B. Sequence Formatting

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp is very careful. He even types numeric sequences carefully, unlike his classmates. If he sees a sequence without a space after the comma, with two spaces in a row, or when something else does not look neat, he rushes to correct it. For example, number sequence written like "`1,2 ,3,...,     10`" will be corrected to "`1, 2, 3, ..., 10`".

In this task you are given a string s, which is composed by a concatination of terms, each of which may be:

- a positive integer of an arbitrary length (leading zeroes are not allowed),
- a "comma" symbol ("`,`"),
- a "space" symbol ("` `"),
- "three dots" ("`...`", that is, exactly three points written one after another, also known as suspension points).

Polycarp wants to add and remove spaces in the string s to ensure the following:

- each comma is followed by exactly one space (if the comma is the last character in the string, this rule does not apply to it),
- each "three dots" term is preceded by exactly one space (if the dots are at the beginning of the string, this rule does not apply to the term),
- if two consecutive numbers were separated by spaces only (one or more), then exactly one of them should be left,
- there should not be other spaces.

Automate Polycarp's work and write a program that will process the given string s.

Input

The input data contains a single string s. Its length is from 1 to 255 characters. The string s does not begin and end with a space. Its content matches the description given above.

Output

Print the string s after it is processed. Your program's output should be *exactly* the same as the expected answer. It is permissible to end output line with a line-break character, and without it.

Examples

input	Copy
`1,2 ,3,...,     10`	
output	Copy
`1, 2, 3, ..., 10`	

input	Copy
`1,,,4...5......6`	
output	Copy

```
1, , , 4 ...5 ... ...6
```

input Copy

```
...,1,2,3,...
```

output Copy

```
..., 1, 2, 3, ...
```