CMSC 113: Computer Science I
Homework: Loops
due on Gradescope by the beginning of class on Wednesday, March 8, 2018

1. This program is based on primality testing. Remember, a number is prime if and only if the only numbers that divide evenly into it are 1 and the number itself. The first few prime numbers are 2, 3, 5, 7, 11, 13, 17, …

   It is important to note that it is **not** sufficient to test if a number is divisible by 2, 3, and 5 to see if it is prime. For example, 49 is not divisible by 2, 3, or 5, and it is not prime. This does not mean that checking for 2, 3, 5, and 7 is sufficient either. No finite amount of numbers is sufficient; you must use a loop.

   Your program can be fairly short. (For example, even with the extra challenge, mine is 45 lines, including some comments.) Think before you code! Primality testing is more involved than other programs we have seen. You may wish to examine how you, as a human, would determine whether a number is prime. Duplicate that process in your program.

   Write a program in *Prime.java* that asks the user for a number and then prints whether that number is prime. If the number is not prime, include in your printout two integers (neither of which should be 1) that multiply to the number entered.

   Regardless of whether the number is prime, your program should print only one result; it's easy to make a mistake coding this that prints out `Prime` many times before printing out the factors of a composite number. A program that does this does not get full credit.

   Extra challenge: Write a program *in Factors.java* that prints out the prime factors of a number entered by the user. The prime factors of a number are a bunch of prime numbers such that when they are all multiplied together, they yield the number you started with. For example, the prime factors of 24 are 2, 2, 2, and 3. The prime factors of 66 are 2, 3, and 11.

2. Write a program in *Reverse.java* that asks the user how many numbers they have and then collects that many numbers in an `ArrayList`. Create a new, separate `ArrayList` that stores the *reverse* of the original `ArrayList`. Print out the contents of the new `ArrayList`. Note: *do not* just print out the values in reverse order; make sure to create the reversed `ArrayList` before printing.

   Here is an example session:

   ```
   How many numbers? 4
   Enter a number: 13
   Enter a number: 2
   Enter a number: 7
   Enter a number: 5
   Reversed:
   5
   7
   2
   13
   ```