1. Consider the following two transactions. Assume that the two transactions preserve database consistency in isolation.

T_1: $r_1(A)$; $w_1(A)$; $r_1(B)$; $w_1(B)$;
T_2: $r_2(A)$; $w_2(A)$; $r_2(B)$; $w_2(B)$;

Give all possible schedules (including serial schedule) of the two transactions that are conflict-equivalent to the serial schedule (T_1, T_2).

r1(A)w1(A)r1(B)w1(B)r2(A)w2(A)r2(B)w2(B)
r1(A)w1(A)r1(B)r2(A) w1(B)w2(A)r2(B)w2(B)
r1(A)w1(A)r2(A)r1(B)w1(B)w2(A)r2(B)w2(B)
r1(A)w1(A)r2(A)r1(B)w2(A)w1(B)r2(B)w2(B)
r1(A)w1(A)r2(A)w2(A)r1(B)w1(B)r2(B)w2(B)
r1(A)w1(A)r1(B)r2(A)w2(A)w1(B)r2(B)w2(B)

2. Consider the following two transactions and some possible schedules.

T_1: $r_1(x)$; $w_1(x)$; $r_1(y)$; $w_1(y)$; c_1;
T_2: $r_2(x)$; $w_2(x)$; c_2;

S_1: $r_1(x)$; $w_1(x)$; $r_1(y)$; $w_1(y)$; $r_2(x)$; c_1; $w_2(x)$; c_2;
S_2: $r_1(x)$; $r_2(x)$; $w_1(x)$; $r_1(y)$; $w_1(y)$; $w_2(x)$; c_1; c_2;
S_3: $r_2(x)$; $w_2(x)$; $r_1(x)$; $w_1(x)$; $r_1(y)$; c_2; $w_1(y)$; c_1;

For each of the above schedules, indicate whether the schedule is ***recoverable***, ***cascadeless*** and ***conflict serializable***. If the schedule is conflict serializable, show its ***serializability order***.

S_1: recoverable and conflict serializable ($T_1 \rightarrow T_2$)
S_2: cascadeless but not conflict serializable
S_3: recoverable and conflict serializable ($T_2 \rightarrow T_1$)

3. Consider using ***two-phase locking*** on the following schedule of two transactions, T_1 and T_2.

$$r_1(A); r_2(B); r_2(A); r_1(B); w_1(B); r_2(C);$$

(a) Insert ***shared*** lock (*sl*), ***exclusive*** lock (*xl*) and unlock (*ul*) instructions at appropriate locations.

(b) Describe the execution of the above schedule, assuming that a waiting transaction does not block the following non-conflicting instructions of other transactions. Show the resultant schedule, if it is different from the one you constructed in (a).

(c) What will be the ***serializability order*** of the resultant schedule?

(d) Any changes if the schedule is modified to insert $w_2(A)$ at the end as follows?

$$r_1(A); r_2(B); r_2(A); r_1(B); w_1(B); r_2(C); w_2(A);$$

(a)

$sl_1(A); r_1(A); sl_2(B); r_2(B); sl_2(A); r_2(A); xl_1(B); r_1(B); w_1(B); ul_1(A); ul_1(B); sl_2(C); r_2(C); ul_2(B); ul_2(A); ul_2(C);$

(b)

(i) T_1 is delayed. T_1 cannot get the exclusive lock on B because T_2 already has a shared lock on B.

(ii) T_2 completes and unlocks all.

(iii) T_1 then can get the exclusive lock on B and completes.

Resultant schedule:

$sl_1(A); r_1(A); sl_2(B); r_2(B); sl_2(A); r_2(A); xl_1(B); sl_2(C); r_2(C); ul_2(B); ul_2(A); ul_2(C); r_1(B); w_1(B); ul_1(A); ul_1(B);$

(c) The serializability order is $(T_2 \rightarrow T_1)$.

(d) The execution can result in deadlock, as shown in the following partial schedule.

$$sl_1(A); r_1(A); sl_2(B); r_2(B); xl_2(A); xl_1(B);$$