A few words on collaboration. I strongly believe that you will gain the most from this assignment if you try it alone (with the possible exception of the second question of Problem 1). If you decide not to do so, and elect to discuss solution ideas with colleagues, please acknowledge your collaborator(s) for each of the problems. Even if you discuss ideas with a colleague, you are responsible for writing your own, independent, solution. If you have any doubt about the policy outlined in this paragraph, please don’t hesitate to ask.

Problem 1 In class we have seen an algorithm that guarantees that a special process, the monitor, will deliver messages directed to it in causal order. In this problem, you are asked to develop protocols that allow all processes to deliver the messages they deliver in causal order.

• Consider an asynchronous system of $n$ processes; all channels are reliable, and processes never fail. Suppose that processes communicate only by broadcasting messages. Design a protocol to ensure that messages are delivered in causal order by each process, i.e. that $\text{bcast}(m_1) \rightarrow \text{bcast}(m_2)$ implies that at every process $\text{deliver}(m_1) \rightarrow \text{deliver}(m_2)$.

• (Harder) Consider now the same system, but assume that processes communicate in a point-to-point fashion. Design a protocol to ensure that messages are delivered in causal order by each process.

Problem 2 Consider an anonymous ring where processors start with binary inputs.

• Prove that there is no uniform synchronous algorithm to compute the AND of the input bits.

• Present an asynchronous (non-uniform) algorithm for computing the AND. The algorithm should send order $n^2$ messages in the worst case. Prove your algorithm correct.

• Present a non-uniform synchronous algorithm for computing the AND; the algorithm should send order $n$ messages in the worst case. Prove your algorithm correct.
Problem 3 The snapshot protocol for asynchronous systems that we have discussed in class has a single initiator. We further assumed that protocol completes before a new snapshot protocol is invoked.

Suppose now that multiple snapshots are initiated concurrently in an asynchronous system by a set of processes $I$. Write an algorithm that returns a consistent global state to each of the processes in $I$, and prove it correct. You should try to minimize the number of local checkpoints taken by your solution. The global state collected by your solution does not need to contain the state of the channels (the state of the processes suffices), and you may assume that channels are FIFO and that the network is completely connected.

Note: Two snapshots $s_1$ and $s_2$ are initiated concurrently if $s_1 \not\rightarrow s_2$ and $s_2 \not\rightarrow s_1$. 