Problem Set 2 Exercise #11: Collatz Conjecture

Reference: Lecture 4 notes

Learning objectives: Repetition statements; Modular design

Estimated completion time: 25 minutes

Problem statement:

There is simple yet still unresolved question in computing, known as the Collatz Conjecture. The conjecture revolves around the following computation:

$$f(n) = \begin{cases} \frac{n}{2}, & if\ n\ is\ even \\ 3 * n + 1, & if\ n\ is\ odd \end{cases}$$

The Collatz Conjecture states that:

If we repeatedly apply the computation above, using the result of f(*n*) as the next *n*, the process will *always* reduce to the value of 1.

For example, if *n* is 6:

f(6) = 3 (6 is even, 6/2 = 3)

f(3) = 10 (3 is odd, 3*3+1 = 10)

f(10) = 5

f(5) = 16

f(16) = 8

f(8) = 4

f(4) = 2

f(2) = 1 (Reached!)

It is very fascinating to note that the value of *n* has no known relationship with the number of steps required to reach the value of 1. For example, *n* = 6 takes 8 steps but *n* = 27 takes 111 steps! The number of steps required to reach the value of 1 is known as the **cycle length**.

Write a program **PS2_Ex11_CycleLength.java** that (1) reads two positive integers *low* and *high*; (2) calculates the cycle length for every number in the range [*low*, *high*] (both inclusive); (3) prints out the largest cycle length in the given range.

For example, sample run #1 (next page) shows that the largest cycle length among all the integers in the range [3, 6] is 8 (for *n* = 6).

Modular design makes your programming more manageable. You are advised to divide the given task into several sub-problems and define methods to solve sub-problems one by one.

Notes:

This task is a continuation of Exercise #10, in which you take a natural number and compute its cycle length. Therefore you may reuse the code written in that exercise as the building block of this exercise.

After attempting these two exercises, you should reflect and have a better understanding of modular design – how it makes your logic clearer and coding incremental.

Sample run #1:

```
Enter range: 3 6
Maximum cycle length = 8
```

Sample run #2:

```
Enter range: 20 27
Maximum cycle length = 111
```