“Increase my killing power, eh?”
— *Homer Simpson*

You are probably familiar with the binary representation of integers, i.e. writing a nonnegative integer n as $\sum a_i 2^i$, where each a_i is either 0 or 1. In this problem, we consider a so called *signed binary* representation, in which we still write n as $\sum a_i 2^i$, but allow a_i to take on the values -1, 0 and 1. We write a signed binary representation of a number as a vector $(a_k, a_{k-1}, \ldots, a_1, a_0)$. For instance, $n = 13$ can be represented as $(1, 0, 0, -1, -1) = 2^4 - 2^1 - 2^0$.

The binary representation of a number is unique, but obviously, the signed binary representation is not. In certain applications (e.g. cryptography), one seeks to write a number n in signed binary representation with as few non-zero digits as possible. For example, we consider the representation (1, 0, 0, -1) to be a better representation of $n = 7$ than (1, 1, 1). Your task is to write a program which will find such a minimal representation.

Input

The input consists of several test cases (at most 25), one per line. Each test case consists of a positive integer $n \leq 2^{50000}$ written in binary without leading zeros. The input is terminated by a case where $n = 0$, which should not be processed.

Output

For each line of input, output one line containing the signed binary representation of n that has the minimum number of non-zero digits, using the characters ‘`-`’ for -1, ‘`0`’ for 0 and ‘`+`’ for +1. The number should be written without leading zeros. If there are several possible answers, output the one that is lexicographically smallest (by the ASCII ordering).

Sample Input

```
10000
1111
10111
0
```

Sample Output

```
+0000
+000-
++00-
```