Java streams- exercises

Software Development Methods

Master in Data Science and Scientific Computing

1. Define a stream operation that computes the number of elements in an instance of an **IntStream** without using **count()**. It is equivalent to **/+ ∘ α $\overline{1}$** .
2. Define a stream operation that when applied to an instance of an **IntStream** returns a stream with all elements incremented by one. It is equivalent to **α(+ ∘ [id, $\overline{1}$]).**
3. Define a stream operation that computes the summation of all positive elements in an instance of an **IntStream**. It is equivalent to **/+ ∘ α(> ∘ [id, $\overline{0}$] -> id; $\overline{0}$).**
4. Given the following piece of code:

```
int []a = IntStream.rangeClosed(0, 2)
                   .flatMap(x -> IntStream.rangeClosed(0, 2)
                                          .map(y -> x + y))
                   .toArray();
```

   a. Indicate the result produced in the array **a**.
   b. Explain the need to use **flatMap** instead of **map** in the second line.
5. Define a stream operation that when applied to an instance of an **IntStream**, returns a stream with only the positive values (zeroes and all negative elements removed) without using **filter()**. It is equivalent to **flatten ∘ α (> ∘ [id, $\overline{0}$] -> [id]; $\overline{<>}$).** Hint: try **flatMap()**.
6. Define a stream operation sequence based on a reduce operation to concatenate all strings in a **Stream<Strings>**. It is equivalent to **/concat**.