this was the key element for linearizability.

↓

operations should only once as if they are done on a single system linearly.

It ensures that the behaviour of the system is consistent with what would be expected if operations were executed one at a time, in a sequential manner.

Q

	10	11
S_1 :	3	
S_2 :	3	3
S_3 :	3	3

is this situation possible?
Yes, assuming S3 be the leader of 3rd Term, then it is sending RPCs both the index, but S1 didn't recieve it as it might be temporary down.

Q

	10	11	12	13
S_1 :	3			
S_2 :	3	3	4	
S_3 :	3	3	5	

is this situation possible?
Yes, assuming S3 be the leader during Term 3 at index 10, it sends RPCs, everyone accepts it, in index 11, S3 again sends it but just after sending it to (S2) it dies/breakdown leaving S1 index 11 empty, so new election takes places & S2 becomes leader their as it has more entries than (S1). Hence it has term 4 at index 12 now again before sending RPC to S1, S2 goes down & (S3) comes online & new election happens & S3 become leader of Term 5.

Q

	10	11	12
S_1	5	6	7
S_2	5	8	
S_3	5	8	

Is this possible?

Yes in a scenario, initially S_2 is leader at term 5 (index 10) so it sends RPCs for it which every server acknowledge not at ~~term~~ index 11, (S_2) goes offline & election happens, (S_1) becomes the leader for term 6. then S_1 dies again & respawns immediately & hence new election, & again (S_1) becomes leader for term 7, now (S_1) again goes offline and S_2 comes back. so election happens with S_2 && S_3 and S_3 becomes leader and ~~vote~~. Send RPC with term 8.

=> A server needs only 3 items to be persistent. (very costly action to maintain these)

-> Log : only record of application state

-> currentTerm : So that in case the server with latest term dies, they still get correct Term after new election

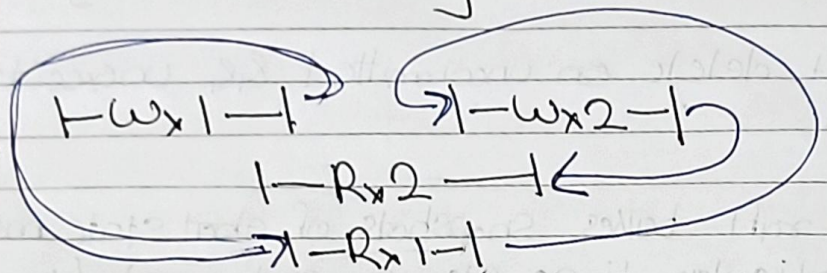
-> votedFor : ~~to~~ flag to know that the server has already voted or not.

ref for this: here if S_1 dies & S_2, S_3 does an election with last term 5 then it might lead to problem Hence they should know current Term.

Linearizability: an we use this term to judge a system execution history to see if the system in consistent.

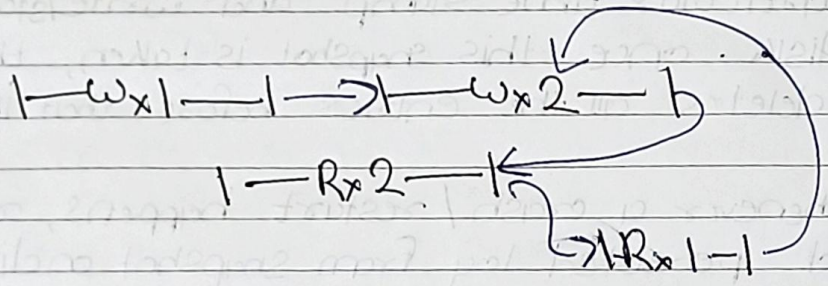
an execution history is linearizable if one can find a total order of all operations, that matches real time (for non overlapping op and in each read, sees the value from the write preceeding it in the order.

Q1



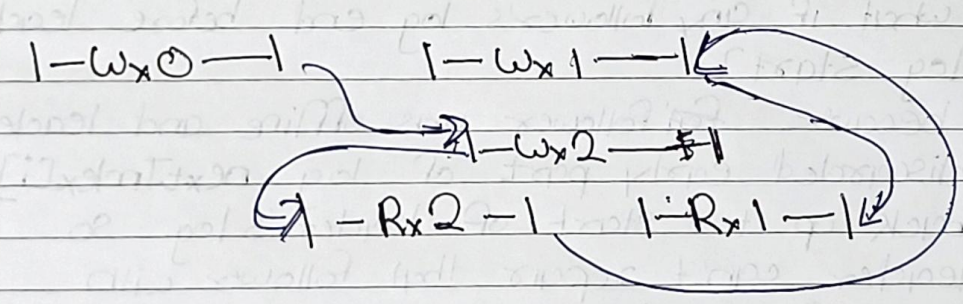
Yes

Q2



Cycle
∴ NO

Q3



Yes

$W_{x0} \rightarrow W_x2 \rightarrow R_x2 \rightarrow W_x1 \rightarrow R_x1$

Q. |-- Wx0 --| |-- Wx1 --|

|-- Wx2 --|

client1 = C1:

client2 = C2:

~~|-Rx2-|~~ ~~|-Rx1-|~~

|-Rx1-| |-Rx2-|

No, it is not, there are only 2 write happening, so either Rx2 should come before Rx1 or vice versa not both. hence only one order should be possible not 2. X.

Q. |--Wx1--|

|--Wx2--|

not possible |--Rx1--|

Q. C1: |--Wx3--| |--Wx4|

C2: |-Rx3----|

(Yes) Server was unable to reply in time,