CSC1003 Practice Outline Oct10-14

Programming exercises about conditionals and loops.

1. Write a Java program to compute the sum of the first N positive even numbers. N can be input from the command argument. Suppose N = 2, then the sum will be 2 + 4 = 6. If N = 3, then the sum will be 2 + 4 + 6 = 12. You can use the following template to implement this Java program.

```
1  public class SumofEvenNum
2  {
3     public static void main(String[] args)
4     {
5        int N = Integer.parseInt(args[0]);
6        int sum = 0;
7
8        // write a while loop or for loop to compute the sum
9
10
11
12
13       System.out.println(sum);
14    }
15 }
```

You can also modify the previous program to compute the sum of the first N positive odd numbers.

2. Write a Java program to compute the greatest common divisor of two given positive integers A and B. You can assume the two positive integers are not larger than 1,000,000. For example, the highest common factor of 54 and 24 is 6.

   Hint: You can use for/while loop(s) and if statement(s) to write this program.

3. Write a program that takes an integer command-line argument n and prints all the positive powers of 3 less than or equal to n.

4. Modify `Factors` to print just one copy each of the prime divisors. Currently, executing Factors with argument 98 gives us:

```
% java Factors 98
% 2 7 7
```

After modifications, the output should be

```
% java Factors 98
% 2 7
```

```
public class Factors
{
   public static void main(String[] args)
   {
      long n = Long.parseLong(args[0]);
      for ( int i = 2; i <= n/i; i++)
      {
         while (n % i == 0)
         {
            System.out.print(i + " ");
            n = n / i;
         }
      }
      if (n > 1) System.out.println(n);
      else       System.out.println();
   }
}
```

5. [Optional] Counting primes. Write a program `PrimeCounter` that takes an integer command-line argument n and finds the number of primes less than or equal to n. Use it to print out the number of primes less than or equal to 10 million. *Note: If you are not careful, your program may not finish in a reasonable amount of time!*

   Hint: To give an efficient program, you might need to use a boolean array.

6. [Optional] 1D random walk simulation. A one-dimensional random walk simulates the behavior of a particle moving in a sequence of points. At each step, the random walker moves left or right with probability equal to 1/2, independent of previous moves. Write a

program `RandomWalker` that takes an integer command-line argument n and estimates how long it will take a random walker to hit the boundary of this sequence centered at the starting point with a length of 2n+1.

For example, the below figure is the case where n = 2, and p = 1/2. The walker starts at 0, and stops when she/he reaches -2 or 2.