# Problem 5—Hailstone Sequences
*written by David Powers*

No one takes Mushmouth seriously because of his speech impediment. Most people, in fact, would be quite surprised to learn that he is a mathematical genius. Mushmouth, in particular, has been making a study of hailstone sequences, or, as he calls them, “hailstone-buh sequences-buh.”

The following algorithm defines this sequence:

1. Start with any positive integer, N.
2. If that number is odd, then multiply it by three and add one; otherwise, divide it by two.
3. Repeat as many times as desired.

The hailstone sequence probably eventually reaches the pattern 4-2-1 (which then loops indefinitely), although this has never been proven. Determine the length of the hailstone sequence up to the point where the first 1 is reached (counting both the starting number and the first 1).

**INPUT SPECIFICATION.** Each input case consists of an unsigned decimal integer. The last integer in the file is a –1. This –1 is not to be processed; it merely signifies the end of input. There may be any number of spaces and/or **<EOLN>**'s preceding, following, or separating these integers.

**OUTPUT SPECIFICATION.** The output cases should appear in the same order as their corresponding input cases. For each case, you should print “Case *c*: length = *l* for N = *N*” followed by two **<EOLN>**'s, where *c* is the case number, *N* is the input number, and *l* is the length of the sequence.

**SAMPLE INPUT.**

```
1<EOLN>
5<EOLN>
-1<EOLN>
<EOF>
```

**SAMPLE OUTPUT.**

```
Case·1:·length·=·1·for·N·=·1<EOLN>
<EOLN>
Case·2:·length·=·6·for·N·=·5<EOLN>
<EOLN>
<EOF>
```