Exercise 11: Counting

Task 1: 1

The goal of this exercise is to understand the consistency properties of the bounded max register implementation from the lecture.

a) Show that if one always writes to $R_<$ if $i < M$, regardless of whether switch reads 0, the implementation is not linearizable!

   Hint: Start a read operation that reads 0 from switch, complete a write operation for $i \geq M$, then another one for $0 < i < M$. Show that the order implied by the "precedes" relation now is incompatible with any sequential execution of the max register!)

b) Show that if a write operation (for $i < M$) reads switch $= 1$, there is a preceding write operation for $i \geq M$. Conclude that it is always possible to determine a valid linearization point for such an operation.

c) Prove that the max register of maximum value $2M$ constructed from two max registers of maximum value M and a read/write register is linearizable.

   Hint: Divide operations into three classes: (i) writes of $i < M$ and reads reading switch $= 0$, (ii) write operations for $i < M$ reading switch $= 1$, and (iii) writes for $i \geq M$ and reads reading switch $= 1$. Order operations from classes (i) and (iii) first and then apply b) to handle those in class (ii).)

Task 2: 2

In this exercise, we're going to implement more powerful registers from weak ones. We start with very simple registers. They are

binary They can hold only values 0 and 1.

single-writer Only one node has write access.

single-reader Only one process has read access. This may be a different process than the one that has write access.

safe They guarantee that (i) 0 or 1 is returned, but (ii) they might return an arbitrary value while a write operation is in progress.[1]

All registers are initialized to 0 in this exercise.

Hint: Make sure to score easy points with g), even if earlier parts prove challenging.

a) Implement a *regular* binary single-writer single-reader register from a safe one. A regular register is a safe register that guarantees that only values of a concurrent or the latest preceding write are returned (or the initial value, if there is no preceding write).

b) Implement a regular M-*valued* single-writer single-reader register from M regular binary single-writer single-reader registers. An M-valued register can take values in $[M]$.

   Hint: Use the i-th register to represent value i. Read in ascending, but write in descending order.

[1] Note that because there is only a single writer, we can require that there is never more than one write in progress.

c) Implement a *linearizable* M-valued single-writer single-reader register that can be written $W-1$ times from a regular MW-valued single-writer single-reader register.

   Hint: Use timestamps, and have the reader always return the latest value.

d) An *n-reader* register is one that can be read by n different nodes. Show that naively using n atomic single-writer single-reader registers to construct a single-writer n-reader register does *not* result in a linearizable implementation.

e) Construct a linearizable M-valued single-writer n-reader register that can be written $W-1$ times out of n^2+n atomic MW-valued single-writer single-reader registers.

   Hint: Use timestamps and leverage the additional n^2 registers to communicate between the readers. The readers will read from "their" incoming registers, then from the writer's register, then write the timestamp/value pair of the maximum seen timestamp to their outgoing registers, and only then return the respective value.

f) Construct a linearizable M-valued n-writer n-reader register that can be written $W-1$ times out of n atomic MW-valued single-writer $2n$-reader registers.

   Hint: Let writers read all registers first and write with a timestamp larger than all timestamps they read.

g) Conclude that for any bounded number of operations, safe binary single-writer single-reader registers are as computationally powerful as atomic multi-valued multi-writer multi-reader registers.

   Hint: Concentrate on *not* thinking about efficiency. Seriously, do not think about efficiency. DO NOT THINK ABOUT EFFICENCY!

Task 3*: 3

Consider a fully connected asynchronous message passing system.

a) Implement a wait-free linearizable single-writer single-reader register!

b) It turns out that this didn't work. Why?

c) Check out what sort of simulations are around in the literature.

d) Write what you've learned to the green shared memory in the exercise session for everyone else to read!