Problem 1. Consider the Disk Paxos algorithm in slides 14-15 of the lecture. The algorithm is reproduced below. If we omit line 11, is the algorithm still correct? Why or why not?

Algorithm 1 Obstruction-free consensus with Memory Failures

1: procedure PROPOSE(v)
2: while true do
3: for every memory m in parallel do
4: \text{Reg}[m][i].T.write(ts)
5: \text{temp}[m][1..n] \leftarrow \text{Reg}[m][1..n].read()
6: until completed for majority of memories
7: val \leftarrow \text{temp}[1..m][1..n].highestTspValue()
8: if val = \bot then val \leftarrow v
9: for every memory m in parallel do
10: \text{Reg}[m][i].V.write(val, ts)
11: \text{temp}[m][1..n] \leftarrow \text{Reg}[m][1..n].read()
12: until completed for majority of memories
13: if ts = \text{temp}[1..m][1..n].highestTsp() then return (val)
14: ts \leftarrow ts + n

Problem 2. Consider the following variant of the Non-equivocating Broadcast algorithm seen today in class. Does this algorithm satisfy the Non-Equivocating Broadcast properties? Why or why not?

Algorithm 2 Non-equivocating Broadcast

1: procedure BROADCAST(m)
2: \text{R}[s].write(m)
3: procedure RECEIVE
4: senderMsg = \text{R}[s].read()
5: for i = 1…n do
6: recvMsg = \text{R}[i].read()
7: if recvMsg \neq \bot \land recvMsg \neq senderMsg then
8: \top found conflicting values (Byzantine sender), don’t deliver
9: return
10: \text{R}[i].write(senderMsg)
11: deliver(senderMsg)