

Is the following schedule **conflict serializable**? Draw a precedence graph to verify.

T1:	T2:	T3:
		read(B)
		write(A)
read(A)		
	write(A)	
read(B)		
write(A)		
COMMIT		
	read(A)	
	COMMIT	
		read(B)
		COMMIT

Name:

Matrikelnummer:

Exercise 2

1 Point

Show the commit order (by inserting the commit commands into the schedule) for the following schedule such that the schedule is **recoverable**.

T1:	T2:	T3:	T4:
read(B)		write(A)	read(A)
write(B)	read(A)		write(A)
	read(A)		read(B)

Exercise 3**1 Point**

Consider the following intended schedule and a validation-based scheduler. Write if transactions T1, T2, and T3 validate or not. If not, indicate which validation rule is broken.

The start and the finish timestamp of the transactions can be assumed to be the point in time when they do the first resp. last operation in the schedule below.

T1:	T2:	T3:
read(A,B)		
validate		
write(C)		
	read(B,C)	
		read(C)
		validate
		write(C)
	validate	
	write(B)	

Name:

Matrikelnummer:

Exercise 4

1 Point

Is the following schedule valid under the **timestamp-ordering** protocol? If yes, show the timestamps for the affected data items for each operation. If not, mark the first problematic operation and explain the problem.

TS(T1)=1, TS(T2)=2, TS(T3)=3

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

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.

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

                start
                read(C)

read(B)
B:=B-5
write(B)
read(D)

                C:=C+15
                write(C)
                COMMIT
-----CHECKPOINT-----

                start
                read(E)

D:=D-10
write(D)

                E:=E-15
                write(E)
                read(F)

COMMIT

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

Name:

Matrikelnummer:

Exercise 6

1 Point

Consider the following log. What would happen during the recovery if the system crashes at the end of the log?

```
<T2, start>
<T2, 01, operation-begin>
<T2, X, 200, 250>
<T2, 01, operation-end, (X, -50)>
<T1, start>
<T1, Y, 150, 100>
<T2, Z, 10, 150>
<T1, 02, operation-begin>
<T1, X, 250, 50>
-----CRASH-----
```

Exercise 7

1 Point

Consider a multi-granularity locking protocol with lock modes S, X, IS, IX, SIX. The following object hierarchy is given:

- root R with blocks A and B,
- block A contains records a1 and a2,
- block B contains records b1, b2, and b3.

Write the list of locks (objects being locked and lock type) required to execute the following transactions. Start with the root.

T1: modify record a2

T2: read all records in block B

Can T1 and T2 be executed concurrently? Why?

Name:

Matrikelnummer:

Exercise 8

1 Point

Answer the following questions (be concise).

- What is a lost update and how can it be prevented? **(0.25 Punkt)**
- What does Read Uncommitted stand for? **(0.25 Punkt)**
- How can deadlocks be prevented? Give an example technique and briefly describe it. **(0.5 Punkt)**