

## OS 2021 Class Problem Sheet #4

### Problem 4.1:

(5 points)

Allen B. Downey describes the “Building  $H_2O$ ” problem in his book “The Little Book of Semaphores”. He states the synchronization problem as follows:

There are two kinds of threads, oxygen and hydrogen. In order to assemble these threads into water molecules, we have to create a barrier that makes each thread wait until a complete molecule is ready to proceed. As each thread passes the barrier, it should invoke `bond`. You must guarantee that all the threads from one molecule invoke `bond` before any of the threads from the next molecule do.

In other words:

- If an oxygen thread arrives at the barrier when no hydrogen threads are present, it has to wait for two hydrogen threads.
- If a hydrogen thread arrives at the barrier when no other threads are present, it has to wait for an oxygen thread and another hydrogen thread.

We don't have to worry about matching the threads up explicitly; that is, the threads do not necessarily know which other threads they are paired up with. The key is just that threads pass the barrier in complete sets; thus, if we examine the sequence of threads that invoke `bond` and divide them into groups of three, each group should contain one oxygen and two hydrogen threads.

Puzzle: Write synchronization code for oxygen and hydrogen molecules that enforces these constraints.

Implement a solution for this problem using POSIX threads. Make sure your program runs quietly under the Helgrind race condition checker:

```
$ valgrind --tool=helgrind ./h2o
```