Task 1 (6 marks)
Concurrent executions of database transactions

Consider the database transactions listed below.

T1	T2	T3
read(x)	read(y)	read(z)
write(y,x+1)	write(x,y+1)	write(x,z+1)
commit	commit	write(y,z+2)
		commit

Assume that the initial values of the persistent data items x, y, and z are the following. x = 1, y = 2, and z= 3.

(1) (2 marks)
Show a sample concurrent execution of the transactions T1, T2, and T3 that is nonconflict serializable and that is view serializable.

Prove, that the execution is nonconflict serializable and that it is view serializable.

When visualizing the concurrent executions use a technique of two-dimensional diagrams presented to you during the lecture classes, for example, see a presentation 10 Introduction to Transaction Processing (1), slide 9.

			persistent		
T1	T2	T3	x	y	z
read(x)			1	2	3
	read(y)		1	2	3
		read(z)	1	2	3
write(y,x+1)			1	2	3
	write(x,y+1)		3	2	3
		write(x,z+1)	4	2	3
		write(y,z+2)	4	5	3
commit					
	commit				
		commit			

The execution above is nonconflict serializable because a conflict between T1:read(x) and T2:writex,y+1) and a conflict between T2:read(y) and T1:write(y,x+1) close a cycle in a conflict serialization graph.

Yet the execution is view serializable because in a serial execution T1, T2, T3 (please see below) the transactions read the same values as in a concurrent execution T1:read(x=1), T1:write(y,2), T3:read(z=3) and the final values of x, and y set by a transaction T3 are the same as in a concurrent execution. A value of z does it change.

			persistent		
T1	T2	T3	x	y	z
read(x)			1	2	3
write(y,x+1)			1	2	3
commit					
	read(y)		1	2	3
	write(x,y+1)		3	2	3
	commit				
		read(z)	1	2	3
		write(x,z+1)	4	2	3
		write(y,z+2)	4	5	3
		commit			

(2) (2 marks)

Show a sample concurrent execution of the transactions `T1`, `T2`, and `T3` that is conflict serializable and that is not order-preserving conflict serializable.

Prove, that the execution is conflict serializable and that it is not order-preserving conflict serializable.

When visualizing the concurrent executions use a technique of two-dimensional diagrams presented to you during the lecture classes, for example, see a presentation 10 Introduction to Transaction Processing (1), slide 9.

T1	T2	T3
		read(z)
read(x)		
write(y,x+1)		
	read(y)	
	write(x,y+1)	
		write(x,z+1)
		write(y,z+2)
commit		
	commit	
		commit

The execution above is conflict serializable because an order of conflicting operations is

`T1:read(x)` → `T2:write(x,y+1)`,
`T1:write(y,x+1)` → `T2:read(y)`,
`T1:read(x)` → `T3:write(x,z+1)`

```
T1:write(y,x+1) → T3:write(y,z+2)
T2:read(y) → T3:write(x,z+1)
T2:write(x,y+1) → T3:write(y,z+2)
```

Hence, there is no cycle in a conflict serialization graph.

Yet the execution is not order-preserving conflict serializable because and order of the transactions determined by timestamps recorded when the transactions started is T3 → T1 → T2.

(3) (2 marks)
Show a sample concurrent execution of the transactions T1, T2, and T3 that is recoverable and that is not strict.

Prove, that the execution is recoverable and that it is not strict.

When visualizing the concurrent executions use a technique of two-dimensional diagrams presented to you during the lecture classes, for example, see a presentation 10 Introduction to Transaction Processing (1), slide 9.

T1	T2	T3
		read(z)
read(x)		
write(y,x+1)		
	read(y)	
	write(x,y+1)	
		write(x,z+1)
		write(y,z+2)
commit		
	commit	
		commit

The execution above is recoverable because a transaction T2 that reads (T2:read(y)) uncommitted modification done by transaction T1 (T1: write(y,x+1)) commits after T2 is committed. Hence, if T2 fails than still T1 can be rolled back.

Yet, the execution is not strict because T2 reads a data item y (T2:read(y)) modified by a transaction T1 (T1: write(y,x+1)) that is not committed yet.