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. In class (March 8) we saw a simple algorithm for Broadcast and Echo (in which one initiator sends a message to all other nodes in the network and receives a signal after all nodes are known to have received a copy of the message). In that algorithm we used one type of messages, message M for both the broadcast and the echo.

   (a) Describe a variant of the algorithm and provide its code (pseudo code) in such a way that two message types are used, type M (the original message that is being disseminated) and message type Ack (Acknowledgement) which is used to collect the acknowledgements in order to detect the termination of the broadcast process.

   (b) What is the exact complexity (messages and time) of the new algorithm.

   (c) Modify the code so at the end the root of the broadcast tree will know $n$ the total number of nodes, and $h$ the height of the created tree.

2. In class we saw an algorithm for Termination Detection of Diffusing computation. In that algorithm there were two types of messages, message type M (of the original diffusing computation) and message type ACK which was used to collect the acknowledgements in order to detect the termination of the diffusing process.

   (a) Here you are required to write the algorithm code (pseudo code). What is the exact complexity (messages and time overhead over the diffusing computation) of the new algorithm as a function of $m$ the number of messages sent by the diffusing computation.

   (b) Modify the code so at the end the root node of the termination detection will know the total number of messages sent by the diffusing algorithm.

3. Here we modify the flood algorithm presented in the first class as follows: The flood algorithm is the phase of the broadcast and echo algorithm where only message M is being forwarded down to the nodes (without the echo). We want to modify the code so at the end of the algorithm the tree that is defined by the flood (as we defined in
class) is a BFS tree. I.e., shortest path spanning tree from the initiator to the rest of
the network. To do that each message M is being tagged with the number of hops its
sender is from the initiator. That is messages are now M(i), where 0 \leq i \leq n. Each
time a node receives a message M(j) where j is smaller than the minimum value it has
received so far, that node sends again a new message M(j + 1) to all its neighbors (or
to all its neighbors except the one from which it got M(j)).

(a) Provide the code of the new algorithm.
(b) What is the message complexity of the new algorithm.

4. Given a graph G = (V,E), the radius r(v) of a vertex v is the maximum distance
\max_{v'} d(v, v') from v to any vertex in the graph. Suppose that you have an anonymous
asynchronous message-passing network with no failures whose topology is a tree.
1. Give an algorithm that allows each node in the network to compute its radius.
2. Safety: Prove using an invariant that any value computed by a node using your
algorithm is in fact equal to its radius. (You should probably have an explicit invariant
for this part.)
3. Liveness: Show that every node eventually computes its radius in your algorithm,
and that the worst-case message complexity and time complexity are both within a
constant factor of optimal for sufficiently large networks.

5. There are n agents arranged in a ring (only can talk to neighbors on the ring) enumerated 0 through n − 1. All calculations are mod n.
Initially, agent 0 has n tokens. The agents take steps asynchronously, and whenever
agent i takes a step in a configuration where it has a token but agent i + 1 does not,
agent i gives one token to agent i + 1. If either agent i + 1 already has a token, or
agent i has none, nothing happens. We assume that a fairness condition guarantees
that even though some agents are fast, and some are slow, each of them takes a step
infinitely often.

(a) Show that after some finite number of steps, every agent has exactly one token.
(b) Suppose that we define a measure of time in the usual way by assigning each step
the largest possible time consistent with the assumption that that no agent ever
waits more than one time unit to take a step. Show the best asymptotic upper
bound you can, as a function of n, on the time until every agent has one token.
(c) Show the best asymptotic lower bound you can, as a function of n, on the worst-

case time until every agent has one token.