Name: Ujjwal Mehta Entry no: 2020CS10401

In Raft, each server remembers the following states. From the paper, we mark which state is persistent, i.e, written to disk before responding to RPCs, and which ones are volatile i.e, stored in memory and may be lost upon a reboot. Please mention *what exactly* goes wrong if a persistent state is made volatile, and *how exactly* is a lost volatile state recovered after reboot. Please be concise and precise.

[2 marks] Persistent **currentTerm:** latest term server has seen (initialized to 0 on first boot, increases monotonically)

Ans: If currentTerm is made volatile then it may happen that after a crash of a server, it may not get to become a leader ever again or may take a lot of time to become leader if it crashed from a high current Term.

[2 marks] Persistent **votedFor:** candidateId that received vote in current term (or null if none)

Ans: If voted for is made volatile, then we won't be having any problems with correctness cause even if that information is lost due to volatility, we can still re-election in next term to choose leader.

[2 marks] Persistent **log entries**: each entry contains command for state machine, and term when entry was received by leader (first index is 1)

Ans: If log entries are made volatile then it might happen that each server follows different part of logs (due to restarting of servers) eventually making no majority for a read request which will delay everything to work.

[2 marks] Volatile **commitIndex**: index of highest log entry known to be committed (initialized to 0, increases monotonically)

Ans: Having commit Index as volatile is fine since we have persistent log entries, we can just ~~[illegible]~~ recommit the log entries to our state even if commit Index is resetted and this will still ensure correctness. (recommit by communication to get highest commit Index)

[2 marks] Volatile **lastApplied:** index of highest log entry applied to state machine (initialized to 0, increases monotonically)

Ans: lastApplied can be volatile too, since we are having persistent log entries, we can just reapply the logs to the state and gradually increase lastApplied again once we have lost lastApplied. Correctness is there due to deterministic outcome on our state.