Whether it's healthcare or programming, at Rally Health we've learned that there are a lot of ways to solve a problem. Whether it's parsing JSON or helping people make healthier choices, there's no single right way and each solution can often new insights overall.

In that spirit, for this problem, you'll be asked to look at numbers a little bit differently than you may be used to.

Write a program that accepts an integer (N) and sequences of digits, and fills in 0 or more operators (+, -, and * -- we're disallowing division for this exercise) to that sequence so that a mathematical expression is generated that totals N. For example, consider the input sequence 2100100. Operators can be inserted in a variety of ways that would evaluate to 2000. Here are some examples:

2100-100
2*100*10+0
2*100*10-0

Operators should only be inserted such that they do not violate the following rules:

- the operators are strictly binary, that is, "minus" and "plus" are never unary. For example, 2*-100*-10+0 is not a valid answer.
- no nonzero numbers are written using a leading zero. For example, 2*10*0100 would not be an answer, even though it adds to N.
- zero is never listed as more than a single '0'. For example, given 21000000, you should not produce 2*1000+000, even though it adds to 2000.
- normal precedence is followed. That is, multiplication before addition/subtraction, otherwise, left to right. For example, given 5007502, one possible answer is 500+750*2, because it would be interpreted as 500+(750*2) which is 2000.
- Inserting parentheses to modify operator precedence is not allowed. For 2500500, for example, 2*(500+500) is not allowed.

The first line of input will contain a single integer N. For each additional line, the program should output ALL legal sequences calculating to N with operators inserted, one per line. If none exist, it should output the word "impossible." At least one line should be skipped between output sets.

Sample input:

2000
2100100
77

Corresponding Output:
2100-100
2*100*10+0
2*100*10-0

impossible