conflict serializable schedules. The general form of view serializability is very expensive to test, and only a very restricted form of it is used for concurrency control.

- **Answer:** There is a serializable schedule corresponding to the precedence graph below, since the graph is acyclic. A possible schedule is obtained by doing a topological sort, that is,  $T_1$ ,  $T_2$ ,  $T_3$ ,  $T_4$ ,  $T_5$ .
- Answer: A cascadeless schedule is one where, for each pair of transactions  $T_i$  and  $T_j$  such that  $T_j$  reads 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 transaction does not lead to the aborting of any other transaction. Of course this comes at the cost of less concurrency. If failures occur rarely, so that we can pay the price of cascading aborts for the increased concurrency, noncascadeless schedules might be desirable.

## 14.8 Answer:

a. A schedule showing the Lost Update Anomaly:

$T_1$	$T_2$
read(A)	
()	read(A)
	write(A)
	WIIIe(A)
$\mathbf{write}(A)$	

In the above schedule, the value written by the transaction  $T_2$  is lost because of the write of the transaction  $T_1$ .

b. Lost Update Anomaly in Read Committed Isolation Level

$T_1$	$T_2$
lock-S(A)	
read(A)	
$\mathbf{unlock}(A)$	
	lock-X(A)
	read(A)
	$\mathbf{write}(A)$
	$\mathbf{unlock}(A)$
	commit
lock-X(A)	
$\mathbf{write}(\hat{A})$	
$\mathbf{unlock}(A)$	
commit	

The locking in the above schedule ensures the Read Committed isolation level. The value written by transaction  $T_2$  is lost due to  $T_1$  write.

c. Lost Update Anomaly is not possible in Repeatable Read isolation level. In repeatable read isolation level, a transaction  $T_1$  reading a