I. Consider relation *Customer[CustomerID, FirstName, LastName, City, DateOfBirth]* and the following interleaved execution in SQL Server:

```
create database DBMS_Sem6_online
go
use DBMS_Sem6_online
go

create table Customer(
CustomerID int primary key identity,
FirstName varchar(50),
LastName varchar(50),
City varchar(50),
DateofBirth date
)

insert into Customer values ('Ana', 'Maria', 'Alba-Iulia', '2000-04-04'), ('Popescu',
'Iulian', 'Timisoara', '1999-05-05')

select * from Customer

-- after each execution
update Customer set City='Timisoara' where CustomerID=2
```

T1	T2
use DBMS_Sem6_online	use DBMS_Sem6_online
go	go
-- first version - as it is	-- first version - as it is
-- second version	-- second version
-- set transaction isolation level READ UNCOMMITTED	-- set transaction isolation level READ UNCOMMITTED
-- third version	-- third version
-- set transaction isolation level READ COMMITTED	-- set transaction isolation level REPEATABLE READ
-- forth version	-- forth version
-- set transaction isolation level REPEATABLE READ	-- set transaction isolation level READ COMMITTED
BEGIN TRAN	BEGIN TRAN
select City from Customer where CustomerID=2	update Customer set City='Cluj-Napoca' where
waitfor delay '00:00:10'	CustomerID=2
update Customer set City='Bucuresti' where	waitfor delay '00:00:10'
CustomerID=2	ROLLBACK TRAN
COMMIT TRAN	

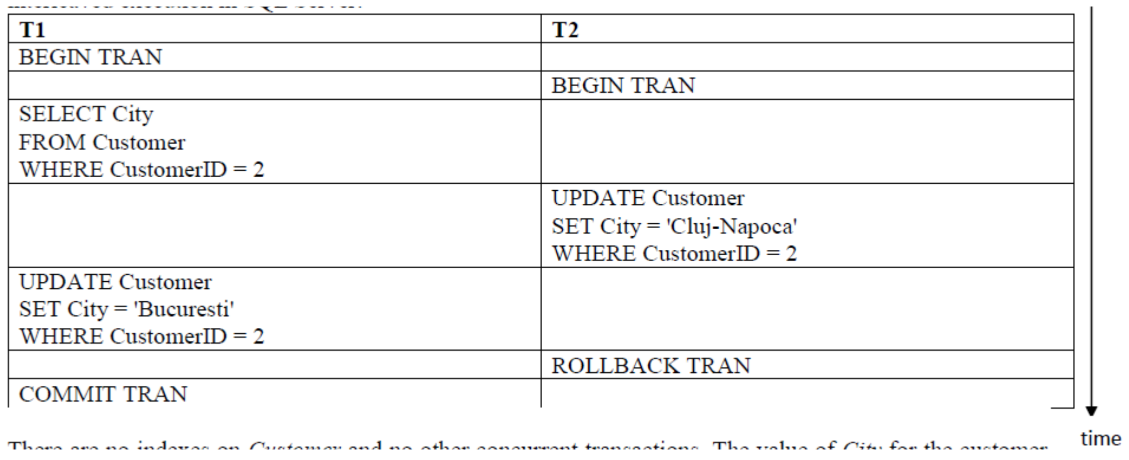
1. T1 and T2 run under READ UNCOMMITTED. After the *COMMIT TRAN* statement in T1, the *City* value for the customer with *CustomerID* 2 is:
a. *Timisoara*
b. *Cluj-Napoca*
c. *Bucuresti*
d. NULL
e. None of the above answers is correct.

2. T1 runs under READ COMMITTED and T2 under REPEATABLE READ. After the *COMMIT TRAN* statement in T1, the *City* value for the customer with *CustomerID* 2 is:
a. *Timisoara*
b. *Cluj-Napoca*
c. *Bucuresti*
d. NULL
e. None of the above answers is correct.

3. T1 runs under REPEATABLE READ and T2 runs under READ COMMITTED. Then:
a. T1 doesn't acquire a shared lock for its SELECT statement.
b. T1 acquires a shared lock for its SELECT statement.
c. T2 needs an exclusive lock for its UPDATE statement.
d. T1 needs an exclusive lock for its UPDATE statement.
e. None of the above answers is correct.

1. c
2. c
3. b, c, d

In this interleaved execution in SQL Server, the order of the operations is:
* the initial value for the record with CustomerID=2 from the table Customer is, for the field City = *Timisoara*



T1	T2
BEGIN TRAN	
	BEGIN TRAN

SELECT City
FROM Customer
WHERE CustomerID = 2	

	UPDATE Customer
	SET City = 'Cluj-Napoca'
	WHERE CustomerID = 2

UPDATE Customer
SET City = 'Bucuresti'
WHERE CustomerID = 2	

	ROLLBACK TRAN
COMMIT TRAN	

- T1, executes the select from the Customer table, getting the shared lock on this table
- The next operation is the update from T2, that is also getting an exclusive lock (we also know, that, exclusive locks protect updates to file resources and can be owned by only one transaction at a time). In this transaction, T2, the table Customer is updated for the record with the CustomerID=2 (the new value for the City becomes *Cluj-Napoca*).

- The next operation is the update from T1, that also needs an exclusive lock. In this transaction, T1, the table Customer is updated for the record with the CustomerID=2 (the new value for the City becomes *Bucuresti*).

So, for the update operation, both transactions need the exclusive lock.

READ LOCK = SHARED LOCK (S) = SELECT statement

WRITE LOCK = EXCUSIVE LOCK (X) = UPDATE statement / INSERT statement / DELETE statement

From Seminar 3:

- READ UNCOMMITTED: no S locks when reading data
- READ COMMITTED: S locks - released as soon as the SELECT operation is performed; X locks - released at the end of the transaction
- REPEATABLE READ: holds S locks and X locks until the end of the transaction
- SERIALIZABLE: holds locks (including key-range locks) during the entire transaction