Concurrency — Exercise 5
Locking 2

Prof. Dr. Oliver Haase

Problem 1

Using hand-over-hand locking, implement a concurrent, ascending list of int values. Program the `insert` and the `delete` operations, and a test program, such that

- within the `insert` operation, the currently inserted value is printed on the screen;
- within the `delete` operation, the currently deleted value is printed on the screen;
- the test program starts a first thread that writes the numbers 1, 2, 3, ... into a hand-over-hand locked list, sleeps after each `insert` operation for a second, and that is responsive to interruption;
- after ten seconds, the test program starts a second thread that deletes the values in the list in the same order 1, 2, 3, ..., sleeps after each `delete` operation for a second, and that also is responsive to interruption;
- after 1 minute, the test program cancels the two threads and terminates.

Problem 2

Rewrite the condition-based dining philosophers solution such that it uses intrinsic locks, `wait()` and `notifyAll()`. Why is this solution less efficient than using a `ReentrantLock` and `Condition` variables?

Problem 3

Modify the factorizer solution from exercise 3, problem 4 such that it uses a read-write-lock to guard the shared cache.

Have fun and good luck!