CS2030 Programming Methodology
Semester 1 2023/2024

25 & 26 October 2023
Problem Set #8
Java Streams

1. Write a method `omega` with signature `LongStream omega(int n)` that takes in an `int n` and returns a `LongStream` containing the first n omega numbers.

   Hint: A `LongStream` *is necessary as* `count` *returns a value of type* `long`.

   The i^{th} omega number is the number of distinct prime factors for the number i. The first 10 omega numbers are $0, 1, 1, 1, 1, 2, 1, 1, 1, 2$.

   The `isPrime` method is given below:

   ```
   boolean isPrime(int n) {
       return n > 1 && IntStream.range(2, n)
           .noneMatch(x -> n % x == 0);
   }
   ```

2. Write a method that returns the first n Fibonacci numbers as a `Stream<Integer>`.

   For instance, the first 10 Fibonacci numbers are $1, 1, 2, 3, 5, 8, 13, 21, 34, 55$.

   Hint: Use an additional `Pair` class that keeps two items in the stream

3. Write a method product that takes in two `List` objects `list1` and `list2`, and produce a `Stream` containing elements combining each element from `list1` with every element from `list2` using a `BiFunction`. This operation is similar to a Cartesian product.

   ```
   <T,U,R> Stream<R> product(List<? extends T> list1,
           List<? extends U> list2,
           BiFunction<? super T, ? super U, ? extends R> func)
   ```

   For example, the following program fragment

   ```
   List<Integer> list1 = List.of(1, 2, 3, 4)
   List<String> list2 = List.of("A", "B")

   product(list1, list2, (str1, str2) -> str1 + str2)
       .toList()
   ```

   gives the output

   ```
   [1A, 1B, 2A, 2B, 3A, 3B, 4A, 4B]
   ```