Pointer Analysis

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Pointer analysis, which aims to figure out which objects accessible via a specific pointer variable in a program during the execution, is one of the fundamental parts of static program analysis. Now we want you to perform the context-insensitive pointer analysis on the test data.

A program contains 26 objects denoted by lowercase letters and each object has 26 member variables (a.k.a. fields, which are pointers that may point to some objects) denoted by lowercase letters as well. Meanwhile, there are 26 global pointers in the programs designated by uppercase letters.

There are four kinds of statements in a program. We use `[Variable]` to represent the name of a pointer, `[Field]` to represent the name of a member variable, and `[Object]` to represent an object.

	Format	Example	Description
Allocation	`[Variable]` = `[Object]`	$A = x$	pointer A can point to object x (i.e., x is accessible via A)
Assignment	`[Variable]` = `[Variable]`	$A = B$	pointer A can point to every object accessible via B
Store	`[Variable].[Field]` = `[Variable]`	$A.f = B$	for every object o accessible via A, the member variable f of o can point to every object accesible via B
Load	`[Variable]` = `[Variable].[Field]`	$A = B.f$	for every object o accessible via B, A can point to every object accessible via the member variable f of o

The context-insensitive pointer analysis assumes that statements of the program will be executed in any order for a sufficient number of times. For example, in the following two programs, both A and B can point to the object x and object o. The reason for that is, in the real world, the exact execution order and execution times of statements are difficult to predict.

First Program	Second Program
A = o	B = A
A = x	A = x
B = A	A = o

Now you are asked to perform a context-insensitive pointer analysis on a given program consists of N statements, and for each pointer, output the objects it can point to.

Input

The first line of the input contains one integer N ($1 \leq N \leq 200$), representing the number of statements in the program. There is exactly one space before and after the equal sign '`=`'.

Each of the following N lines contains one statement.

Output

The output should contains 26 lines.

In the i-th line, output the name of the i-th pointer (which is the i-th uppercase letter) followed by a colon '`:`' and a space, and then list the objects accessible via this pointer in alphabetical order.

Examples

standard input	standard output
5 B.f = A C = B.f C = x A = o B = o	A: o B: o C: ox D: E: F: G: H: I: J: K: L: M: N: O: P: Q: R: S: T: U: V: W: X: Y: Z:
4 A = o B.f = A C = B.f C = g	A: o B: C: g D: E: F: G: H: I: J: K: L: M: N: O: P: Q: R: S: T: U: V: W: X: Y: Z:
3 A = o B = A	A: ox B: ox C: