Homework 3 (due Friday, April 19)

1. *(Masking Fault-tolerant Leader Election)*

This homework exercise deals with the design of a masking fault-tolerant program for leader election in a ring.

Definition: A program is masking fault-tolerant if the occurrence of faults preserves its invariant.

Given: Consider processes in a ring. Each process has a unique identifier and can communicate with its predecessor and successor in the ring. Channel between processes are FIFO.

Problem:
a) Design a program for leader election that is masking in the presence of faults that add processes to the ring. (Assume that when a process is added to the ring, its predecessor and successor processes are instantaneously aware of the addition.)

b) Exhibit an invariant for your program.

2. *(Detection, Fairness)*

Given: Consider processes in a unidirectional ring. Each may send messages only to its successor and receive messages only from it predecessor. Channel between processes are of unbounded length and FIFO.

The system has a fixed but unknown number of tokens. Tokens are neither created nor destroyed. A token may be at a process or in a channel.

Problem: a) Design an *efficient* distributed program to be superposed on the underlying computation of the system by which some process determines the number of tokens in the system.

b) Give an invariant-based proof of safety and progress of the superposed computation.

c) Explain to what extent fairness is necessary (on a per action basis) in your solution.

d) Is your program correct if FIFO channels were replaced by causal-order preserving channels?

3. *(Byzantine Agreement)*

Given are four processes, one of which is called the "general". At most one of these processes is unreliable, i.e., that process executes nondeterministically in a manner that need not conform to its actions. No other faults exist in the system. Assume that all communications take bounded time and that the identity of the sender cannot be "forged".

Each process j has a local variable "byz.j", which is initially unassigned. The general has a local Boolean constant "x".

Problem: Design a terminating computation in which each process j writes its byz.j variable exactly once, so that eventually the byz variables of all reliable processes are identical; and, if the general is reliable, these variables have the value x.

Give a brief (informal or formal) proof of correctness.

You may introduce any number of local variables at each process. Hint: Consider a round-based computation. In each round, one or more processes may broadcast the value of any of their local variables to the other processes.

4. *(Clock Unison and Stabilization)*

Consider an undirected and connected graph of n nodes. Execution of nodes is assumed to be synchronous— in each step, every node u executes its program action (given below) with respect to one of its neighbors, that is chosen in a fixed cyclic ordering nxt of all of its neighbors.

---

variables of node u ::
  $x.u :\ 0..\ n^2-1$,
  $r.u$: whose value ranges over the set
      $\{\, v \mid (u,v) \text{ is an edge in the graph} \,\}$

action of node u ::
  $x.u\ ,\ r.u\ :=\ min(x.u, x.(r.u)) +_m 1\ ,\ nxt.(r.u)$

---

a) Show that this program satisfies the property of *Unison*:

   Starting from any *consensus state*, i.e., one where all x variables have equal values, the value of each x variable is incremented by one modulo m in every following transition of the system. (This implies that all states following a consensus state are also consensus states.)

b) Prove the stabilization of the program. Hint: show that "true leads to consensus states".