Problem A: 28 points–4 points each question. Whenever applicable show the appropriate graph to justify your answer.

Consider the following schedule, where T1 starts before T2 which starts before T3

T1	T2	T3
start		
write(A)		
	start	
		start
	read(A)	
		write(C)
read (C)		
	read(C)	
write(B)		
		read(A)

Please answer the following questions:

Q1. Is this schedule conflict-serializable (explain your answer with a graph)?

Answer: No. Check conflict edges in precedence graph.

Q2. Show what will happen if they try to execute this schedule under strict 2PL and no deadlock prevention (assume that a transaction locks a resource just before it needs it). If there is no deadlock show the complete schedule otherwise prove deadlock by the appropriate graph

Answer: T1 X-lock(A) and execute write(A). Then T2 tries to S-lock(A) and fails. The wait-for graph has one arc from T2 to T1. Thus, T3 can proceed t with X-lock(C) whereby T1 cannot get C: now the wait-for graph has an arc from T1 to T3. But T3 cannot execute its final read(A) and with an arc from from T3 to T1 have deadlock.

Q3. What will happen if we instead execute these transactions using strict 2PL and a wound-wait deadlock prevention scheme (assume that a transaction locks a resource just before it needs it)?

Answer: The read(C) of T1 kills T3. After that, T1 complete and release its X-lock on A, whereby T2 also complete. Then T3 restarts and complete.

Q4. In general, does the protocol used in Q3 guarantee cascadeless schedules?

Answer: Yes, the schedule is recoverable and since no transaction has performed dirty read, there is no cascading of rollbacks.

Q5. What will happen if we instead execute these transactions using a time-stamp based protocol to guarantee the serializability of the transactions? Show the resulting schedule.
Answer: T1 write(A) Wts(A)=T1;

T2 read(A) Rts(T2)= T2

T3 write(C) Wts(C)=T3

T1 read(C) and abort

T2 ead(C) and abort

T3 read(A) and complete

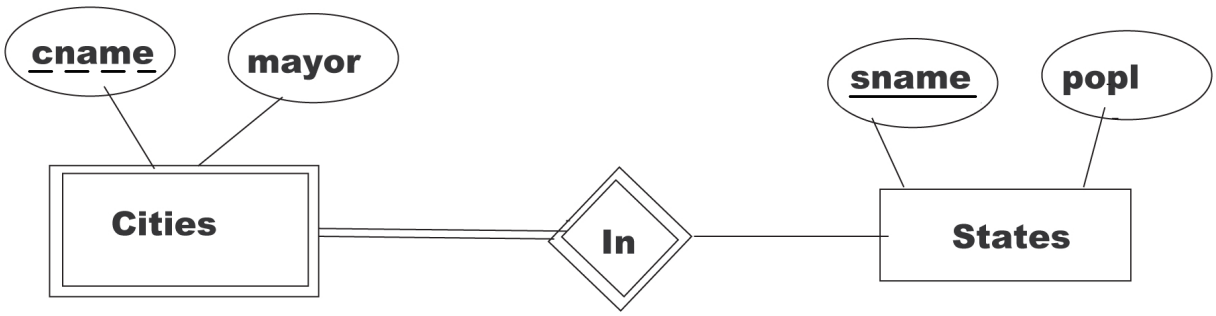
Q6. Does the protocol used in Q5 guarantee freedom from deadlocks (yes/no)? **Answer:**

yes.

Q7. In the wait-die deadlock prevention protocol, transactions that are rolled back keep their old timestamp. What is the reason for that?

Answer: Transactions requesting lock held by older transactions will roll-back; if they can back as a still younger transaction they are likely to have to rollback again, and again ... starvation! However, if they keep their time-stamp they come back with seniority wrt to the others–so they are less likely to roll back again.

Problem B: 24 points: questions 1–6 two points; questions 7–10 three points.
Consider the following ER-diagram:



Mark which of the following six statements are TRUE or FALSE, according to the ER-diagram.
(1) Each City has at most one mayor. **Answer:** T
(2) Two different states cannot have cities that have the same name. **Answer:** F
(3) A city with a given **cname** can be In at most one state. **Answer:** F
(4) No two states can have the same name (i.e., identical **sname**). **Answer:** T
(5) No two cities can share the same mayor. **Answer:** F
(6) No two cities In the same state can have the same name (i.e., identical **cname**). **Answer:** T

Now answer the following four questions about our running example:

Say that we decide to implement the relational database storing the information described by our ER-diagram using the following schema, where the attributes belonging to the key are underlined, `citiesStates(cname, Mayor, sname, popl)`
and answer the following questions:

(7) is this relation 3NF (justify your answer)?

Answer: No sname → popl violates 3NF, since sname is not a key and popl is not contained in a key.

(8) Say that "popl" denotes the state population, as per the last census. Then, what update anomaly will we experience when we have to update with a new census results the **popl** of a state where there are 629 cities?

Answer: For a state where the population changed, we have to update popl in N=629 tuples, where N denotes the number of cities in the state.

(9) Design a BCNF schema, i.e. a set of BCNF relations, that eliminates those anomalies.

