1 Serializability

Consider the transactions T_1, T_2, T_3, T_4 below where R(·) and W(·) stand for 'Read' and 'Write', respectively.

time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
T_1			W(E)		R(D)				W(A)	
T_2		W(B)		R(A)						
T_3								R(E)		W(C)
T_4	W(D)					R(B)	R(C)			

Table 1: A schedule with 4 transactions

Draw the correct dependency graph for the schedule.

Answer all the following.

1. Is this schedule serial?
2. Is the schedule conflict serializable?
3. What is the minimum number of transactions that need to be removed to produce a conflict serializable schedule? (Zero if the original was already conflict serializable).
4. Is the schedule above conflict equivalent to the schedule T_2, T_4, T_1, T_3. Explain your answer.

2 Locking

2.1 Exercise

Rewrite the schedule in Table 1 adding where needed Shared or Exclusive lock request and the corresponding unlock actions. Use 2-Phase locking, but not strict strong 2PL, so release locks as soon as possible. Remember to show the actions of the lock manager. How will the schedule evolve? Will it generate a deadlock? Will it generate dirty reads? Explain your answer.

2.2 Exercise

Consider the the transactions T_1, T_2, T_3, T_4 below where R(·) and W(·) stand for 'Read' and 'Write', respectively.

time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}
T_1	W(E)				R(E)		R(D)				W(A)	
T_2				W(B)		R(A)						
T_3			W(B)							R(E)		W(C)
T_4		W(D)						R(B)	R(C)			

Table 2: A schedule with 4 transactions

Rewrite the schedule in Table 1 adding where needed Shared or Exclusive lock request and the corresponding unlock actions. Use strict strong 2-Phase locking. Remember to show the actions of the lock manager. How will the schedule evolve? Will it generate a deadlock? Will it generate dirty reads? Explain your answer.