EE 382N: Vijay Garg: TEST 2

NAME:

Here are some sample questions. Attempt all questions. I will not provide the solutions on the web. We can discuss any question in the next class.

Q. 1 **(8 points)** True or false. Justify your answer briefly (no credit without proper justification). (2 points for a correct answer, 0 for a blank, wrong answer or a correct answer with wrong justification).

• Dijkstra and Scholten's algorithm for termination detection has overhead of $O(E)$ messages where E is the total number of links in the underlying topology of the network on which the application program is run.

Consensus Problem:

• According to Fischer, Lynch and Paterson paper, consensus is impossible when messages can get lost but is possible when channels are reliable.

• The impossibility result assumes that at most $n/2 - 1$ processes fail.

- Any causally ordered computation is synchronously ordered.

Q.2 **(10 points)** (a) Define causal ordering of messages.

(b) Define synchronous ordering of messages.

Q.3 **(3 points)** Give three conditions that should be satisfied by a binary consensus protocol as given by Fischer, Lynch and Paterson, i.e. when is a protocol for binary consensus considered totally correct?

Q.4 **(3 points)** Give the time and message complexity to simulate one pulse of a synchronous algorithm by the following synchronizers:
(a) Alpha Synchronizer
(b) Beta Synchronizer
(c) Gamma Synchronizer

Please define any notation that you use.

Q.5 **(3 points)** Give an example of a situation where "everybody knows that everybody knows p" is true, but "everybody knows that everybody knows that everybody knows p" is false.

Q. 6 **(3 points)** What is meant by the self-stabilizing property of a system.

Q.7 **(12 points)** Your friend has proposed the following algorithm for providing causal ordering of messages:

P_k::

var

v:array[1..N] of integer initially $\forall i : v[i] = 0$.

Send rule from P_k to P_j:

send the vector v along with message;

increment $v[j]$.

Receive rule for message m from P_j:

enabled only if $\forall i : m.v[i] \leq v[i]$

v := max(v,m.v); (* component-wise *)

increment $v[j]$;

Does the algorithm work? Give a brief argument for correctness if it works, else give a counter-example.

Q.8 **(10 points)** Atomic Broadcast requires the following properties.

- *Validity*: If the sender is correct and broadcasts a message m, then all correct processes eventually deliver m.
- *Agreement*: If a correct process delivers a message m, then all correct processes deliver m.
- *Integrity*: For any message m, q receives m from p at most once and only if p sent m to q.
- *Order*: All correct processes receive all broadcast messages in the same order.

Show that Atomic Broadcast is impossible to solve in asynchronous systems.

Q. 9 **(4 points)** Show the following result used in the dining philosopher algorithm: Any finite directed acyclic graph has at least one source.

Q. 10 **(1 point)**
(a) Give one paper/chapter that you enjoyed most.

(b) Give one paper/chapter that you think should be dropped from the future offerings of the course. (Hint: There is no right or wrong answer for this question. Just don't leave it blank.)