

# Comp125 Homework 2

Author: Ayca Tuzmen

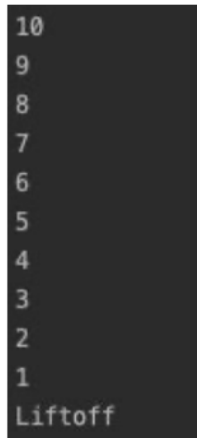
## Important Note:

For each problem, we give you specific guidelines on how to begin decomposing your solution. You can (and should) add additional functions for decomposition.

---

### Question 1

Write a program that prints out the calls for a spaceship that is about to launch. Countdown the numbers from 10 to 1 and then write "Liftoff." You must use a **range()** for loop, and the output should look as it does in Figure 1.



```
10
9
8
7
6
5
4
3
2
1
Liftoff
```

Figure 1. The output of your program in **liftoff.py** .

---

### Question 2

Douglas Hofstadter's Pulitzer-prize-winning book *Gödel, Escher, Bach* contains many interesting mathematical puzzles, many of which can be expressed in the form of computer programs. In

Chapter XII, Hofstadter mentions a problem that is well within the scope of the control statements we have learned so far. The problem can be expressed as follows:

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 from this example, the numbers go up and down, but eventually—at least for all 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 wind 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.

In **hailstone.py**, write a program that **reads** in a number from the user and then displays the Hailstone sequence for that number, just as in Hofstadter's book, followed by a line showing the number of steps taken to reach 1. Note that all of the numbers in the program are integers, not floats. Your code should be able to produce a sample run that looks like Figure 2.

```
Enter a number: 17
17 is odd, so I make 3n + 1: 52
52 is even, so I take half: 26
26 is even, so I take half: 13
13 is odd, so I make 3n + 1: 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
The process took 12 step(s) to reach 1.
```

**Figure 2** : A sample output of your program in **hailstone.py**.

The fascinating thing about this problem is that no one has yet been able to prove that it always stops. The number of steps in the process can certainly get very large. How many steps, for example, does your program take when  $n$  is 27?

---

## Question 3

You decide to buy some stocks for a certain price and then sell them at another price. Write a program that determines whether or not the transaction was profitable. Here are the details:

- Take three separate inputs: the number of shares, the purchase price of the stocks, and the sale price, in that order.
- You purchase the number of stocks determined by the input.
- When you purchase the stocks, you pay the price determined by the input.
- You pay your stockbroker a commission of 3 percent on the amount paid for the stocks.
- Later, you sell the all of the stocks for the price determined by the input.
- You pay your stockbroker another commission of 3 percent on the amount you received for the stock.

Your program should calculate your net gain or loss during this transaction and print it in the following format:

If your transaction was profitable (or if there was a net gain/loss of 0) print:

"After the transaction, you made 300 dollars."

(If, for example, you gained 300 dollars during the transaction.)

If your transaction was not profitable, print:

"After the transaction, you lost 300 dollars."

(If, for example, you lost 300 dollars during the transaction.)

Use string formatting.

A Sample Output:

```
Enter number of shares:123
Enter purchase price:12.3
Enter sale price:12.8
After the transaction, you lost 31.12 dollars.
```