Kick Start 2016 - Round C

Evaluation

Problem

Given an unordered list of assignment statements, write a program to determine whether the assignment statements can be put in some order in which all variables can be evaluated.

For our problem, an assignment statement will consist of an assignment variable, an assignment operator, and an expression, in that order. Statements will be evaluated one at a time, in the order you choose for them. A variable can be evaluated if and only if it has been the assignment variable of a previous assignment statement.

To simplify the problem, all the expressions are single function calls. Functions can take an arbitrary number of arguments, including zero; a function with zero arguments is always valid, and a function with variable arguments is valid as long as all of the variables are evaluatable.

For example, for the following list of assignment statements:

```
a=f(b,c)
b=g()
c=h()
```

this is one order that makes every statement valid:

```
b=g()
c=h()
a=f(b,c)
```

This is because: (1) `b` and `c` can be evaluated because the expressions `g()` and `h()` don't depend on any variables; and (2)`a` can also be evaluated because expression `a` depends on `b` and `c`, which are evaluatable.

However, the order

```
b=g()
a=f(b,c)
c=h()
```

would not be valid, because `f(b, c)` has variable c as an argument, but variable c has not been an assignment variable yet.

Another example is: `a=f(a)`. This list of statements can't be evaluated because the expression `f(a)` depends on the variable `a` itself, which makes it impossible to evaluate the statement.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. The first line of each test case contains an integer **N**: the number of assignment statements. Then, **N** lines follow. Each contains one assignment statement.

Each assignment statement consists of three parts: the assignment variable, the assignment operator, and the expression, with no spaces in between. The assignment operator is always `=`. All expressions consist of a function name, then `(`, then zero or more comma-separated

variable names, then `)`. All variables and function names consist of one or more lowercase English alphabet letters. No variable has the same name as a function. No variable will appear more than once as the assignment variable. However, variables may appear more than once in various functions (even within the same function), and functions may appear more than once.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is `GOOD` if all variables are evaluatable or `BAD` otherwise.

Limits

1 ≤ **T** ≤ 20.
Time limit: 30 seconds per test set.
Memory limit: 1GB.
All functions take between 0 and 10 arguments, inclusive. All variable names consist of between 1 and 20 lowercase English alphabet letters.

Small dataset (Test set 1 - Visible)

1 ≤ **N** ≤ 100.

Large dataset (Test set 2 - Hidden)

1 ≤ **N** ≤ 1000.

Sample

Sample Input

```
4
3
a=f(b,c)
b=g()
c=h()
2
a=f(b)
b=f(a)
2
aaa=foo(x,y)
bbb=bar(aaa,bbb)
2
x=f()
y=g(x,x)
```

Sample Output

```
Case #1: GOOD
Case #2: BAD
Case #3: BAD
Case #4: GOOD
```