Streams Part 2

For all the exercises, start with a List of Strings similar to this:

- List<String> words = Arrays.asList("hi", "hello", ...);

1. Produce a single String that is the result of concatenating the uppercase versions of all of the Strings. E.g., the result should be "HIHELLO...". Use a single reduce operation, without using map.

2. Produce the same String as above, but this time via a map operation that turns the words into upper case, followed by a reduce operation that concatenates them.

3. Produce a String that is all the words concatenated together, but with commas in between. E.g., the result should be "hi,hello,...". Note that there is no comma at the beginning, before “hi”, and also no comma at the end, after the last word. Major hint: there are two versions of reduce: one with a starter value, and one without a starter value.

4. Make a static method that produces a List of a specified length of random numbers. E.g.:

- List<Double> nums = StreamUtils.randomNumberList(someSize);
  // Result is something like [0.7096867136897776, 0.09894202723079482, ...]

5. Make a static method that produces a list of numbers that go in order by a step size. E.g.:

- List<Integer> nums = StreamUtils.orderedNumberList(50, 5, someSize);
  // Result is [50, 55, 60, ...]

6. The lecture notes showed three ways to use streams to compute sums. Use one of the three variations, and compute the sum of some ints. Redo it in parallel and verify that you get the same answer both times.

7. Now, use streams similarly to compute the product of some doubles. Show that serial and parallel versions do *not* always give the same answer. (Note: this is a bit tricky, because it seems at first that multiplication is associative, as required by parallel reduce. And, it will be impossible to illustrate the result if you have a single-core computer.)