1. For each of the following schedules, state which of the following properties hold: *conflict serializable*, *recoverable*, *cascadeless*, or *strict*. Recall the following definitions:

   - A schedule is *conflict serializable* if its precedence graph is acyclice.
   - A schedule is *recoverable* if T_j reads an item written by T_i then T_i commits before T_j commits.
   - A schedule is *cascadeless* if T_j reads an item written by T_i then T_i commits before T_j reads that item.
   - A schedule is *strict* if a value written by a transaction T is not read or overwritten by other transactions until T either aborts or commits.

   (a) T1:W(X), T2:R(X), T1:W(X), T2:Commit, T1:Commit

   (b) T1:R(X), T2:W(X), T1:W(X), T2:Abort, T1:Commit

   (c) T1:R(X), T2:W(X), T1:W(X), T2:Commit, T1:Commit

   (d) T1:R(X), T2:R(X), T1:W(X), T1:Commit, T2:W(X), T2:Commit

   (e) T1:W(X), T2:R(X), T1:W(X), T2:Commit, T1:Abort

   (f) T1:W(X), T2:R(Y), T1:R(Y), T2:R(X), T1:Commit, T2:Commit

   (g) T1:R(X), T2:R(X), T1:Commit, T2:W(X), T2:Commit

   Display your answer in the table below, indicating 'y' whenever the property is satisfied by the schedule.

	c-serializable	recoverable	cascadeless	strict
a)				
b)				
c)				
d)				
e)				
f)				
g)				

Solution:

	c-serializable	recoverable	cascadeless	strict
a)				
b)		y	y	
c)		y	y	
d)		y	y	y
e)				
f)	y	y		

g)	y	y	y	y

2. Consider the following sequence of log records from a database.

```
<T0, start>
<T0, A, 0 -> 5>
<T0, B, 500 -> 100>
<T0, commit>
<T1, start>
<T2, start>
<T1, A, 5 -> 10>
<checkpoint {T1, T2}>
<T2, B, 100 -> 200>
<T2, commit>
<T3, start>
<T3, C, 30 -> 40>
<CLR T1, A, 5>
<T1, abort>
```

(a) Suppose there is a crash immediately after <T1, abort> is written to the log. Illustrate what happens during recovery following the recovery protocol described in class and also described in the readings (p. 736-737).

Specifically, tell me the sequence of writes – to *both* the database and the log – that occur when the database recovers from the crash. Use the following notation:

- WO(X, V) denotes that the value V is Written for item X and flushed Out to disk
- entries in the log can be written using the format shown above (which is similar to that used in the book and in class).

Solution:

```
#redo phase
WO(B, 200)
WO(C, 40)
WO(A, 5)
# undo phase: T3 on undo list
<CLR T3, C, 30>    # should come before WO(C,30), write ahead logg
WO(C, 30)
<T3, abort>
```

(b) Repeat the previous question except suppose there was a crash immediately after <CLR T1, A, 5> and therefore <T1, abort> was not written to the log.

Solution:

```
# same as above plus
<CLR T1, A, 5>   # write to log first
WO(A, 5)
<T1, abort>
```

(c) In the above examples, the recovery phase should include `WO(B, 200)` even though the log says that T2 has committed. We could make the recovery phase more efficient by not redoing the actions of committed transactions. Is this a good idea? Briefly explain.

Solution: It's necessary to redo unless the database applies a force policy.

If it does not use a force policy then even though T2 has committed, the database disk block might still have been in memory when the system crashed (the DBMS implements a no force policy); thus, the B,200 might not yet be written to the DB disk.

(d) One advantage of checkpointing is that less work needs to be done during recovery. Another advantage is that it is not necessary to retain the *entire* log, which can take up a lot of disk space for a long-lived database. What part of log can be discarded? Hint: the answer is *not* everything before the last checkpoint.

Solution: Can delete everything before the earliest <Ti, start> for any Ti in the checkpoint list. For this example, can delete everything above <T1, start>.

3. Consider the following sequence of log records.

```
<T2, start>
<T2, B, 5 -> 10>
<T1, start>
<T1, A, 10 -> 20>
<checkpoint {T1, T2}>
<T2, A, 20 -> 30>
<CLR T2, A 20>
<CLR T1, A 10>
<T1, abort>
```

(a) Illustrate what happens during recovery. Use the same notation as with the previous problem.

Solution:

```
#redo phase from checkpoint
WO(A, 30)
WO(A, 20)
WO(A, 10)
# undo phase: T2 on undo list
<CLR T2, A, 20>
WO(A, 20)
<CLR T2, B, 5>
WO(B, 5)
<T2, abort>
```

(b) The recovery protocol leaves the database in an inconsistent state. What are the values of A and B after recovery? If recovery was done correctly what would the values of A and B be?

Solution: $A = 20$ but it should be 10. $B = 5$ and that's fine. The problem is that there was a crash during recovery and T1 was written as aborted but T2 was not. This is a problem because T2 modified A after T1.

(c) The reason for the inconsistency is that recovery protocol makes an assumption about database modifications and that assumption does not hold above sequence. What is the assumption? (See p. 736 from the Boat book reading)

Solution: A "data item that has been updated by an uncommitted transaction cannot be modified by any other transaction, until the first transaction has either committed or aborted."

(d) Explain why the log sequence above can *never* happen under strict 2PL.

Solution: T1 released the lock on A before committing!