You are developing a new fashionable language that is not quite unlike C, C++, and Java. Since your language should become an object of art and fashion, you call it $i^{\#}$ (spelled i-sharp). This name combines all the modern naming trends and also hints at how smart you are.

Your language caters for a wide auditory of programmers and its type system includes arrays (denoted with '`[]`'), references (denoted with '`&`'), and pointers (denoted with '`*`'). Those type constructors can be freely combined in any order, so that a pointer to an array of references of references of integers (denoted with '`int&&[]*`') is a valid type.

Multiple variables in $i^{\#}$ can be declared on a single line with a very convenient syntax where common type of variables is given first, followed by a list of variables, each optionally followed by additional variable-specific type constructors. For example, the following line:

```
int& a*[]&, b, c*;
```

declares variables a, b, and c with types '`int&&[]*`', '`int&`', and '`int&*`' correspondingly. Note, that type constructors on the right-hand sides of variables in $i^{\#}$ bind to variable and their order is reversed when they are moved to the left-hand side next to type. Thus '`int*& a`' is equivalent to '`int a&*`'.

However, you discover that coding style with multiple variable declarations per line is confusing and is outlawed in many corporate coding standards. You decide to get rid of it and refactor all existing $i^{\#}$ code to a single variable declaration per line and always specify type constructor next to the type it refers to (instead of the right-hand side of variable). Your task it to write such refactoring tool.

Input

The input file contains several test cases, each of them as described below.

The input contains a single line with a declaration of multiple variables in $i^{\#}$. The line starts with a type name, followed by zero, one, or more type constructors, followed by a space, followed by one or more variable descriptors separated by '`,`' (comma) and space, and terminated by '`;`' (semicolon). Each variable descriptor contains variable name, followed by zero, one, or more type constructors.

Type name and variable names are distinct and consist of lowercase and uppercase English letters from '`a`' to '`z`' or '`A`' to '`Z`'. The line contains at most 120 characters and does not contain any extra spaces.

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

Write to the output a line for each variable declared in the input. For each variable write its declaration on a single line in the same format as in the input, but with all type constructors next to its type. Separate type with all type constructors from a variable name by a single space. Do not write any extra spaces.

Sample Input

```
int& a*[]&, b, c*;
Double[][] Array[];
```

Sample Output

```
int&&[]* a;
int& b;
int&* c;

Double[][][] Array;
```