

Juggling Numbers

Program Name: `juggle.java` Input File: `juggle.in`

Did you ever wonder how jugglers can keep track of all of those balls while not letting any of them fall? They do it by mastering a certain number of basic throws and then chaining them together in exciting ways.

One convenient side-effect of this technique is that it is possible to represent a juggling pattern fairly well with a string of single digits such as “5313” or “441441”. Each digit represents a single throw, with the height of the throw corresponding to the size of the number (the exception is the 0 digit, which represents no ball is thrown during that step). For instance, a ball thrown with a height of 5 will have to be caught five steps later in the sequence. Not all sequences are possible for jugglers, however, since they can only catch one ball during any given step. In the sequence “321” all three balls would be landing during step 4! It’s very useful to be able to determine whether or not a sequence is possible.

Input

The first line of input will contain a single integer n indicating the number of datasets.

The following n lines will each contain a sequence of digits (0-9) representing a juggling pattern. Each sequence will contain from 1 to 40 digits.

Output

For each dataset in the input, determine if there is any step where more than one ball must be caught. If there is no such step, then the sequence is valid and the string “VALID” should be displayed. Otherwise, for sequences with steps where multiple balls have to be caught simultaneously, the sequence is invalid, and we want to know the number of balls that will drop the first time the juggler misses. This value should be one less than the total number of balls that need to be caught during the first step where the juggler has to catch more than one. In the case of an invalid sequence, display “DROPPED X on step Y ”, where the first drop occurs on step Y , with X balls missed, and steps are numbered starting at 1.

Note the sequences will likely end with some balls still in the air. Solutions should treat this situation as if the sequence ended in just enough zeros (“0”) to ensure all balls were caught.

Example Input File

```
4
333333333
441441441441
333321
445441441441
```

Example Output To Screen

```
VALID
VALID
DROPPED 2 on step 7
DROPPED 1 on step 8
```