17.6 . A serializable schedule corresponding to the precedence graph exist because the graph is acyclic. A possible schedule is T_1, T_2, T_3, T_4, T_5.

17.7 A cascadeless schedule is one where, for each pair of transaction T_i and T_j that T_j read data items previously written by T_i, the commit operation of T_i appears before the read operation of T_j.

cascadeless schedules are desirable because the failure of a transction does not lead to the aborting of any other transaction

18.1 If two-phase locking does not ensure serializability, then there will be a set of transection $T_0, T_1, \dots, T_{n-1}$ which obey 2PL and which produce a nonserializabe schedule. A non-serializable schedule implies a cycle in the precedence graph, and we shall show the 2PL cannot produce such cycles. without loss of generolity, assume: $T_0 \to T_1 \to \dots \to T_{n-1} \to T_0$. Let α_i be the time at which T_i obtains its last lock. Then for all transactions such that $T_i \to T_j$, $\alpha_i < \alpha_j$. Then we have $\alpha_0 < \alpha_1 < \cdots < \alpha_{n-1} < \alpha_0$. Because $\alpha_0 < \alpha_0$ is impossible, hence 2PL cannot produce non-serializable schedules.

18.7 a. Serializability can be shown by observing that if two transactions have an I mode lock on the same item. However, any pair of conflicting operations must be serialized in the order of the lock points of the corresponding transactions.

b. The increment lock mode being compatible with itself allows multiple incrementing transactions to take the lock simultaneously thereby improving the concurrency of the protocol. With the absence of this mode, an exclusive mode will have to be taken on a date item by each transaction that wants to increment the value of this data item

18.18. 1. easy to implement
2. low rollback
3. lowe level of concurrency.

18.32 The phantom phenomenon arises when, due to an insertion or deletion, two transaction logically confict despite not locking any data items in common. If T_i delete a tuple from a relation while T_j scans the relation. An interpretation of 2PL as just locking the accessed tuples in a relation is incorrect. There is also an index or a relation data that has information about the tuple in the relation.