1. Evaluate the following expressions using Python's rules of precedence:
   (a) `3 + 3 * 3 * 3 + 3 + 3 * 3`
   (b) `10 ** ((9 + 8) % 7) + 6 * (5 + 4 + 3 // 2 + 1)`

2. Write a Python function `max3(a, b, c)` that returns the largest of its three arguments, but does so without calling any built-in Python functions. Later in the semester, you will learn how to write a version of `max` that takes any number of arguments, similar to the built-in library version does.

3. As far as we know, the only computer scientist to win a Pulitzer Prize is Douglas Hofstadter, who won one for *Gödel, Escher, Bach*, a delightful exploration of the intricacies of computation. Hofstadter's book includes many famous mathematical puzzles, including this one:

   - Pick some positive integer and call it n.
   - If n is even, divide it by two.
   - If n is odd, multiply it by three and add one.
   - Continue this process until n is equal to one.

   On page 401 of the Vintage edition, Hofstadter illustrates this process with the following example, starting with the number 15:

15	is odd, so I make $3n + 1$:	46
46	is even, so I take half:	23
23	is odd, so I make $3n + 1$:	70
70	is even, so I take half:	35
35	is odd, so I make $3n + 1$:	106
106	is even, so I take half:	53
53	is odd, so I make $3n + 1$:	160
160	is even, so I take half:	80
80	is even, so I take half:	40
40	is even, so I take half:	20
20	is even, so I take half:	10
10	is even, so I take half:	5
5	is odd, so I make $3n + 1$:	16
16	is even, so I take half:	8
8	is even, so I take half:	4
4	is even, so I take half:	2
2	is even, so I take half:	1

   As you can see in this example, the numbers go up and down, but eventually–at least for all the numbers that have ever been tried–comes down to end in 1. In some respects, this process is reminiscent of the formation of hailstones, which get carried upward by

the winds over and over again before they finally descend to the ground. Because of this analogy, this sequence of numbers is usually called the **Hailstone Sequence**, although it goes by many other names as well.

Write a function `hailstone` that takes an integer as an argument and then displays the Hailstone sequence for that number, one number on each line. At the end, your function should *return* the number of steps it took the input value to make its way all the way down to 1. For example, your program should be able to reproduce the following Python REPL session:

```
>>> steps = hailstone(17)
17
52
26
13
40
20
10
5
16
8
4
2
1
>>> print(steps)
12
```

The fascinating thing about this problem is that no one has yet been able to prove that it always stops. The conjecture that this process always terminates is called the **Collatz conjecture**, and appears in the following XKCD comic by Randall Munroe: