

## Exercise 10: Exercising in Style

### Task 1: Very Exclusive!

Throughout this exercise you are allowed to assume that registers can hold unbounded values, i.e., that overflows do not occur.

- Give a solution to mutual exclusion using a fetch-and-add register.
- Give a solution to mutual exclusion using a compare-and-swap register.
- Give a solution to mutual exclusion using a load-link/store-conditional register.
- Your solutions should all be obstruction-free. Can you prevent lockouts, too?

**Hint:** Give a generic solution that works for a) – c). There are plenty of different solutions. A generic one that uses only RW registers is to have a “want” flag for each node that it sets to 1 if it wants to enter the critical section, and then let whoever wins mutual exclusion close the bidding phase (no more new wants). Subsequently, switch to a mode that lets the nodes with raised flag each enter the critical section once and then return to the basic mutual exclusion algorithm.

### Task 2: Bonsai Splitter Tree

In this exercise, we construct a highly space-efficient randomized variant of the splitter tree from the lecture.

- Show that there is a (randomized) splitter such that each node that does not **stop** turns **left** or **right** with probability  $1/2$  each, independently of other nodes entering the splitter! Note that this allows, e.g.,  $k - 1$  nodes to turn **left**, or  $k - 1$  nodes to turn **right**. We still require that at most one node **stops** at the splitter, and if only one node enters the splitter it must **stop**.
- Show that for a tree of  $\Theta(n)$  leaves (constants are your choice), w.h.p., a constant fraction of all nodes obtains **stop** at some splitter.

**Hint:** Fix a node and show that regardless of what the other nodes do it **stops** in the tree with constant probability. Then it’s Chernoff time!

- Now iterate: Let all nodes that did not **stop** enter a second splitter tree, rinse, and repeat. Show that this way, you can achieve
  - $\mathcal{O}(\log k)$  expected step complexity for the first STORE of each node
  - $\mathcal{O}(\log^2 n)$  step complexity w.h.p. for the first STORE of each node
  - $\mathcal{O}(n)$  total space (this should suffice w.h.p.)
  - $\mathcal{O}(k)$  expected step complexity for COLLECT

**Hint:** For the space bound, just let the size of each new tree be by a constant factor smaller than the previous. For everything else, apply the results from the lecture and use probabilistic bounds where needed (for our randomized splitters all nodes might turn, e.g., **left**!).

### **Task 3\*: Stage Names**

- a) Find out what the *renaming* problem is!
- b) Can it be helpful with STORE & COLLECT?
- c) Do you think renaming is useful for mutual exclusion?
- d) What happens if we consider mutual exclusion with crash failures? Do things go south, or is there a way out?
- e) Present these newest trends in the TA session!