Run the following Python script:

```python
def bloom_false_positive(hash_func):
    def hashes(x, num_hashes=3):
        matches = 0
        print(x, end=':"
        for i in range(1, num_hashes+1):
            h = hash_func(x, i)
            print(h, end='",")
            if not bloom[h]:
                bloom[h] = 1
            else:
                matches += 1
        print()
        return matches == num_hashes

    bloom = [0] * num_bits
    x = step = 3
    while True:
        if hashes(x):
            break
        x += step
    return x  # first false-positive value

m = 64
h1 = lambda x, i: (i**(2*x)+x-i)%m
h2 = lambda x, i: (x**(2*i)+x-i)%m
print("h1: first false positive integer: %d" % bloom_false_positive(h1))
print("h2: first false positive integer: %d" % bloom_false_positive(h2))
```

For h1 = (i^2x+x-i)%64, answer is 69. The hashed bits are (5, 3, 43) which have all been taken by previous integers (15, 3, 45).

For h2 = (x^2i+x-i)%64, answer is 54. The hashed bits are (25, 4, 51) which have all been taken by previous integers (9, 21, 27 and 36).
2. (Solution and Grading by: <Yigong>.)
   a. Advantage: less memory usage (no need to store bloom filters in memory); writes to
disk faster (no need to update the bloom filter on write to SSTables).
   Disadvantage: searches slower / reads slower (no quick way to check whether the
search key exists in a SSTables for read).

   b. Advantage: reads faster (no need to merge data on disk (in SSTables) with data in
RAM (in memtables)); no cache management needed; less memory usage.
   Disadvantage: writes slower (lots of disk accesses).

   c. Advantage: reads faster (no need to touch multiple SSTables).
   Disadvantage: extra bandwidth on flush (overhead to fetch and write multiple SSTables
from/to disk); slower writes (need to modify SSTables every time).

3. (Solution and Grading by: <Yigong>.)
   a.
   Linearizability: each operation by a client is visible (or available) instantaneously to all
other clients.
Sequential consistency: the result of any execution is the same as if the operations of all
the processors were executed in some sequential order, and the operations of each
individual processor appear in this sequence in the order specified by its program. Can
be seen as linearizability without the real-time constraint.
Causal Consistency: reads must respect partial order based on information flow. Only
write operations that are causally related need to be seen in the same order by all
processes.

   b.

   \[
   \begin{align*}
   &A \xrightarrow{W(K, a)} R_1(K) \quad R_2(K) \\
   &B \xrightarrow{W(K, b)} R_1(K) \xrightarrow{R_2(K)} \\
   &C \xrightarrow{R_1(K)} R_2(K) \xrightarrow{R_3(K)}
   \end{align*}
   \]

   Linearizability: A: \( R_1(K) = R_2(K) = b \) \; C: \( R_1(K) = R_2(K) = b \)
Sequential consistency: A: \( R_1(K) = a, \; R_2(K) = b \) \; C: \( R_1(K) = a, \; R_2(K) = b \)
Causal Consistency: A: \( R_1(K) = a, \; R_2(K) = b \) \; C: \( R_1(K) = b, \; R_2(K) = a \)
4. (Solution and Grading by: <Bhavana>. )
We know that the error using Cristian’s algorithm = \((\text{RTT} - \text{min1} - \text{min2}) / 2\)

Plugging values using the data in the question -
(Note - if we do not know the value for something, assume zero. The more data you have about min1 and min2, the lesser your error bounds.)

\[ \text{RTT} = 1230\mu s \]
\[ \text{min1} = (\text{app to NIC) at client + packet transfer from client to server + (NIC to app) at server} \]
\[ = 23.4 + 300 + 0 \mu s \]
\[ = 323.4 \mu s \]

\[ \text{min2} = (\text{app to NIC) at server + packet transfer from server to client + (NIC to app) at client} \]
\[ = 0 + 300 + 50 \mu s \]
\[ = 350 \mu s \]

\[ \text{Error} = (1230 - 323.4 - 350) / 2 \]
\[ = 278.3 \mu s \]

5. (Solution and Grading by: <Ruiyang>. )
Error bound should be 0.
Since \(|\text{Oreal-O}| < |\text{L1-L2}|/2\) and here we have identical L1,L2, thus \(|\text{Oreal-o}|\) is bounded by 0.

6. (Solution and Grading by: <Ishani>. )

**Lamport Timestamps**

![Lamport Timestamps Diagram]
7. (Solution and Grading by: <Ishani>.)

Vector Timestamps

8. (Solution and Grading by: <Binyao>.)

Lamport timestamp figure:

Yes, there will be causality violations as shown in above figures. Besides, any other causality violation is okay.

9. (Solution and Grading by: <Zhanghao>.)

\[2 \text{(a)} + 2 \text{(b)} + 2 \text{(c)} + 2 \text{(d)} + 2 \text{(reasoning)} = 10, -0.5 \text{ for incorrect rounding}\]

For any two quorums of size \(Q\), their intersection size is at least \(\max(0, Q + Q - N) = 2Q - N\). The minimum intersection size between this intersection and the third quorum set is \(\max(0, 2Q - N + Q - N) = \max(0, 3Q - 2N)\). Therefore, since
this intersection size between these three quorums needs to be at least K, this implies
$Q_{\min} = \lceil (2N + K)/3 \rceil$.

a) $Q_{\min} = \lceil (100 + 2)/3 \rceil = 34$.
b) $Q_{\min} = \lceil (2N + N/2)/3 \rceil = \lceil 5N/6 \rceil$.
c) $Q_{\min} = \lceil (2N + 2)/3 \rceil$.
d) $Q_{\min} = \lceil (2N + K)/3 \rceil$.

10. (Solution and Grading by: <Zhanghao>.)

[4 (correct answer) + 4 (use induction) + 2 (correct induction details) = 10, +4 for intuitive reasoning but no induction, -0.5 for incorrect rounding]

Claim: The minimum intersection size between any M quorums of size Q is
$max(0, M * Q - (M - 1) * N)$.

Proof: By induction.

- **Base Case ($M = 1$):** the intersection size is 0 and the claim obviously holds.
- **Inductive Step ($M > 1$):** Assume the claim holds when $M = L$ ($L >= 1$). When $M = L + 1$, the minimum intersection size between any L quorums of size Q is $max(0, L * Q - (L - 1) * N)$ by inductive hypothesis. We know that the minimum intersection size between any 2 subsets of size S1 and S2 in a universal set of size N is $max(0, S1 + S2 - N)$. Take $S1 = max(0, L * Q - (L - 1) * N)$, $S2 = Q$, it follows that the minimum intersection size between any $(L + 1)$ quorums of size Q is $max(0, Q + (L * Q - (L - 1) * N) - N) = max(0, (L + 1) * Q - (L + 1 - 1) * N)$.

Thereby, the claim holds for $M = L + 1$.

Therefore, the claim holds by induction.

The intersection size between any M quorums is at least K implies
$M * Q - (M - 1) * N >= K$. Solving for $Q_{\min}$ and we get
$Q_{\min} = \lceil ((M - 1) * N + K) / M \rceil$. 