SJTU — July 2013	**Problem Set #1**
Distributed Computing	**Due July 16, 2013 by 23:59:59**

Problem 1 The snapshot protocol discussed in class assumes that communication channels are FIFO. Derive a snapshot protocol for an asynchronous system that does not depend on the FIFO assumption, and prove it correct (i.e. prove that the protocol produces a consistent global state). Your protocol should not introduce unnecessary blocking. You may assume that at most one snapshot is being computed at any point during a run. Note: A solution that is tantamount to re-implementing FIFO channels is *not* an acceptable solution.

Problem 2 Consider the 2-Generals problem discussed in class. For each of the following statements specify whether it is (1) a safety property; (2) a liveness property; (3) a combination of safety and liveness or (4) no property at all.

1. If one general decides to attack, then within at most five minutes the other general also decides to attack.
2. No general ever attacks
3. Both generals eventually decide on the same value
4. Not to attack is a possible decision
5. General A sends at most 10 messages
6. General B sends at least 5 messages

Problem 3 Consider again the 2-Generals Problem. In class we saw that there can be no algorithm that ensures the required properties in a synchronous model, when messages are lost. We would like to see if there is an algorithm that ensures the required properties under different assumptions (i.e. different, stronger, models). To ensure that there is no ambiguity, we define the problem as follows:

- There are two generals, A and B.
- Each has an input value (inp_A and inp_B, respectively) taken from the set { "ready", "not ready" }
- each can execute a sequence of actions chosen among the following (here, $p \in \{A, B\}$:
  - $decide_p(v), v \in \{$"attack", "not attack"$\}$
  - $send_p(m), m \in \{$"yes", "no"$\}$
  - $deliver_p(m)$

We want a solution that satisfies the following properties:

Agreement: If both generals decide, their decision is the same.

Validity:
1. If both inputs are "not ready" to attack, then no general decides "attack".
2. If both inputs are "ready" and every message sent is delivered, then no general decides "not attack"

Termination: Every general eventually decides

1. For each of the following assumptions, give an algorithm that satisfies the requirements (*Agreement, Validity, and Termination*)—obviously, in this case you'll have to *prove* that the requirements are met—or prove that no such algorithm exists.
   (a) *Bounded loss*: At most 10 messages are lost on each channel (from general A to general B and vice-versa)
   (b) *Eventual delivery*: An *unknown* but finite number of messages are lost on each channel
2. Consider the following halt requirement:
   - *Halt*: both generals eventually halt (i.e. reach a halting state).

   where a *halting state* is defined as a state of a process (e.g. a general) from which no further activity can occur. That is, no messages are generated and the only transition is a self loop.

   Repeat part 1, where the required properties of the algorithm are *Agreement, Termination, Validity,* and *Halt.*

Problem 4 Consider a set of processes that are communicating in an asynchronous system by broadcasting messages to each other—think of the broadcast as a sequence of unicasts. Assume that the broadcast is reliable—we will spend quite a bit of time on what that means, but for now, to make things easy assume that there are **no failures**, and that, therefore, if a process broadcasts a message m, then all processes eventually deliver m.

It is often valuable to add additional ordering properties to such a broadcast primitive. For instance, consider the following definition of FIFO Order:

> If a process broadcasts a message m before broadcasting a message m', then no (correct) process delivers m' if it has not already delivered m.

Another useful ordering property is Causal Order:

> If the broadcast of a message m causally precedes the broadcast of a message m', then every (correct) process delivers m before delivering m'.

As we discussed in class, causal order implies FIFO order. Indeed, Causal Order is FIFO order, plus some "mystery property" X. So, here is the problem:

- Specify property X.
- Prove that Causal Order is equivalent to FIFO Order plus X.

Problem 5 We saw in class how to implement causal delivery in an asynchronous system when all messages are reliably sent to a single destination. Let us consider instead an asynchronous system with n nodes, where communication is through reliable broadcasts and where any node may send messages to any other node.

Provide an algorithm that guarantees that messages between nodes are delivered in causal order.