Los *tests de primalidad* son ampliamente conocidos y utilizados en el mundo de la seguridad y, en concreto, en muchos de los cifrados asimétricos. Uno de esos tests de primalidad es la *criba de eratóstenes* que, dado un número n, y siguiendo el siguiente algoritmo en pseudocódigo, es capaz de determinar si un número es primo o no:

1. Si el número es dos entonces devuelve que es primo.
2. Si el número es divisible por dos entonces devuelve que no es primo.
3. Para $i = 3$ hasta $\lfloor \sqrt{n} \rfloor$ y aumentando de dos en dos, haga lo siguiente:
   (a) Si $n \mod i$ es cero, entonces devuelve que no es primo.
4. Devolver que es primo.

Esta versión, por desgracia, tiene una complejidad de $\mathcal{O}(n \log \log n)$. Sin embargo, podemos mejorar el tiempo de este test haciendo una versión concurrente del algoritmo propuesto anteriormente.

Por tanto se pide:

- Diseñe una versión secuencial del algoritmo en `eratostenesSecuencial.cpp` y mida el tiempo para determinar la primalidad de los 10000 primeros números naturales.
- Diseñe una versión concurrente del alcoritmo en `eratostenesConcurrente.cpp` y mida el tiempo para determinar la primalidad de los 10000 primeros números naturales.