Question 1:

This question relates to quorum replication (of R/W registers).

a) Show an example why removing line 4 from the client’s read protocol (but keeping all other code as is) results only in a regular register semantics rather than atomic register.

The example scenario involves a write operation W(x,v) and two read operations R1(x) and R2(x) such that W overlaps with both R1 and R2 but R2 starts after R1 ends. Suppose W updates a single server S1 and now R1 receives r-replies from a READ quorum that includes S1. Hence, R1 will return v and terminate. Since it does not wait for w-replies here, it is possible that R2 finishes its Line 2 before any additional servers are updated with v. In this case, R2 will only find previously written values and return the one with highest timestamp, but it will be a previous value to v.

b) Suppose the protocol is executed over a quorum system rather than bi-quorum system. Describe an optimization that enables skipping line 4 (sometimes) without violating the atomic register semantics.

The optimization is that if the values returned from a quorum in Line 2 are all the same, then it is possible to skip Line 4. This is because now each quorum intersects with every quorum. So we know that there is already a quorum of processes that are updated with the value v and timestamp ts we are about to return (or an even later one). Hence, any following read will find in Line 2 a value that is at least as updated as v (and ts).

Question 2:

Suppose you have access to a fault tolerant distributed compare&swap (CAS) service.

A CAS(obj,old,new) operation accepts an object obj, an expected value old and atomically stores the value new in obj if its value just prior to invoking the CAS was old.

a) Show how it can be used to implement a Consensus protocol, i.e., provide the missing code in the following:

```c
static obj global_decision = nil; // shared object

obj Consensus(my_init_val) {
    obj decision;
    decision = CAS(global_decision,nil,my_init_val);
    if (decision==nil) then
        decision = my_init_val;
    return decision;
}
```

There are multiple variants of this solution and all correct ones were accepted.

b) What can you conclude about the ability to implement a CAS in a fault tolerant manner in an asynchronous distributed environment? Explain.
We can conclude that CAS cannot be implemented in an asynchronous distributed environment prone to failures since otherwise Consensus would have been due to our reduction above and that would contradict the FLP impossibility result.

**Question 3:**

In this question we will prove the agreement property of Paxos. We will use the version of the protocol in which acceptors send ACCEPTED message only to the leader and then the leader sends the COMMIT message.

**Lemma:** Let \( p \) be a leader of a round \( r \) that gather (ACCEPTED,\( r,ACK \)) messages from a quorum. Then no leader with a round number \( r_1 > r \) would send an (ACCEPT,\( r_1,v_1 \)) message with \( v_1 \neq v \).

**Proof:** Assume by way of contradiction that the lemma does not hold, and let \( q \) be the leader with minimal \( r_1 \) that sends an (ACCEPT,\( r_1,v_1 \)) message with \( v_1 \neq v \). Hence, \( q \) receives at least one (PROMISE,\( r_1,ACK,r',v' \)) message such that \( r' \) was maximal among received PROMISE messages and \( v' = v_1 \).

Due to the quorum intersection property, at least one of the PROMISE messages received by \( q \) was from a process that sent an (ACCEPTED,\( r,ACK \)) message to \( p \), adopted \( v \) as its value in round \( r \), and set last_good_round to \( r \).

Due to the minimality of \( r_1 \) and the code, all \( r' \) values in the (PROMISE,\( r_1,ACK,r',v' \)) messages received by \( q \) for which \( v_1 \neq v \) must be smaller than \( r \). A contradiction to the assumption that \( q \) chose \( v_1 \neq v \) for its (ACCEPT,\( r_1,v_1 \)) message.

**Theorem:** Paxos satisfies the agreement property of consensus.

**Proof:** Assume, by way of contradiction, that the theorem does not hold. That is, since a non-leader node decides by receiving a COMMIT message from a leader, the above means that there are two leaders \( L_1 \) and \( L_2 \) that send (COMMIT,\( r_1,v_1 \)) and (COMMIT,\( r_2,v_2 \)) messages such that \( v_1 \neq v_2 \). We also assume uniqueness of round numbers, e.g., by assuming they include the id of the leader that proposed the round number to break symmetry. Without loss of generality, suppose \( r_1 < r_2 \) and these are the minimal rounds for which this holds.

From the code, we conclude that \( L_1 \) must have gathered (ACCEPTED,\( r_1,ACK \)) messages for its (ACCEPT,\( r_1,v_1 \)) message from a quorum. Similarly, \( L_2 \) must have sent an (ACCEPT,\( r_2,v_2 \)) message, but this contradicts the lemma given the assumptions that \( r_1 < r_2 \) and \( v_1 \neq v_2 \).

Correct variants of this proof were accepted as well.

**Question 4:**

a) Show a simple transformation from \( \Omega \) to \( \diamond S \). Explain your construction. An unnecessarily complex construction will receive a lower grade than a simple one.

The simplest solution is to suspect all processes except for the output of \( \Omega \) (and maybe myself). This ensures eventually strong completeness since \( \Omega \) ensures that eventually
outputs a live process, and hence all faulty nodes will eventually be suspected. The solution ensures weak accuracy since $\Omega$ ensures that eventually its output at all processes will be the same live process meaning that in our solution this eventual identical live process outputted by $\Omega$ will not be suspected by any live process.

b) Explain the fundamental differences between the safety properties of Non-Blocking Atomic Commit and Consensus.

In Consensus, the decided value has to be one of the initial values of any of the processes. In contrast: (a) In NBAC, once at least one process votes ‘no’, only abort can be decided and (b) even if all processes vote ‘yes’ but there are failures, it is suddenly permissible to decide abort (even though no one voted ‘no’).

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.

Try to be brief. If your answer is very lengthy, it could be a sign that it is wrong.

The submission date is Thursday 30/11/2017 (extended to 4/12/2017) before midnight.

Good luck!