This document contains a presentation of the second week programming problem that differs slightly from the one given at the end of the lecture. It should ease the understanding of the solution given at the end of the document.

Sieve of Erathostenes

The goal of this exercise is to compute prime numbers using a functional variant of the *Sieve of Erathostenes*.

Informal idea

- Start with the integers beginning with `2`, which is the first prime.
- Eliminate all multiples of `2` from the list.
- The first element of the remaining list is `3`, which is prime.
- Eliminate all multiples of `3` from the list.
- Continue in the same way. At each step, the first number in the list is prime, and we eliminate all multiples of it.

Mathematical description The technique described above consists in defining a series of sets $(P_i)_i$ of integers by induction. For each i, the minimum of the set P_i is the i-th prime number p_i:

- We begin with $P_0 = \{x \mid x \geq 2\}$.
- The first prime number, p_0, is the least integer (greater than 0) which is in P_0, i.e. 2.
- We continue with $P_1 = P_0 \backslash \{x \mid p_0 \text{ divides } x\}$.
- The next prime number, p_1, is the least integer (greater than p_0) which is in P_1, i.e. 3.
- For each $i \geq 0$, $P_{i+1} = P_i \backslash \{x \mid p_i \text{ divides } x\}$ and p_{i+1}, is the least integer (greater than p_i) which is in P_{i+1}.

Implementation In functional programming a set of integers defined *by comprehension*, i.e. by an expression $\{x \mid P(x)\}$ where P is a predicate can be represented by a function \$P\$: int => boolean.

1. Define a function $first$ that returns the least integer greater than a given integer and that satisfies a given predicate.
2. Define P_0.
3. Define a looping function *iter* that calls itself recursively with successive elements of $(P_i)_i$ and $(p_i)_i$, i.e.
$$\begin{array}{rcl} iter(0, P_0) & \rightarrow & iter(p_0, P_1) \\ & \rightarrow & iter(p_1, P_2) \\ & \rightarrow & iter(p_2, P_3) \\ & \rightarrow & \ldots \end{array}$$
4. Adapt the function iter to get the n-th prime number.