Code Jam 2017 - World Finals

Operation

Problem

Here at Code Jam, we love to play a game called "Operation". (No, it has nothing to do with surgery; why would you think that?) The game is played with cards, each card is labeled with a basic arithmetic operation (addition, subtraction, multiplication or division) $\mathbf{O_i}$ and an integer right operand $\mathbf{V_i}$ for that operation. For example, a card might say `+ 0`, or `- -2`, or `/ -4` — note that operands can be negative or zero, although a card with a division operation will never have 0 as an operand.

In each round of the game, a starting integer value **S** is chosen, and a set of **C** cards is laid out. The player must choose an order for the cards, using each card exactly once. After that, the operations are applied, in order, to the starting value **S**, and a final result is obtained.

Although all of the operands on the cards are integers, the operations are executed on rational numbers. For instance, suppose that the initial value is 5, and the cards are `+ 1`, `- 2`, `* 3`, and `/ -2`. If we put them in the order given above, the final result is (5 + 1 - 2) * 3 / (-2) = -6. Notice that the operations are performed in the order given by the cards, disregarding any operator precedence. On the other hand, if we choose the order `- 2`, `/ -2`, `+ 1`, `* 3`, the result is ((5 - 2) / (-2) + 1) * 3 = -3 / 2. That example turns out to be the maximum possible value for this set of cards.

Given a set of cards, can you figure out the maximum possible final value that can be obtained? Please give the result as an irreducible fraction with a positive denominator.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each case begins with one line with two integers **S** and **C**: the starting value for the game, and the number of cards. Then, **C** lines follow. The i-th of these lines represents one card, and contains one character $\mathbf{O_i}$ representing the operation (which is either `+`, `-`, `*`, or `/`) and one integer $\mathbf{V_i}$ representing the operand.

Output

For each test case, output one line containing `Case #x: y z`, where `x` is the test case number (starting from 1), and `y` and `z` are integers such that `y`/`z` is the maximum possible final value of the game, `y` and `z` do not have common divisors other than 1 and -1, and `z` is strictly greater than 0.

Limits

Memory limit: 1 GB.
$1 \le \mathbf{T} \le 100$.
$-1{,}000 \le \mathbf{S} \le 1{,}000$.
$\mathbf{O_i}$ is one of `+`, `-`, `*`, or `/`, for all i.
$-1{,}000 \le \mathbf{V_i} \le 1{,}000$, for all i.
If $\mathbf{O_i}$ = `/`, then $\mathbf{V_i} \neq 0$, for all i.

Small dataset (Test Set 1 - Visible)

Time limit: 60 seconds.
1 ≤ **C** ≤ 15.

Large dataset (Test Set 2 - Hidden)

Time limit: 120 seconds.
1 ≤ **C** ≤ 1000.

Sample

Sample Input

```
5
1 2
- 3
* 2
5 4
+ 1
- 2
* 3
/ -2
1000 7
* -1000
* -1000
* 1000
* 1000
* 1000
* 1000
* 1000
-1 3
- -1
* 0
/ -1
0 1
+ 0
```

Sample Output

```
Case #1: -1 1
Case #2: -3 2
Case #3:
1000000000000000000000000 1
Case #4: 1 1
Case #5: 0 1
```

In Sample Case #1, the optimal strategy is to play the `* 2` card before the `- 3` card, which yields a result of -1. The unique rational expression of this as specified in the problem is -1 1.

Sample Case #2 is the one described in the third paragraph of the problem statement.

In Sample Case #3, we get the same answer regardless of the order in which we use the cards. Notice that the numerator of the answer is too large to fit in 64-bit integer.

In Sample Case #4, the largest result we can achieve is 1. One way is: `/ -1`, `* 0`, `- -1`.

In Sample Case #5, note that the only valid representation of the answer is `0 1`. `0 2` is invalid because it can be reduced. `0 -1` is invalid because the denominator must be positive.