1a) 1b)

1c)
We accepted any of the following two answers:
1. Average size of a finger table is 
   \[ \log_2 N = \log_2 5 \approx 3 \]
2. Average size of a finger table (in this example) is
   \[ (2+2+2+3+3) / 5 = 13 / 5 = 2.6 \]

1d)
Node 5 stores the file with ID 0. The steps (16 – 50 – 60 – 5) are shown in the figure.

1e)
The file with ID 0 is moved from node 5 to node 4 since files with IDs 61~4 now belong to node 4 instead of node 5.

Some of you mentioned about successor and predecessor. You got 1 point if you mentioned successor of 60 is 4 and successor of 4 is 5, and predecessor of 5 is 4 and predecessor of 4 is 60.

2)
3) The conditions were taught in class. Each condition has two points, and you got 1 point for each.
   - Safety: \( \forall \) non-faulty process \( p \): (\( p \)'s elected = (q: a particular non-faulty process with the best attribute value) or \( \perp \))
   - Liveness: \( \forall \) election: (election terminates) & \( \forall p \): non faulty process, \( p \)'s elected is not \( \perp \)

4) Assume external clock is at \( S(t) \), and two physical clocks read \( C_1(t) \) and \( C_2(t) \). Since \( C_1(t) \) and \( C_2(t) \) are externally synchronized within bound \( D>0 \), in the worst case one of them can be at \(+D\) and one of them can be at \(-D\).

Assume
\[
C_1(t) = S(t) + D \ldots \ldots (1)
\]
\[
C_2(t) = S(t) - D \ldots \ldots (2)
\]

Hence, (1) - (2)
\[
C_1(t) - C_2(t) = 2D
\]

Similarly, it can be shown that \( C_1(t) - C_2(t) = -2D \)

Therefore \( C_1(t) \) and \( C_2(t) \) can be maximum 2\( D \) apart from each other.

So, \( |C_1(t) - C_2(t)| < 2D \)

We also accepted your answer if you just explained the above in words.

5)

![Diagram](attachment:image.png)

The actual time can be anytime between \((T+\text{min2})\) or \((T + (\text{RTT-min1}))\), where \( T \) is
6:00PM, RTT is 7 sec, min1=2sec (the minimum one way transmission time from client to server), and min2=1sec (the minimum one way transmission time from server to client).

The two extremes are:

(6:00 + 1 sec) = 6:00:01 PM and

(6:00 + (7-2) sec) = 6:00:05 PM

The client sets its time to the mid point of the two extremes, i.e.,

(6:00:05 + 6:00:01)/2

= 6:00:03 PM

6a) False. The following figure is a counter example.

6b) True. The reason is that every process $i$ increments the $i$th element of its vector upon send or receive event. Also, in receiving a message from process $j \neq i$, $V_{receiver}[j]$ is set to $\text{Max}(V_{receiver}[j], V_{message}[j])$. In this way, events are always assigned different vector timestamps.

6c) False. The following figure shows a counter example, in which causal ordering is satisfied, while total ordering is violated. In this figure, A->B and every process delivers A first and then B. So, it satisfies causal ordering. But it violates total ordering, since P1 delivers B first and then C, while p2 delivers C before B.
7) (a) 1-6, 4-5, and 7-8.
   (b) No. You can reorder 4 and 5 and that would work.
   (c) Transaction X would block acquiring a write lock on C at step 5
   (d) Yes, deadlock would occur. At step 6, transaction Y would wait
   for a write lock on A, while X is waiting for a lock on C.

8) Preconditions are:
   - Exclusive access to a resource
   - Circular wait
   - No preemption

Edge-chasing violates no preemption because it aborts a transaction once a cycle
has been found.