Natural numbers

Implement arithmetic operations for nonnegative integers whose values are allowed to be beyond the range supported by the computer's built-in integer arithmetics. Given two nonnegative integers A and B, the code should be able to decide whether A < B, A = B, or A > B, and to compute

- A + B,
- A - B, with the convention that A - B = 0 for A < B,
- A * B,
- A / B (integer division)
- A % B (remainder).

Moeover, we introduce the new operation called *truncated multiplication* A # B [M], as follows. This operation will depend on the particular base in which the numbers are represented, and within the tests, it is assumed that the base is 100. In other words, we assume that any number A is represented within the code as

A = A_0 + A_1 * BASE + A_2 * BASE^2 + ... ,

where 0 <= A_k < BASE are the digits, and we set BASE = 100 for the purposes of the tests. One can write the product A * B as

A * B = A_0*B_0 + (A_0*B_1+A_1*B_0) * BASE + (A_0*B_2+A_1*B_1+A_2*B_0) * BASE^2 + ...

If we remove the first M - 1 terms from this expansion, and divide the result by BASE^M, we get the truncated product A # B [M]. Note that truncated multiplication depends on a parameter M, which may be assumed to be a moderate sized integer (in particular well within the 32 bit range). For example, we have

910 * 820 = (10+9*100)*(20+8*100) = 10*20 + (10*8+9*20)*100 + (9*8)*100^2 = 200 + 260*100 + 72*100^2 = 746200

and hence

910 # 820 [M=1] = 260 + 72*100 = 7460

and

910 # 820 [M=2] = 72

If M is not too large, the digits of A # B [M] approximate the most significant digits of the product A * B well, so this operation can be used in multiplying mantissas of floating point numbers (Multiplying the mantissas exactly would result in too many digits, and a lot of them woud be meaningless anyway).

Input

All numbers in input and output should be nonnegative integers in decimal notation. The first line of the input is the number N of test cases. Then each of the following N lines has either the format

c A B

or

c A B M

where c is one of the characters '<', '+', '-', '*', '#', '/', describing the arithmetic operation to be performed on the numbers A and B (and possibly M). The second format (and hence the number M) is used only when c = '#'. We emphasize again that in the tests we have, it is assumed that BASE = 100.

Output

The output should consist of N lines, with each line containing the result of the arithmetic operation in the corresponding line of the input.

- For division, the output is 2 integers A / B and A % B, separated by a space. If B = 0, then return 0 0 (two zeroes).
- For all other operations, the ouput is one integer.
- For '<', the output should be 1 if A < B, 0 if A = B, and -1 if A > B.
- In case of subtraction A - B with A < B, the ouput should be 0.

Example

Input:

```
15
< 1000 1999
< 9898 9898
< 1234 123
< 0 0
+ 1791593436984766559642626609333245051871 307634751338542477034677711517175
- 1000000000000000000000000000000000000000000000 1
- 9851454318615645743724544440468029862 40153566442954896526103872421591156
* 333333333 0
* 1 3333333335555
* 9999999999999999999999999999999999999999 999999999999999999999999999999999999999
/ 1 999999999999999999999999999999999999999999999
/ 999999999999999999999999999999999999999999999 2
/ 861564574372454444046802986230763475133854247703 6442954896526103872421591156
# 1010 2020 1
# 619665458372652911688999608 5642672187122910629212227277 6
```

Output:

```
1
0
-1
0
1791593744619517898185103644010956569046
999999999999999999999999999999999999999999999
9811300752172690847198440568046438706
0
```

3333333335555

99999999999999999999999999999999999999989000000000000000000000000000000000000001

0 1

4999999999999999999999999999999999999999999999 1

133721962703322764598 5829723863328422309867552415

20400

3496569047280138337581751280571264068913704