Arithmetic

A common sort of puzzle or problem one sees in newspapers, particularly at the beginning of each year, goes along the lines of "make up a formula whose value is 2017, using the numbers 1 to 10 inclusive in that order". Of course the rules of the game are never entirely clear—in particular whether parentheses are allowed, and what operations may be used.

In this étude we'll look at a simplified model. The only operations allowed are $+$ and $\times$ and parentheses are not permitted. However, we'll consider both "proper" order of operations where

$$1 + 2 \times 3 = 7$$

and "left to right" where

$$1 + 2 \times 3 =_{LR} 9.$$

Task

Input from `stdin` will be a sequence of scenarios. Each scenario is exactly two lines. The first line is the set of numbers to use, with a space separating any two of these numbers and the second line consists of a target value, followed by a space, followed by the character `N` or the character `L` indicating whether normal order of operations, or left to right order is to be assumed.

There will be no more than 25 numbers in the first line of any scenario. For a "super-sized" version of this étude, you might increase this limit significantly.

The output for a scenario should be a character representing the order used, a space, and then an expression of the required value (using the character `*` for multiplication, and with spaces before and after each operation), or the word `impossible`.

Answers that are incorrect due to failure (overflow) of your programming language's integer arithmetic are incorrect.

Example

Input:

```
1 2 3
7 N
1 2 3
9 L
1 2 3
100 N
```

Output:

```
N 1 + 2 * 3
L 1 + 2 * 3
N impossible
```

(2 point, Individual)

Notes

The relevance of this étude is analysis. If there are n numbers that must be used in a fixed order, and 2 different operators that can be placed at each of $n - 1$ places between numbers, how many possible expressions are there?

There is also a programming aspect to this. Two people wrote model answers for this étude. One used 200 lines of Java, taking four classes, one of them an `Operator` class. The other used 60 lines of C, with `char` for operators. Is there such a thing as over-using classes?

There is also a software engineering question about efficient use of your time. Given the amount of time you have available, how hard should you work on this problem? If you have a solution that works, should you look for a better one, or go on to the next étude?

If you write a "smart" program for some problem, it can be hard to tell whether you have a fault in your program or in your test data. So it's useful to quickly write a brute force program so that you can test your test data. That's why this problem has been limited so that a brute force solution will work. But this pays off only when your brute force solution is obviously right. So you want to have a *simple* program here.

Testing

As usual, it is a good idea to test your program with inputs where you know the outputs. For example, let n be any integer. What should

n

n N

n

n L

do? If $n > 0$, what should

n

0 N

do? If $a + b + \cdots + n < p$ and $a \times b \times \cdots \times n < p$, what should

$a\ b\ c \ldots n$

p L

do? Is this legal?

100 -55 0

0 N

Suppose you have a procedure
evaluate : (bool NL, []int nums, []bool ops) $\rightarrow$ int
where NL says whether to use Normal or Left-to-right rules, nums is an array of n integers, and ops is an array of $n-1$ booleans with true meaning $\star$ and false meaning +, and the result is the value of the expression. Then you can generate random test cases like this:

```
let $n$ be a random integer between 1 and 24.
let NL be a random Boolean.
let nums be an array of $n$ random integers.
let ops be an array of $n - 1$ random Booleans.
let $v$ be evaluate(NL, nums, ops).
print the elements of nums on one line.
print $v$ followed by (N or L) on another line.
print the elements of ops in a second file.
```

Of course, this only generates test cases that have solutions. How might you generate test cases that have the answer `impossible`?

Hint: If nums = (2, 3) and ops = (false), that gives us 2+3=v=5. Adding 1 to v gives us 2?3=6, which has the answer ?=`*`. The same example shows that subtracting 1 from v doesn't always give us an `impossible` problem either.

Suppose the tester provided the scenario

```
2 2
4 N
```

intending 2+2=4, and your program reported

```
N 2 * 2
```

Would your program be wrong? What does this tell us about testing this problem?