Lab Exercise # 15

Warning: This lab is long because it contains many important exercises to get the hang of using loops. Do not worry if you can’t finish it during the lab period, but you should finish any exercises you can’t complete as (ungraded) homework.

If you did not do the loop exercises with algorithms first (lab #14), you should start there!

Exercise 1: Printing Numbers

Write an algorithm and then a Java program using a while loop to print all the numbers from 1 to 1000.

Exercise 2: Printing Numbers Again

Write an algorithm and then a Java program that asks the user to enter two integer numbers *low* and *high.* Your program should print all the numbers from *low* until *high.* If low is more than high, your program should print a message and then exit.

Exercise 3 part a: Printing Numbers YET AGAIN

Extend the program from the previous exercise to let the user enter a 3rd number *increment*. Your program should print every number from *low* until *high* still, but it should count by increments of the 3rd number. You may assume that increment is a positive number. For example, if low is 10, high is 20, and *increment* is 3, your program should print 10 13 16 19

Exercise 3 part b: Modularizing printing

Modularize your program from exercise four part a, by separating the code so that there is a method *printArithmeticSeries*(int start, int end, int increment) which does all the printing. Your main method should only be left with reading from the user with Scanner and then calling the method *printArithmeticSeries*.

Exercise 3 part c: Geometric Series

Add to your program a method *printGeometricSeries* which takes as input the same 3 values as *printArithmeticSeries*. Instead of incrementing at each step by *increment,* you should *multiply* by increment at each step. For example *printGeometricSeries*(10,1000,2) should print

10 20 40 80 160 320 640

Call your method from the main method also.

Exercise 3 part d: Combining the methods

There is a lot of code duplication between part b and c. Can you figure out a way to combine the 2 methods into 1? Hint: Use a 4th Boolean parameter.

Exercise 4: Factorial

A factorial of a number n is written as n! and is defined as:

n! = 1 \* 2 \* 3 \* … \* (n-1) \* n

For example, 3! = 1\*2\*3 = 6. Likewise 7! = 1\*2\*3\*4\*5\*6\*7 = 5040

Write an algorithm and then a method that takes in a whole number and returns the corresponding factorial.

Now in the main method, ask the user for a number between 1 and 20, send this as an argument to the factorial method and display the result.

Bonus: If the user gives you a number outside the range, keep asking them for correct input (this is your first foray into input validation!).

Exercise 5: Loop for fixed length series

Write an algorithm and then a method that calculates and returns the sum of the following series of numbers:

1/30 + 2/29 + 3/28 + …. 29/2 + 30/1

Think about how you code this – you want it to be very easy to change, so that if I change the specifications to use the number 20 instead of 30, you can easily make the change.

If your answer is a whole number … did you get caught in integer division??

Exercise 6: Count and average

Write a method that repeatedly asks the user to enter a positive number, until they type -1. The application counts the number of entries and calculates the sum. It returns the average.

Exercise 7: The Fibonacci series

The Fibonacci series is defined as:

* The first 2 numbers are equal to 1.
* Every subsequent number in the series is taken by adding the two previous numbers.

For example:

1,1,2,3,5,8,13,21,34,55, etc

Write an algorithm and then a method that takes as input an integer n and returns the nth Fibonacci number. For example Fibonacci(3) should return 2 since the 3rd Fibonacci number is 2. Fibonacci(4) should return 3, and Fibonacci(5) should return 5.

Exercise 8: Generalized Fibonacci series

Why start with the numbers 1 and 1? It turns out that you can generalize the Fibonacci series to start with any 2 numbers. For example,

4,3,7,10,17,27,44 ,….

Is an example of a Fibonacci series.

Modify your method from exercise 7 so that it takes as input *three* values. The first value should represent the same thing as before, the 2nd value should represent the first number of the new Fibonacci series (4 in the above example) and the last number should represent the 2nd number of the new Fibonacci series (3 in the above example)