Problem 1  Leader election is an important problem that occurs often in distributed systems: for instance, we have seen how a new primary needs to be elected among the backups once the original primary fails. Solving this problem is, in general, not easy. In this homework, you will devise a protocol and prove it correct for the following simpler special case:

- The system consists of a set of \( n \) processes, communicating on a ring network.
- Every process has a unique id.
- Communication on the ring is uni-directional (either clock-wise or anticlock-wise; take your pick).
- Communication is synchronous.
- Processes do not fail (kinda nice, uh?)

A leader election protocol should guarantee that in every admissible execution of the protocol, exactly one process is elected.

1. Assume \( n \) is known to all processes Given this system model, you will:
   (a) Give the pseudo-code for a leader-election protocol that require \( O(n) \) messages
   (b) Prove that your protocol is correct (i.e. meets the guarantee above).

2. (extracredit) Assume now that no process knows \( n \).
   (a) Give the pseudo-code for a leader-election protocol that require \( O(n) \) messages
   (b) Prove that your protocol is correct (i.e. meets the guarantee above).

Problem 2  Complete the proof that consensus can be solved using the \( S \) failure detector.

Problem 3  Consider Ben-Or Lite, a modified version of Ben Or’s randomized algorithm for solving Consensus in asynchronous systems in which line 12 of the algorithm is omitted. In Ben-Or Lite, a process that does not receive \( n - f \) identical \( b \) values, all different from \( ⊥ \), always determines its \( a \)-value for the next round by flipping a coin. Does Ben-Or Lite solve Consensus in an asynchronous system?

If so, prove Validity, Agreement, and Termination. If not provide a counterexample, i.e. an execution in which at least one of the above three properties is violated.