Learning objective: parallelize the reduce phase. The parallelization of the reduce phase is illustrated with an algorithm to estimate the value of $\pi$. 

Exercise 1: pi computation algorithm
What is the area of a circle with radius 1?
Let us draw a random point uniformly in the square defined by the coordinates $(0, 0)$ and $(1, 1)$. What is the probability that this point is in the circle of radius 1 and which center is $(0, 0)$?
By relying on the law of large numbers, specify an algorithm to estimate the value of $\pi$ by drawing successively $n$ random points.

Exercise 2: stream formulation with generate
Propose a stream algorithm with the method generate to estimate the value of $\pi$.

Exercise 3: stream formulation with range
The method generate leads to a stream that is unfortunately difficult to parallelize because it does not convey the size and the splitting step is thus inefficient.
Propose an alternative using the method range.

Exercise 4: tryAdvance
In the following two exercises, we consider the implementation of a custom spliterator to split and iterate over a stream while retaining the size information, which is not the case by default with generate and limit.
We assume the spliterator is initialized with the appropriate supplier and size (supplier and size).

```java
class MySpliterator<T> implements Spliterator<T> {
    long size;
    Supplier<T> supplier;

    MySpliterator(Supplier<T> supplier, long size) {
        this.size = size;
        this.supplier = supplier;
    }
}
```

Let us consider the following simplified prototype: boolean tryAdvance(Consumer<T> consumer). This function returns true if there are still elements in the stream, false otherwise. Its main purpose is to apply the consumer (with method Consumer.accept(T)) on a newly generated element (with method Supplier.get()) of the stream.
Propose an implementation of this method.

Exercise 5: trySplit
The method trySplit returns a new spliterator with half the elements or null if the stream cannot be split. Its prototype is Spliterator<T> trySplit(). Propose an implementation of this method.
Exercise 6: factorial
Write a stream that computes the factorial of $n$ in parallel.
Discuss its parallelization compared to the parallelization of computing the sum of the first $n$ numbers of the Fibonacci sequence.

Exercise 7: matrix multiplication
We assume that two matrices are given as in the following example:

```java
double[][] m1 = { { 4, 8 }, { 0, 2 }, { 1, 6 } };
double[][] m2 = { { 5, 2 }, { 9, 4 } };
```

In this case, the result of their product must be:

```java
double[][] res = { { 92, 40 }, { 18, 8 }, { 59, 26 } }
```

Propose an implementation with external iterations (with `for` loops).
Replace the most inner loop with a stream.
Propose a complete stream implementation of the matrix multiplication that relies on the method `toArray`, which is a terminal operation that transforms the stream into an array.
Propose and discuss multiple options to parallelize this stream.