Classification Problem

Source file name: `problem.c`, `problem.cpp`, `problem.cs` or `problem.java`

The Smartmatic Programming Team (SPT) is developing a new programming language based on the functional programming language, Haskell. Right now they are focused on the Type Inference System (TIS), so they have contacted you to help them with this module.

Actually, the TIS is working but the types of every identifier is coded on a strange way, and they want to be able to print types on a human readable way.

Due to the simple nature of the language, they only allow 26 different identifiers (from 'a' to 'z'), and 26 different basic types (from 'A' to 'Z'). They have also two special types: tuples and functions (yes, a function is a type!). For functions they use the symbol '@' and for tuples the symbol '&'.

For example, the following expression denotes a function f with a parameter of type A returning a value of type B:
f:@AB

The following expression denotes an identifier x that is a tuple of two values of type C:
x:&CC

Tuples can have two or more elements, for example a tuple *t = (B, A, C)* can be written as x:&&BAC or as x:&B&AC, this is *t = (B, A, C) = ((B, A), C) = (B, (A, C))*.

Now we can form more complex expressions like a function g with a tuple *(A, C, B)* as parameter, returning a tuple *(X, Y)*:
g:@&&ACB&XY

In order to get a human readable type, tuples must be printed out as a series of types separated by comas between parenthesis, for example, the tuple t:&B&AC, or the tuple t:&&BAC must all be printed as:
t: (B, A, C)

Functions must be printed using the -> operator. For example, function f:@XY should be printed:
f: X -> Y

Input

Each input consists of a line containing the name of a variable V , $V \in \{a, b, . . . , z\}$ followed by a colon ':', and a type expression as defined in the problem statement. Each line will contain at most 1024 characters.

The end of the input is a line containing only the character '*'.

The input must be read from standard input.

Output

For each input case you must print a line with the corresponding type expression on a human readable form as described previously.

The output must be written to standard output.

Sample input	Output for the sample input
f:@AB	f: A -> B
x:&CC	x: (C, C)
x:&&BAC	x: (B, A, C)
x:&B&AC	x: (B, A, C)
g:@&&ACB&XY	g: (A, C, B) -> (X, Y)
z:@I@ID	z: I -> I -> D
*	