This is war time again! Our beloved Sultan has assumed the post of a general and going to the war with his huge army. Being an intelligent fellow as he is, he has discovered a handful of new weapons to help him win the war. In order to facilitate instructing his units, he assigns each type of weapon a single binary code. For example, he has three types of weapons —

FordCannon (code: 1)
EdmondMortar (code: 10)
HopcroftRifle (code: 00)

So, now when he makes one long beep and one short beep (that is, `10`) from his grand whistle, an EdmondMortar unit is fired while when he makes two short beeps means, `00`), a HopcroftRifle comes into action. Please note that, he has an unlimited supply of every types.

Problem arised when in an war, the units were coded like — FordCannon (01), EdmondMortar (001), HopcroftRifle (01001). Once, he went for a HopcroftRifle (01001) but his men fired a FordCannon (01) and then an EdmondMortar (001) instead. The furious Sultan realized that the coding itself was ambigous i.e. more than one sequence of the weapons can translate into the same signal.

Now, your task is to determine if a given coding scheme is ambiguous or not before the war starts.

Input

There will be around 100 test cases. Every test case starts with a single integer N ($1 \leq N \leq 100$), the number of weapon types. Each of the following N lines contains a string (denoting the name of the weapon, less than 21 characters) and a binary number (the code, less than 21 characters).

The last test case will be followed by a single '`0`'.

Output

For every test case, print '`Case #`x`:`', where x is the case number. Then print '`Ambiguous.`' if a code leads to multiple sequences or print '`Not ambiguous.`' otherwise.

Sample Input

```
3
FordCannon 1
EdmondMortar 10
HopcroftRifle 00
3
FordCannon 01
EdmondMortar 001
HopcroftRifle 01001
0
```

Sample Output

```
Case #1: Not ambiguous.
Case #2: Ambiguous.
```