You are to work on the following questions *alone*. Do not discuss these questions with anyone. Typeset your answers and submit as a `PDF`.

1. (10 points) **Paxos Acceptor States**

   Consider a deployment of single-instance Paxos with three acceptors. Decide whether each of the following is a valid state of the three acceptors. If the state is not valid, explain why in one sentence. (Hint: A state is valid if there is some sequence of message deliveries and message drops and node failures that leads to the state, assuming a correct implementation of proposers and acceptors.)

   For each part, we give you the highest accepted proposal at all three acceptors (A_1, A_2, A_3) at a single instance in time. Each acceptors highest accepted proposal is either in the form (n, v) where n is the proposal number (à la Paxos Made Simple) and v is a value or $\perp$ which indicates that the acceptor has not accepted any proposal.

   (a) A_1: $\perp$, A_2: $\perp$, A_3: $\perp$

   (b) A_1: $(1,x)$, A_2: $(2,y)$, A_3: $\perp$

   (c) A_1: $(2,x)$, A_2: $(2,y)$, A_3: $\perp$

   (d) A_1: $(1,x)$, A_2: $(2,y)$, A_3: $(3,z)$

   (e) A_1: $(1,x)$, A_2: $(2,y)$, A_3: $(3,x)$

2. (10 points) **Acceptor States in a Larger System**

   Consider a deployment with five acceptors. Is the following state valid? If it is valid, describe an execution that results in this state. If it is not valid, explain why.

   A_1: $(20,x)$, A_2: $\perp$, A_3: $(22,y)$, A_4: $(20,x)$, A_5: $(18,x)$

3. (10 points) **A Dubious Execution**

   Consider another Paxos deployment with acceptors A_1, A_2, and A_3, proposers P_1, P_2, and a distinguished learner L. According to the Paxos paper, a value is chosen when a majority of acceptors accept a proposal with that value, and only a single value is chosen. How does Paxos ensure that the following sequence of events cannot happen? What actually happens, and which value is ultimately chosen?

   - P_1 prepares proposal number 1, and gets responses from A_1, A_2, and A_3.
   - P_1 sends $(1,x)$ to A_1 and A_3 and gets responses from both. However, P_1's proposal to A_2 was dropped. Because a majority accepted, P_1 informs L that x has been chosen. P_1 then crashes.
   - P_2 prepares proposal number 2, and gets responses from A_2 and A_3.
   - P_2 sends $(2,y)$ messages to A_2 and A_3 gets responses from both, so P_2 informs L that y has been chosen.

4. (10 points) **Paxos Liveness**

   In the absence of a distinguished proposer, it is possible for Paxos to fail to make progress even if no messages are dropped and no nodes fail. Briefly describe how this can happen in a system with two proposers and three acceptors. Be specific about which messages are sent and in what order they are delivered.

5. (10 points) **Alternate Paxos Implementation**

   The *Paxos Made Simple* paper has the following definition in page 3.

   > A value is chosen when a single proposal with that value has been accepted by a majority of the acceptors.

   Consider pursuing an alternate implementation based on the following definition.

   > A value is chosen when proposals with that value have been accepted by a majority of the acceptors.

   Would the resulting implementation be correct? Justify your answer in a few sentences either with an informal proof or a scenario where this implementation would violate safety.

6. **Holes in the Paxos Log**

   In both PMMC or the multi-slot version of Paxos, when a new leader takes over (after the old leader fails, or because enough messages from the old leader are lost), the new leader receives information from each acceptor – for each slot, the value of the highest ballot accepted by that acceptor for that slot.

   Suppose the new leader hears back from only a bare majority of acceptors (e.g., three of five). This constraint applies for all three parts.

   (a) (5 points) Explain in a sentence (or three) how the new leader can use this to classify slots into ones that definitely have an operation chosen, ones that definitely have not had an operation chosen, and those where the result is indeterminate, without sending any further messages.

   (b) (5 points) A hole in the log occurs when for slot i no value was accepted by any acceptor (whose reply reaches the new leader), while for slot j, $j > i$, the leader learns that some operation was chosen. Give an example to illustrate how this might occur.

   (c) (10 points) The Paxos paper suggests that the new leader should attempt to fill any log holes with no-op operations before accepting any new client requests. And that the leader should not return an operation as completed, even if it is chosen, until all slots prior to that slot have been chosen.

   An overly ambitious student suggests that we don't need no-ops. If the log has a hole, simply wait until a new client request arrives and put it into that slot. If a put request is chosen for some slot after a hole, we can return that operation to the client immediately since the result won't depend on what operation is put into the missing slot.

   Explain how this optimization might compromise both liveness and linearizability.