Question 1: (28 points)

Consider the quorum replication protocol shown in class.

a) Describe a scenario that demonstrates why lines 3&4 in the client side read protocol are required for correctness. In particular, which property might be violated without these lines?

Answer: The property that might be violated is "a read should not return a value that is older than the value returned by a previous read".

Specifically, consider a bi-quorum system in which not all quorums are the same (e.g., majority or a grid) and a register whose "current" value is $v$ with a timestamp $t$ held by all servers, as well as a write operation and two read operations as follows:

During the execution, suppose $w1$ obtains a timestamp $t' > t$ and updates server $s1$; $s1$ now holds $u$ with timestamp $t'$. When $r1$ executes, it receives replies from a read quorum that includes $s1$ and therefore returns $u$ and terminates. However, $r2$ receives replies from a read quorum that does not include $s1$. Hence, it returns $v$, which is an older value than $u$. Only after $r2$ terminates, the w-request messages of $w1$ reach the other servers.

b) Suppose the protocol runs with a general quorum system (rather than just a bi-quorum system, i.e., every read quorum is also a write quorum and vice versa). Propose an optimization that would enable skipping lines 3&4 sometimes without impacting the correctness of the protocol. The optimization should depend on the above assumption. No need to prove, but you should explain clearly why you think it works.

Answer: When using a general quorum system rather than a bi-quorum system, every quorum intersects every other quorum. Here, if all values returned in line 2 are the same, lines 3&4 can be skipped. In this case, it is already guaranteed that any future read operation will receive a reply from at least one server whose time stamp is at least as large as the one obtained by the current read.

Question 2: (30 points)

a) What is the purpose of Promise and Accepted messages sent with a NACK (rather than ACK) in the Paxos protocol?

Answer: In the version of Paxos we saw in class, without these operations, a leader $ll$ might be stuck waiting for a quorum forever, e.g., if most servers (acceptors) skipped to a higher round number. In particular, suppose another concurrent leader has proposed a higher round number and crashes, but its prepare message was received by a quorum that does not include $ll$ - this may cause $ll$ to be stuck forever.

Some people claimed that it is only a performance optimization since some implementations use timeouts. However, in this case, if the latency becomes longer than the timeout, this can cause a livelock. It is possible to increment the timeouts in each round, and then in practice it would terminate (very inefficiently), although theoretically the livelock is still possible. If you went this path and did not mention these issues with timeouts, it was considered an incorrect/incomplete answer.
b) Prove the termination property of Paxos under the assumptions that it is invoked with an $\Omega$ failure detector, there is at least a quorum of processes that never crash (never fail), and the network eventually delivers all messages sent between two processes that do not crash.

**Answer:**

**Claim 1:** Under the assumptions above, if a process $p$ starts executing a round $r$, then $p$ eventually finished executing $r$ (either decides, or skips to a higher round).

**Proof:** If $p$ decides, we are done. So assume $p$ does not decide in $r$.

If $p$ is a leader of $r$, the only places it waits are in lines 5 and 9. Under the assumption of a live quorum, it will always receive replies from a quorum (ACKs + NACKs).

If $p$ is not a leader, then since we showed that no leader can be stuck forever in round $r$, eventually $p$ will receive a promise/accept message with a higher round number and will skip to that round number, or will become the leader himself of such a round.

**Main proof:** Assume by contradiction that Paxos does not terminate by the above assumptions. Let $p$ be the ultimate unique leader output by $\Omega$ at all live processes. As shown in claim 1, as long as $p$ does not decide, it will continuously go through higher and higher round numbers. Hence, eventually, $p$ will pick a round number that is larger than any other round number that was ever suggested and there will be no other leader who will suggest higher round numbers. At this point, a quorum of processes will receive its prepare message and answer with promise-ACK and will then receive the corresponding accept message and answer with accepted-ACK. As a quorum would send accepted-ACK messages, $p$ will receive a quorum of such messages, decide and r-cast its decision to all live processes who will then decide. In the faster version of the protocol, all live processes will receive a quorum of accepted-ACK messages directly and will decide. A contradiction. Q.E.D.

**Question 3:** (42 points)

a) What would happen to Paxos’ correctness if it was given a ◊S failure detector and the presumed leader of a round is locally chosen by each process as some deterministic function of all unsuspected nodes (of the ◊S failure detector)? Explain.

**Answer:** The unsuspected output set of ◊S may be different at different processes. While eventually there has to be at least one live process in the intersection of all sets of unsuspected processes, these sets might differ on the identities of other live processes. Consequently, a deterministic function might continuously output different leaders to different processes forever, which could prevent Paxos from ever terminating - e.g., each leader will continuously propose higher round numbers before the other manages to bring the system to a decision.

b) Suppose we run Mostefaoui&Raynal but change the way a round’s coordinator is chosen to be the output of an $\Omega$ failure detector. Would the protocol run correctly? Explain.
Answer: No. First, according to the text of the question, the protocol still uses ◊S as failure detector and Ω is only used to choose a round leader. Hence, it is possible that in a given round no process will select itself to be the coordinator and this way all processes will wait forever for the est_c message from a coordinator.

Second, even if the above scenario does not occur, but rather two processes consider themselves coordinators (this can happen with Ω until some eventuality). This may lead to the following scenarios: s1 starts with v and all others with u. In the first round, s1 and s2 both consider themselves coordinators. f processes receive s1's value and echo it in their EST message (Line 7), while f-1 processes receive s2's value and echo it in their EST message (line 7). s1 receives these f EST messages, all carrying v and including itself, it has a majority supporting v and so it decides. However, all other processes receive the f-1 EST messages echoing u first, so after hearing from a majority, they hold \{u,v\}. In this case, they maintain their previous estimate u and go to round 2. In round 2, all process (except for s1 that already decided v) start with u, and therefore decide on u. A violation of agreement.

It was enough to mentioning any of these scenarios.

c) Is it possible to use an Ω failure detector (instead of ◊S) in Mostefaoui&Raynal without changing the algorithm? If yes, describe how? If not, explain why?

Answer: Yes, all that is needed is to suspect all nodes not considered as leader by Ω and trust the leader. Recall the M&R only relies on the failure detector for termination. Since eventually Ω outputs the same live process as leader at all processes, all processes will suspect all failed processes and there will be at least one live process not suspected by all of them (which is equivalent to ◊S). If the protocol does not terminate earlier, once this common leader becomes the round coordinator, no one will suspect it and it will terminate the protocol.

Submission instructions:

You should solve this exercise alone – submissions are individual. Solutions must be submitted through the course web site – either printed or a high-resolution scan of handwriting. Solutions must be written in Hebrew unless you get an authorization from Prof. Friedman to submit in English.

Notice, each question has a brief solution. If your answer is lengthy, it could be a sign that it is wrong.

The submission date is Wednesday 14/12/2016 before midnight.

Good luck!