1. Many distributed algorithms do not have a built-in termination detection mechanism. Given such an algorithm, we have to impose on it a termination detection mechanism, which will signal the nodes of the network that the algorithm has terminated.

Here is one such mechanism for non-FIFO model of computation, suggested by J.M. Helary, C. Jard, N. Plouzeau, and M. Raynal from IRISA France:

Every node \( v \) maintains 2 matrices \( \text{SENT} \) and \( \text{RECEIVED} \). Entry \( \text{SENT}(i, j) \) is the number of messages sent from \( i \) to \( j \) in the algorithm, to the best of \( v \)'s knowledge. Entry \( \text{RECEIVED}(i, j) \) is the number of messages received by \( j \) from \( i \) in the algorithm, to the best of \( v \)'s knowledge.

When ever a node, \( x \), becomes idle (from the underlying algorithm’s point of view) it floods the network with a message containing 2 vectors: \( \text{SENT}(x, \star) \) and \( \text{RECEIVED}(\star, x) \). Upon receiving a flood message every node updates its matrices accordingly.

Prove under the following assumptions, that if in some node \( \text{SENT}(i, j) = \text{RECEIVED}(i, j) \) for all pairs \((i, j)\), then there is no message of the underlying algorithm in transit on any link of the network.

Assume a non-FIFO asynchronous network. Also assume that, magically all the \( \text{SENT} \) matrices are initialized to the number of messages sent by the spontaneously starting nodes when they start (i.e., the first messages generated by the nodes which wakeup spontaneously to start the underlying algorithm, hence we assume that all start nodes start at the same time). \( \text{RECEIVED} \) matrices are initialized to 0.

2. (a) There are \( n \) identical finite automata connected by \( n \) unidirectional, asynchronous links to form a unidirectional ring, \( n > 1 \). The same automata works for any size ring. Exactly one automaton (one node) receives a WAKE signal from the environment. Design an algorithm (pseudo code of the finite automata) which informs the predecessor of the awakened automaton that it is the predecessor of the awakened automaton. Show that your algorithm has asymptotically optimal bit complexity (i.e., prove a lower bound on the total number of bits exchanged by any such solution between all the nodes).
(Hint: The lower bound should hold even for arbitrary identical processors in place of finite automata, and variable length messages, even if the processors know $n$. Can you relax these restrictions farther?)

(b) Give a tight upper and lower bound (i.e., pseudo-code and lower-bound) for this problem in the \textbf{synchronous} model.

3. The roman senate wants to \textit{randomly} elect one of its members as the head of the senate. The senators gather in a ring around the roman forum, and the announcer declares the method (distributed algorithm) by which the senate-head will be elected. Senators then converse, each with his two neighbors on the ring, according to the announced method (distributed algorithm).

However, each senator might prefer that a certain senator will be elected (itself or some other) as the head, and thus may cheat in an attempt to increase the probability of its preferred senator being elected. You have to devise an election algorithm that the announcer will declare to the senators, such that, no senator will gain anything from deviating (cheating) from your algorithm. Your algorithm should elect exactly one senate-head, and each member has an equal probability of becoming head. The cheating is constrained as follows:

(a) If the algorithm fails to elect exactly one leader, the senators are sent to prison for the rest of their life. Meaning, senators will not cheat if the cheating will definitely cause the algorithm failure. However, senators do take chances, if there is some positive probability to succeed.

(b) A senator cheats only if the cheating increases the probability of its preferred candidate being elected.

(c) Cheating senators assume they are the only cheaters.

Further make the following assumptions: (i) each senator has a unique name (known only to itself), (ii) each knows $n$, the total number of senators (iii) the senators converse in synchronous rounds (i.e., assume a synchronous ring), and (iv) assume all senators start together at the same round.

For any algorithm that you provide give its message and time complexities.

(a) Provide an algorithm when either one senator or a subset of the senators may cooperate in cheating, to increase the probability of a candidate that the sub-set prefer. Each subset assumes that it is the only one to cheat. What is the largest size sub-set under which your algorithm works correctly? It should be the maximum size for which there is a solution. Notice, members of any subset may communicate only through the ring.

(b) How can you eliminate the assumption that all start together? (should work for all the above sub-questions).

(c) Repeat the above questions for an asynchronous ring.
4. (a) It is known that there is no ring orientation algorithm for anonymous rings of any size. Here we change the model and ask again whether such an algorithm exists, and if it exists give the most efficient algorithm you can come up with. Since the ring is anonymous (processors are identical with no unique ids) we refer only to algorithms that message terminate, that is: that reach a state in which no message is in transit and all processors are in idle state. But no processor is suppose to be able to detect the fact that such a state has been reached. Further assume the following:

- The number of processors on the ring is unknown.
- Each link on the ring is equipted with a symmetry breaking marking. That is, each pair of neighbors has agreed on an ordering between them prior to the beginning of the algorithm (such an order can be marked for example by a unique "head" mark on each link).

Each node locally marks one of its incident links "left" and the other "right". The problem is to design an orientation algorithm that will assign those marks at the nodes in such a way that on every link one end is marked "left" and the other is marked "right".

(b) Assume now that the symmetry breaking markings are not available and give a message terminating randomized orientation algorithm (a Las-Vegas algorithm), with as good expected complexity as you can.

5. (a) Write the pseudo code of the snapshot detection algorithm which was discussed in class (including the collection of messages in transit on the links, but without termination detection).

(b) Modify the algorithm to work in the non-FIFO model, i.e., when the order in which messages received over a link is not necessarily the order in which they were sent. Notice, you may not delay messages of the original algorithm beyond the side effects of your algorithm. Farther notice, you may mark the messages of the original algorithm with some small constant number of bits.