Jimmy is a hard-working pupil in his second year at primary school. Recently he decided to convert all his notes into an electronic version. Sadly, he found that his math notes were full of ink blots.

He scanned the notes and sent them through his own OCR package (yes, he coded it all by himself at the age of 8). The OCR package replaced all ink blots by the string "machula".

**Problem specification**

You are given Jimmy's notes, processed by the OCR. They contain simple math exercises, which were used to practice addition on positive integers. Your task is to recover the damaged part of the notes.

**Input specification**

The first line of the input file contains an integer **T** specifying the number of test cases. Each test case is preceded by a blank line.

Each test case consists of exactly one line. The line represents an equation of the form "number + number = number", where each number is a positive integer. One part of the equation will be replaced by the string "machula". The string always covers a contiguous non-empty sequence of digits, possibly even an entire number. You may assume that for each equation in the input there will be exactly one way to fill in the missing digits.

**Output specification**

For each test case, the output shall contain one line of the form "number + number = number". The line must represent the equation from that test case with all missing digits filled in.

**Example**

**Input:**

3

23 + 47 = machula

3247 + 5machula2 = 3749

machula13 + 75425 = 77038

**Output:**

23 + 47 = 70

3247 + 502 = 3749

1613 + 75425 = 77038

Note: **int** in C++/C/Java or **long int** in Pascal is en

|  |
| --- |
| Problem code: ALCHE |

Alchemy is a discipline that is believed to span at least 2500 years of human history.  
It is most known for its intention of transforming matter, typically trying to come up  
with a recipe to make gold based on much less valued metals, aided by some non-metal  
components.  
Most scientists and scholars think that alchemy has failed. They surely do not know that  
a particular alchemist named Albert Ainstain, managed to create gold from a simple  
combination of ordinary iron (much more common and cheap than gold) and some good-  
old-fashioned water. However, the combination must have the exact proportion of grams  
of iron and centiliters of water to work, otherwise the alchemist would end up with useless  
rusty iron.  
Many alchemists and bussinessman had tried to recreate Albert Ainstain’s findings to  
achieve recognition, fame, prestige or economical welfare, but none of them have suceeded.  
Since you know very little about chemistry, alchemy and ancient practices, you believe  
that your chances for success in this task are bounded to using a computer in your benefit.  
Therefore, you decide to create a program that automatically tests a given combination  
of iron and water, and informs whether that combination has the correct proportion  
to produce gold. Of course, once you find the right proportion, you can double both  
amounts and get double the gold, cut both in half and get half the gold, or multiply both  
by any other real number to obtain the amount of gold you want. Your task is then,  
given the number of grams of iron and the number of centiliters of water, say whether  
the proportion between both components is the right one.

### Input

The input contains several test cases, each one described in a single line. The line  
contains two integers I and W separated by a single space, representing grams of iron  
and centiliters of water, respectively (1 ≤ I, W ≤ 106 ). The last line of the input contains  
the number −1 twice separated by a single space and should not be processed as a test  
case.

### Output

For each test case output a single line containing an uppercase “Y” if the combination  
produces gold, or an uppercase “N” otherwise.

**Where is the proper degree**

### Example

**Input:**  
1000 37  
999 37  
10000 370  
10001 370  
-1 -1  
  
**Output:**  
Y  
N  
Y  
N

Alice has recently learned to use the Sieve of Eratosthenes, an ancient algorithm for finding all

prime numbers up to any given limit. As expected, she was really impressed by it's simplicity and

elegancy.

Now, she has decided to design her own sieve method: The Sieve of Alice, formally defined by the

following procedure, which determines the Sieve of Alice up to a given limit N.

1. Create a list of consecutive integers from N to 2 (N, N-1, N-2, ..., 3, 2). All of those N-1

numbers are initially unmarked.

2. Initially, let P equal N, and leave this number unmarked.

3. Mark all the proper divisors of P (i.e. P remains unmarked).

4. Find the largest **unmarked** number from 2 to P – 1, and now let P equal this number.

5. If there were no more **unmarked** numbers in the list, stop. Otherwise, repeat from step 3.

Unfortunately, Alice has not found an useful application for it's Sieve. But she still wants to know,

for a given limit N, how many integers will remain unmarked.

### Input

The first line contains an integer T, the number of test cases (1 ≤ T ≤ 10^4) . Each of the next T lines

contains an integer N ( 2 ≤ N ≤ 10^6).

### Output

Output T lines, one for each test case, containing the required answer.

### Example

**Input:**

3

2

3

5

**Output:**

1

2

3

You: What's the score? Did I miss much?   
  
Me: It's 2-1 for elAhli and the second half just started. The first half was quite boring.   
  
You: Who scored first? elAhli or ezZamalek?   
  
Me: What difference does it make?   
  
You: Big difference! I can predict the outcome of the match if I knew the order of which goals were scored in the first half.   
  
Me: What do you mean?   
  
You: It's 2-1 for elAhli, right? One of three things could have happened: elAhli scored two goals then ezZamalek scored; Or, elAhli scored its first goal, then ezZamalek, then elAhli again; Or, ezZamalek scored first, then elAhli scored its two goals.   
  
Me: So?!! I still don't understand what difference does that make? It's still 2-1 for elAhli! Why don't you just relax and let us continue watching the game in peace.   
  
You: You don't understand!! I believe the probability of who'll win depends on the order of how goals were scored. Now I have to predict the outcome for 3 possibilities.   
  
Me: And what if the score was 3-2? What would you have done then?   
  
You: I would have to work for 5 different possibilities. No?   
  
Me: Of course not! The number of possibilities isn't always equal to the sum.   
  
You: Can you tell me when will it be equal to the sum?   
  
Me: You're a programmer, why don't you write a program that counts the number of possibilities and compare it to the sum?   
  
You: I don't have the time, I want to watch the match. Besides, I have nine other problems to worry about.   
  
Me: I'll give you a hint. The possibilities will be equal to the sum only if one of the teams scored a certain number of goals.

### Input

Your program will be tested on one or more test cases. Each test case specifies two natural numbers (A and B ) (separated by one or more spaces) representing the score of the first half. No team will be able to score more than 10 goals. The last line of the input file contains two -1's (which is not part of the test cases.)

### Output

Format For each test case where the number of possibilities is equal to the sum, print:   
  
A+B=C   
  
Where A and B are as above and C is their sum. If the number of possibilities is not equal to the sum, replace the '=' sign with '!=' (without the quotes.)

### Example

**Input:**

2 1

1 0

-1 -1

**Output:**

2+1=3

1+0=1