a prefix of the first given name, and the last part should be a prefix of the second last name. The last name prefix can only be longer than the first name prefix once the entire first name has been used.