Homework Assignment #8
Due: April 10, 2023 at 10:00 p.m.

Consider an asynchronous shared-memory model where any number of processes may crash. In the renaming problem, n processes initially have, as input, distinct names from the set $\{1, 2, \ldots, N\}$ and they must output distinct names from a smaller set $\{1, 2, \ldots, M\}$. All processes must run identical code. In class we saw an algorithm that solved renaming using read/write registers with $M = 2n - 1$.

(a) Prove that there is no deterministic, wait-free renaming algorithm with $M = n$ that uses only registers. Hint: Your answer can be very short.

(b) A sticky object stores one bit (which is either 0 or 1) and supports one operation: STICK, which sets the bit to 1 and returns the value that the bit had before the STICK operation. Give a simple, deterministic, wait-free renaming algorithm with $M = n$ using sticky objects. Briefly explain why your algorithm is correct. How many steps do processes take (in the worst-case) in your algorithm?

(c) In class, we saw a deterministic renaming algorithm with $M = O(n^2)$ using splitters (which were built from read/write registers). That algorithm took $O(n)$ steps per process. We will try to get a better running time (with high probability) using randomized splitters. A randomized splitter still has two read/write registers X and Y, both initialized to 0. When a process whose original name is *name* enters the splitter, it executes the following code.

```
1: function SPLITTER
2:     X ← name
3:     if Y = 0 then
4:         Y ← 1
5:         if X = name then
6:             return "stop"
7:         end if
8:     end if
9:     if flipping a fair coin comes up heads then
10:        return "left"
11:    else
12:        return "right"
13:    end if
14: end function
```

To design a renaming algorithm, we can build a binary tree of randomized splitters. The depth of a splitter is the length of the path from the root to that splitter. (The root has depth 0.) Splitters are labelled $1, 2, 3, \ldots$ in a pre-order traversal of the tree. Each process enters at the root and follows left or right child pointers as instructed by the splitters until a splitter tells the process to stop, and then the process outputs the label of that splitter.

(i) Explain why two processes can never output the same name. Your explanation should refer to the code for the splitter in detail.

(ii) Fix a schedule that describes the order in which processes take steps of the renaming algorithm. Consider the random experiment of running the algorithm using this schedule.

Show that, for any splitter at depth d, the expected number of processes that enter that splitter is at most $\frac{n}{2^d}$.

(iii) For the random experiment described in part (ii), pick a depth d such that the probability that process number 1 does not stop within one of the first d splitters it accesses is at most $\frac{1}{n}$. How big is M if all processes stop within the first d splitters they access?

Hint: an easy way to answer this question is to use Markov's inequality.