

Exercise 1

1 Point

Is the following schedule **conflict serializable**? Draw a precedence graph to verify. If it is not, explain why. If it is, give an equivalent serial schedule.

| T1: | T2: | T3: | T4: |
|----------|----------|---------|----------|
| | | read(A) | |
| read(A) | | | |
| | write(C) | | |
| read(B) | | | |
| write(A) | | | |
| | | | read(C) |
| | | | write(B) |
| | read(A) | | |
| | write(A) | | |
| | | read(B) | |

Exercise 21 Point

Answer the following questions (be concise):

1. Explain three *undesirable phenomena* of concurrent transactions.
2. What does *read uncommitted* stand for?
3. What is a *lost update* and how can it be prevented?

Exercise 3**1 Point**

Consider the following schedule and a two-phase locking scheduler *with* lock conversions. Insert *lock*, *unlock*, and *upgrade* instructions such that all read and write operations in the schedule can be performed in this order.

| T1: | T2: | T3: | T4: |
|----------|---------|----------|----------|
| read(B) | | | |
| | | read(A) | |
| | | | read(A) |
| | read(A) | | |
| write(B) | | | |
| | read(C) | | |
| | | | read(C) |
| | | | write(A) |
| | | write(B) | |

Exercise 4

1 Point

Consider the **snapshot isolation** concurrency control scheme and the following transactions.

| T1: | T2: |
|----------|----------|
| read(A) | |
| | read(B) |
| read(B) | |
| | read(A) |
| write(B) | |
| COMMIT | |
| | write(A) |
| | COMMIT |

- Can T1 and T2 both commit? Explain why.
- What would using `select ... for update` change in this scenario?

Exercise 5

1 Point

Consider the following schedule. Indicate what happens at each step when the schedule is processed by a **multiversion timestamp-ordering** scheduler. The transactions start in order with $TS(T1)=1$, $TS(T2)=2$, $TS(T3)=3$, $TS(T4)=4$. Assume that no versions of data item A exist. State if a transaction must be rolled back and also consider cascading rollbacks.

| T1: | T2: | T3: | T4: |
|----------|----------|----------|---------|
| write(A) | | | |
| | read(A) | | |
| | | write(A) | |
| | | | read(A) |
| | write(A) | | |
| write(A) | | | |

Exercise 6

1 Point

Consider the following schedule and the **two-phase locking** scheduler.

- (a) Assume that all transactions want to issue a **write** operation on item **A** in the following order: **T1**, **T2**, **T3**. Complete the schedule by inserting lock requests (e.g., **R:lock-S(A)**) and granted locks (e.g., **G:lock-S(A)**) for all remaining **read** and **write** operations. Denote in the schedule if a transaction has to wait.
- (b) Does the schedule result in a deadlock? Why?
- (c) Briefly explain the *wound-wait* deadlock prevention scheme on this example by assuming $TS(T1) < TS(T2) < TS(T3)$.

| T1: | T2: | T3: |
|-------------|---------|---------|
| <hr/> | | |
| R:lock-S(A) | | |
| G:lock-S(A) | | |
| read(A) | | |
| <hr/> | | |
| | read(A) | |
| <hr/> | | |
| | | read(A) |
| <hr/> | | |
| <hr/> | | |
| <hr/> | | |
| <hr/> | | |

Exercise 7

1 Point

With the following starting values: A=10, B=20, C=30, D=40, E=50, F=60, write the log file (physical logging) for the following schedule including the log records generated during recovery. After the redo phase, which transactions are contained in the undo list?

```

T1:          T2:          T3:
start
read(A)
A:=A-5
write(A)

                start
                read(C)

read(B)
B:=B-15
write(B)
read(D)

                C:=C+20
                write(C)

                                start
                                read(E)

                COMMIT

-----CHECKPOINT-----
D:=D-20
write(D)

                                E:=E-15
                                write(E)

read(F)

                                COMMIT

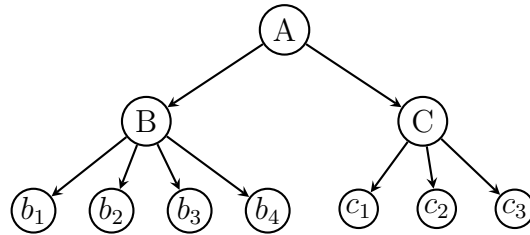
F:=F-15
write(F)
-----CRASH-----

```


Exercise 8

1 Point

Consider the **multiple granularity** locking scheme with intentional locks. The following figure shows a data hierarchy: for instance, the root level A is a database, B and C are relations, and b_i, c_i are data items.



Now, consider the following tasks that the transactions should execute.

- T1: Update a value of data item c_3 .
 - T2: Read all data items contained in B such that it is possible to update a few of them that satisfy a specific condition.
 - T3: Read data items b_1 and c_1 .
- (a) Indicate the locks acquired by each of the transactions (assume that no locks are released).
- (b) Which of the transactions can run concurrently, and why?