1. The major flaw in the protocol we discussed for installing a next view $G_{i+1}$ in a virtually synchronous reliable multicast is that it cannot deal with process failures while a new view change is being announced. In particular, the protocol assumes that until the new view $G_{i+1}$ has been installed by each member in $G_{i+1}$, no process in $G_{i+1}$ will fail. Adapt the protocol so that it can tolerate process failures during installing a new view.

2. In the original two-phase commit protocol (2PC) it might be possible that all participants block until the coordinator recovers. In this case, participants cannot cooperatively decide on the final action to take. For this reason, 2PC is also referred to as a blocking commit protocol.
   - Describe when such a scenario might occur.
   - Describe whether it is possible to completely eliminate blocking in 2PC, assuming that participants can elect a new coordinator.
   - Can you suggest an adaptation to 2PC to allow participants to reach a final decision, even if the coordinator has not yet recovered?

3. Use Amdahl’s law to resolve the following questions:
   - Suppose a computer program has a method $M$ that cannot be parallelized, and that this method accounts for 10% of the original program’s computation. What is the maximum speed-up of the program on 10 processors vs. on 1000 processors?
   - Suppose that you want to achieve a speedup of 20 with 32 processors. What fraction of the original computation can be sequential?
   - Suppose the method $M$ accounts for 0.5% of the original program’s computation.
     - What will be the maximum speed-up ratio on an unlimited number of processors?
     - What observation can you make regarding obtaining a high degree of scalability?
4. Read the paper entitled “MapReduce: Simplified Data Processing on Large Clusters” by Jeffrey Dean and Sanjay Ghemawat and answer the following questions:
   o Summarize the paper in 2 paragraphs.
   o What kind of constraints does MapReduce place on its problem domain? Said in another way, what applications you think would not work (well) in MapReduce?
   o Given the following characteristics of a reliable distributed system: (1) fault-tolerant, (2) highly available, (3) recoverable, (4) consistent, (5) scalable, and (6) predicable performance. For each of these, write a few sentences describing how MapReduce has been designed to exhibit that specific characteristic.