Some of the answers to these questions can be found using the lecture slides, the recommended textbooks or other sources (question marked with *). Some questions may have more than one possible answer, or be more or less open for discussion. Note that no answers (solutions) will be given to these questions, but if help is needed the assistants will be available to answer questions. The concepts marked with yellow are important and should be fully understood.

## Semaphores

1. Semaphore Basics[1]

   (a) What is a semaphore? What is it good for?

   (b) Give an example use case for a binary semaphore. How are binary semaphores called?

   (c) Name a case where integer binary semaphores are used.

2. Barrier problem

   (a) Describe the concept of a barrier and provide a use case where its usage would make sense.

   (b) Given the following code template used to solve the barrier problem, explain to usage of each variable.

```
N = number of threads
count = 0
mutex = Semaphore(1)
barrier = Semaphore(0)

// rendez-vous
...
// critical section
```

   (c) Provide a pseudo code solution for the barrier problem based on the provided template.

   (d) Now suppose that the critical section is in a loop. Assume that we want to make sure that a thread is only allowed to get to the next iteration if the other threads have finished the current one and are ready to proceed to the next iteration too. In other words, threads should synchronously execute the iterations in the loop. Given the following template, provide a pseudo code solution to this problem.

```
N = number of threads
count = 0
```

[1] We recommend that you have a look at the "Little Book of Semaphores", a free PDF copy can be found here: `http://www.greenteapress.com/semaphores/`

```
mutex = Semaphore(1)
barrier_0 = Semaphore(0)
barrier_1 = Semaphore(1)

// rendez-vous
...
// critical section
...
//go to rendez-vous
```

3. Browse, write, remove problem:

   Three kinds of threads share access to a singly-linked list: browsers, writer and removers. Browsers merely examine the list; hence they can execute concurrently with each other. Writers add new items to the end of the list; write operations must be mutually exclusive to preclude two writers from writing new items at about the same time. However, one write can proceed in parallel with any number of browsers. Finally, removers remove items from anywhere in the list. At most one remover process can access the list at a time, and removal must also be mutually exclusive with browsers and write operations.

   Provide a pseudo code solution for the browse, write, remove problem using semaphores.