The aim is to design a distributed ledger using simple distributed systems' primitives. Unlike BitCoin, we will not use cryptographic
primitives, or other hashing mechanisms. However, the system that we design can be used to implement BitCoin, and all other systems
that rely on distributed ledgers. Write all your code in Google Go.

1) Let us assume N nodes, where each node knows the identify of the rest of the nodes. (N >= 75)

2) A node can have two states at any point in time: <in the network, online> , or <not in the network, offline>
This means that at any point of time, M nodes can actually be considered to be alive, where M <= N

3) You can assume that with a random probability, a node suddenly chooses to become offline. Furthermore, it
becomes offline without informing anyone (fail-stop failure mode)

4) Let us define a transaction, as an agreement between three nodes: A, B, and C. All three need to say commit
for the transaction, for it to be actually committed. Assume that with some random probability, a node decides
to say ``abort''. Each transaction is uniquely identified by its scalar Lamport clock: node id, local transaction
number (monotonically increasing sequence).

5) Task 1: Simulate the 2-phase commit protocol among the three nodes.

6) Task 2: If a transaction is successful, then broadcast it to all the nodes in the network that are online. We
need to preserve virtual synchrony here. See the algorithms for ensuring virtual synchrony in the slide set.

7) Task 3: It is possible that multiple transactions are broadcasted concurrently. All the nodes should perceive
the same order of broadcasts (Atomic globally ordered broadcast). All the nodes add the transactions
to a linked list.

8) At the end the linked lists of all the online nodes should have the same contents. Let us verify this by printing
a hash of the contents of the linked list. All the hashes should be the same.

9) Bonus marks: replace broadcast in task 3, with multicast to a quorum