Note: YOU MUST DO THE HOMEWORK BY YOURSELF. If you have difficulties in solving a question you may discuss it with friends, BUT you MUST phrase, write and formulate the answers by yourself, after you understand the solution. The language of the solution must be entirely your own. If you got some idea from a friend or a certain paper please cite her/him/it and give her/him/it credit. It will not harm you.

Notice 2: Please type (Word, or LaTeX, etc.) and submit your homework as pdf file to the email address that was given.

Notice 3: Put your name and id number on each page of the solution.

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 wokeup 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 \textit{synchronous} model.

3. Here we consider the population protocols model discussed in class. Consider a flock of birds, where each bird monitored with an electronic bracelet on its leg. The scientist would like to detect whether the number of birds in the flock is at least \( k \). Each bracelet has a finite automata cpu in it and a communication module. The modules interact with each other in pairs: * When two birds meet, the bracelets exchange their full state. * During communication, the birds have a side assigned to them: one bird is defined as Left and the other as Right. The birds know their side.

Every two birds will meet infinitely often, and meetings are uniformly distributed (Each pair of birds has the exact same probability to meet at any given time).

There are \( n \) birds, but none of the birds know \( n \).

Design an algorithm (a finite automata algorithm) for each bird so that they can detect if there are more than \( k \) birds in the wild for some constant \( k \). In other words: In case of more than \( k \) birds, eventually, by checking any birds bracelet state, we will receive a positive confirmation.

Here is an alternative phrasing that suppose to be equivalent:

Here we consider the population protocols model discussed in class. In that model there are \( n \) agents, each a finite automata. Every pair of two agents is connected by directed link. The execution of an algorithm proceeds by an infinite sequence of interactions. In each interaction the scheduler selects two agents uniformly at random and tell them to ”talk” to each other (interact), and the agents know the direction of the link between them. The agents do not know \( n \) the total number of agents. The scheduler schedules the interactions in such a way that each pair interacts infinitely often. In other words, at any point of time during the execution each pair will eventually be scheduled again.

Design an algorithm (a finite automata algorithm) for each agent so they can detect if there are more than \( k \) agents in the network for some constant \( k \).

4. (Ring Leader Election) Recall the \( \Omega(n \log n) \) lower bound that we have seen on the number of messages required for electing a leader in an asynchronous ring. The proof of the lower bound relied on two additional assumptions: (a) the process with the maximum identifier is elected, and (b) all processes need to learn the identifier of the elected leader.

(a) Prove that the lower bound holds also when these requirements are removed. Hint: use a reduction.
(b) Explain exactly where in the lower bound proof shown in class we rely on the assumption that the algorithm is uniform (that processors do not a-priori know \( n \), the ring size).

(c) Can you state more generally what properties are required by this lower bound proof, or alternatively can you suggest another problem to which that proof would apply.

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. Further notice, you may mark the messages of the original algorithm with some small constant number of bits.

6. Show that any asynchronous algorithm for computing the AND of all inputs on a ring, requires \( \Omega(n^2) \) messages in the worst case, even if the ring is oriented and \( n \) is known (but it is an anonymous ring).

Hint: Consider two rings of the same size, one holding all 1’s and the other all 1’s with a single 0. Show that until processors ”opposite” to the 0 learn about it, they have spent ”enough” messages. You should rely heavily on the asynchrony of the ring, since the argument fails for synchronous rings. Where does it fail?

How would you generalize your argument to any function with the property \( f(0, 0, ..., 0) \neq f(1, 1, ..., 1) \) ?