Answer: We decompose into a pair of relations:

```
citiesInSts(cname, sname, Mayor), States(sname, popl)
```

(10) Show all the keys and foreign keys (with their referenced attributes) that must declared in your schema to assure that all the constraints defined by the ER-diagram are enforced. (no SQL required)

Answer: First relation, the key should be cname and sname (combined). Second relation, key should be sname. Moreover `sname` in the first relation must be declared FK reference to the second relation—an obvious constraint since according to the ER diagram, this value is imported from States.

Problem C: Transaction Recovery: 20 Points–4 points per question

The log used by the transaction manager contains information about commits, aborts, updates, and checkpoints. Please answer the following questions about checkpoints as TRUE or FALSE and also give justification for your answer:

1) An important function of checkpoints is to manage the cascading of rollbacks for transactions that had read dirty data from other transactions. **Answer:** *No the recovery manager does not know about any dirty read.*

2) A key function of checkpoints is to give the database administrator the ability to check on the current state of transactions and decide whether they must abort of commit. **Answer:** No the recovery is automatic.

3) The main function of checkpoints is to expedite the recovery process. **Answer:** *Yes, so that the recovery manager does not need to walk back to the beginning of the log.*

4) The transaction manager performs (fuzzy) checkpoint operations by forcing each running transaction to commit and making sure that their updates are actually recorded on disk. **Answer:** No the fuzzy checkpoint does not force uncommitted transactions to commit. It only forces the updates of committed transaction into disk.

5) Assume that for unforeseen reasons a few checkpoint records of a lengthy log were damaged in the last crash, and therefore they cannot be read by the recovery manager. If only checkpoint records were damaged, then the recovery manager will be able to complete the recovery process anyway and bring back the database to a consistent state, in spite of such losses.

Answer: *Yes, the recovery manager can skip the damaged checkpoint and keep walking back on the log till it finds a undamaged one.*

Problem D: FDs, and Schema Design: 28 Points–7 points each question:

Given the following FDs for **R(A, B, C, D, E)**

1. AC → B
2. B → A
3. D → E
4. C → D E

Answer the following four questions by considering these FDs in ascending order:

Q1. Is this schema BCNF (justify your answer)? **Answer:** *FD1. (AC)+= {A,C,B, D, E} AC is a key no BCNF violation B+= {B, A} BCNF violated by FD2.*

Q2. Check if these given FDs are in canonical form (only one attribute on the right side) required by the BCNF design algorithm. if they are not replace them with an equivalent set of FDs that are in canonical form. **Answer:** *For canonical form, replace 3 by*
4a. C → D
4b. C → E

Q3. Compute a lossless decompositions of **R(A, B, C, D, E)** *into BCNF relations.* **Answer:** *We can start from 2.*

(B, A) with key B and (B, C, D,E);

3a. C+={C, D,E} a violation because B is not there. Decompose as follows:

(C,D,E) with key C and (C, B) with key (CB)

Now in (C,D,E) let us test 5: D+ ={D, E} but C is not there. Decompose into (D,E) with key D and (D, C) with key C

In summary we got:

R1(B, A) with key B: thus B → A;
R2(C, B) with key CB no FD.
R3(D,E) with key D thus D → E
R4(D,C) with key C. C → D

(Pay attention to students who miss R4 while grading!)

Q4. Is your decomposition FD-preserving (if applicable list the lost FDs)?
Answer:*(AC)+= {A, C, D, E} but B is missing. So, this is not FD-preserving*

Extra Credit Problem : 6 points—each question 2 points.

D is a cohort in a 2PC where A is the coordinator and B and C are the other two cohorts. Each cohort can communicate with the coordinator and the other cohorts, and all the sites and the communication lines operate with great reliability EXCEPT for the communication line between D and A which undergoes failures that require a long time to repair. The distributed DB (DDB) administrator is concerned about what happens when a failure of the line between A and D occurs and ask the following questions. Please, select the correct answer to each question.

EC1. At the end of the 2PC protocol, is it possible for site D to have committed even though A,B,C aborted?

(A) No. Because the coordinator or one of the cohorts aborted, Coordinator A must issue an ABORT request in phase 2 that will undo (ABORT) any commit site D may have made during phase 1.

(B) Yes. Site D can commit once it has received a COMMIT request and release its resources as soon as it has completed the transaction issued by a COMMIT-REQUEST. Aborts by the coordinator or other cohorts will not affect this commit.

(C) Yes. Site D can prepare the transaction after receiving a COMMIT-REQUEST in phase 1. After coordinator A receives an AGREE message from site D, it will issue a COMMIT message to site D to commit the transaction.

Answer: *A*

EC2. If the Coordinator does not get the ready/abort message from a Cohort (say, D) after timeout in phase 1, then:

(A) It can decide to commit independently if it gets "ready" from all other cohorts, and send the decision to D.

(B) It needs to wait indefinitely until it gets decision from all cohorts.

(C) It aborts the transaction only if it gets at least one abort from another cohort.

(D) It can abort the transaction.

Answer: *D*

EC3. If the Cohort site does not get the commit/abort message in phase 2, which of following is invalid?

(A) It can learn the decision from other cohorts.

(B) It can decide whether to commit or abort independently, and send the decision to coordinator.

(C) It must wait until its communication with coordinator recovers.

(D) The coordinator can still consider the transaction is

committed.

Answer: B