Waterloo 2013 Fall B - Java vs. Scheme Smackdown

Time limit: 1.0s **Memory limit:** 64M

Edgar thinks he knows it all.

He has programmed in Java before, and he believes all numbers on a computer are stored inexactly. Moreover, Java programs are mostly composed of dots: `System.out.please.oh.please.println()` for example. Edgar thinks that all numbers between 0 and 1 in Java are stored only partially, and the rest of the number is a sequence of dots, since it is Java. For example, the number $\frac{1}{6}$ is stored in Java as $0.1666\ldots$. However, after Edgar is enlightened by the beauty of Scheme, he realizes that numbers can be stored exactly. He needs to rewrite his Java programs to use this exact representation.

In order to make this task feasible, he assumes that the original fraction is always the simplest one that produces the given sequence of digits; by *simplest*, he means the the one with smallest denominator. Also, he assumes that Java always stores enough important digits; no digit from the repeating portion of the decimal expansion was left unrecorded (even if this repeating portion was all zeroes).

Input Specification

There are several test cases. For each test case there is one line of input of the form $0.dddd\ldots$ where $dddd$ is a string of 1 to 9 digits, not all zero. A line containing `0` follows the last case. For each case, output the original fraction.

Note that an exact decimal fraction has two repeating expansions (e.g. $\frac{1}{5} = 0.2000\ldots = 0.19999\ldots$).

Sample Input

```
0.2...
0.20...
0.474612399...
0
```

Sample Output

```
2/9
1/5
1186531/2500000
```