Assignment 04 - Semaphores

The **dining philosophers** is a classic synchronization problem. N philosophers are sitting around a large round table with a huge hotpot at the center. However, there are exactly N chopsticks arranged around the table so that each philosopher shares the chopstick that is on the right (respectively on the left) with the neighbor on its right-hand (resp. left-hand) side.

Each philosopher P has the same repetitive behavior: P thinks for a while, then becomes hungry and takes the chopsticks to the left and to the right in order to eat. P stops eating after a while and puts the chopsticks back on the table to think before getting hungry again. This goes on indefinitely. In order to eat, a philosopher must have both chopsticks in hand at the same time.

Disregard the obvious health hazard presented by this behaviour, and focus on synchronization.

Download the coding canvas for the simulation.

A problem can occur when all philosophers are hungry simultaneously, and each of them picks up the chopstick to their left before picking up the chopstick to the right. The file `philosophers-bad.c` contains an implementation that can lead to this situation.
Question: What could happen? Explain your answer in the `README` file.

Each philosopher is represented by a separate process that has an identifier i (0 < i < N). All N philosopher processes share chopsticks in an array of N integers. A philosopher process i must obtain chopsticks i and (i + 1)% N in order to eat

Design your own solution to the Dining Philosophers problem, detail it in the `README` file, and implement it in the `philosophers.c` file.
Your solution must:

- enforce safe usage of the chopsticks: in other words, two philosophers sat next to each other can never eat at the same time;
- not lead to deadlock;
- allow maximum concurrency: an eating philosopher does prevent its two immediate neighbors from eating, but any other philosopher should be able to eat if the chopsticks to its left and right are available.

Warning. *You will find many possible solutions to this problem if you look it up on the internet; some of them good, others very much not so. Please stick to internet searches that help clarify the problem itself, and come up with your own solution.*