Distributed Algorithms - HW 2

Submission in singles until 09/12/2014.
Submit to amipaz@cs, or to Ami Paz’s mailbox on the 5th floor.

1. Consider Peterson’s 1st algorithm to find a leader in a unidirectional ring.
   (a) What is the worst case (longest-chain) time complexity of the algorithm?
   (b) Give an example with as long time complexity as you can.

2. The question relates to Peterson’s 2nd algorithm to find a leader in a unidirectional ring.
   (a) Prove that the algorithm always determines exactly one processor as a leader.
   (b) Show that this leader is not always the processor with the maximal identity. Give an example with as few processors as possible where this happens.
   (c) Show exactly the point in the execution which is crucial.
   (d) Modify the algorithm so that it will determine the processor with largest identity as a leader. Explain!
       Note: It is not allowed to first isolate a single processor, and then have it send a message around the ring to identify the maximal identity.

3. Give an algorithm to find a leader in a complete, asynchronous network, with a complete sense of direction.
   In such a network, the $n$ processors are enumerate by $0, \ldots, n-1$, “clockwise”. Processor $i$ can send messages to all other processors, and it knows which of its communication lines leads to the other processors $i+1, \ldots, n-1, 0, 1, \ldots, i-1$ (its first neighbor clockwise, \ldots, its $(n-1)$-th neighbor clockwise, which is its first neighbor counterclockwise).
   However, the identifiers themselves ($0, \ldots, i, \ldots, n-1$) are NOT known to the processors. Additionally, each processor $i$ has a unique value $x_i \in \mathbb{Z}$ it can send and compare with the values of the other processors.
   Prove that your algorithm is correct, and analyze its message complexity. Give as good algorithm as you can, both in asymptotic terms and in the multiplicative constants.
   (An $O(n \log n)$ algorithm is better than an $O(n^2)$ algorithm, and then an algorithm with upper bound of $5n \log n + o(n \log n)$ messages is better than one with $6n \log n + o(n \log n)$ messages.)