15.1 **Answer:** Suppose two-phase locking does not ensure serializability. Then there exists a set of transactions $T_0, T_1 \ldots T_{n-1}$ which obey 2PL and which produce a nonserializable schedule. A non-serializable schedule implies a cycle in the precedence graph, and we shall show that 2PL cannot produce such cycles. Without loss of generality, assume the following cycle exists in the precedence graph: $T_0 \rightarrow T_1 \rightarrow T_2 \rightarrow \ldots \rightarrow T_{n-1} \rightarrow T_0$. Let α_i be the time at which T_i obtains its last lock (i.e. T_i's lock point). Then for all transactions such that $T_i \rightarrow T_j$, $\alpha_i < \alpha_j$. Then for the cycle we have

$$\alpha_0 < \alpha_1 < \alpha_2 < \ldots < \alpha_{n-1} < \alpha_0$$

Since $\alpha_0 < \alpha_0$ is a contradiction, no such cycle can exist. Hence 2PL cannot produce non-serializable schedules. Because of the property that for all transactions such that $T_i \rightarrow T_j$, $\alpha_i < \alpha_j$, the lock point ordering of the transactions is also a topological sort ordering of the precedence graph. Thus transactions can be serialized according to their lock points.

15.2 **Answer:**

a. Lock and unlock instructions:

T_{34}:

lock-S(A)
read(A)
lock-X(B)
read(B)
if $A = 0$
then $B := B + 1$
write(B)
unlock(A)
unlock(B)

T_{35}:

lock-S(B)
read(B)
lock-X(A)
read(A)
if $B = 0$
then $A := A + 1$
write(A)
unlock(B)
unlock(A)

b. Execution of these transactions can result in deadlock. For example, consider the following partial schedule:

T_{31}	T_{32}
lock-S (A)	
	lock-S (B)
	read(B)
read(A)	
lock-X (B)	
	lock-X (A)

The transactions are now deadlocked.