T1	T2
X(A) R(A) W(A) U(A)	
	X(A) R(A) W(A) X(B) R(B) W(B) U(A) U(B) Commit
Abort	

Operation	A			B			C		
	RTS	WTS	C	RTS	WTS	C	RTS	WTS	C
R1(A)	1	0	True	0	0	True	0	0	True
R2(B)	1	0	True	2	0	True	0	0	True
R3(B)	3	0	True	2	0	True	0	0	True
W1(A) Abort	3	0	True	2	0	True	0	0	True
R2(C)	3	0	True	2	0	True	2	0	True
W3(B)	3	0	True	2	3	False	2	0	True
W2(C)	3	0	True	2	3	False	2	2	False
C1	3	0	True	2	3	False	2	2	False
R2(A)	3	0	True	2	3	False	2	2	False
W3(C)	3	0	True	2	3	False	2	3	False
C3	3	0	True	2	3	True	2	3	False
W2(B) Ignore	3	0	True	2	3	True	2	3	True
C2	3	0	True	2	3	true	2	3	True

Operation	A			B			C		
	RTS	WTS	C	RTS	WTS	C	RTS	WTS	C
R1(A)	2	0	True	0	0	True	0	0	True
R2(B)	2	0	True	3	0	True	0	0	True
R3(B)	2	0	True	3	0	True	0	0	True
W1(A)	2	2	False	3	0	True	0	0	True
R2(C)	2	2	False	3	0	True	3	0	True
W3(B) rollback	2	2	False	3	0	True	3	0	True
W2(C)	2	2	False	3	0	True	3	3	False
C1	2	2	False	3	0	True	3	3	False
R2(A)	3	2	True	3	0	True	3	3	False
W3(C) reject	3	2	True	3	0	True	3	3	False
C3 reject	3	2	True	3	0	True	3	3	False

W2(B)	3	2	True	3	3	False	3	3	False
C2	3	2	True	3	3	True	3	3	true

T1	T2
X(A) R(A) W(A) X(B) U(A)	
	X(A)
R(B) W(B) U(B)	
	X(B) R(B) R(A) W(A) U(A) W(B) U(B)

Q6-7

Yes, it is allowed by two-phase locking. The schedule with lock requests and releases obeying 2PL:

T1: X(A), R(A), W(A), U(A), Abort T1

T2: X(A), R(A), W(A), X(B), R(B), W(B), U(A), U(B), Commit T2

T1	T2
X(A) R(A) W(A) U(A)	
	X(A) R(A) W(A) X(B) R(B) W(B) U(A) U(B) Commit
Abort	

Q8

a) Yes, it is guaranteed since both transactions follow strict two-phase locking.

b) Yes, deadlock is possible. It can happen in the case that T1 starts first and lock A and T2 starts and lock B. Then deadlock will happen since T1 is waiting for releasing lock B and T2 is waiting for releasing lock A. In the wait-die deadlock prevention scheme latter transaction should be aborted, so T2 would be aborted.

c) It is not possible. Since transactions follow strict two-phase locking and strict schedules are cascadeless, cascading rollback is not possible.

d) Both transactions are two-phase locking since they don't demand new locks after acquires locks. In 2PL protocol, it is guaranteed that the schedule is conflict serializable.

e)Both transactions want to lock A first. T1 locks A first and then locks B. After releasing A, T2 locks A. So, deadlock is not possible.

f)It is possible since locks are released before commits. This cause that one of the transactions could write or read uncommitted data. Thus, if one of the transactions is aborted, the other one should be aborted too. The scenario that would show cascading rollback:

Assume T1 reads and writes on A, then T2 reads and writes on A. In later steps, if T1 is failed and should be aborted and rollbacked, we should roll back T2 too. Example schedule with the given scenario:

T1	T2
X(A) R(A) W(A) X(B) U(A)	
	X(A)
R(B) W(B) U(B)	
	X(B) R(B) R(A) W(A) U(A) W(B)

	U(B)

Yellow ones reads and write uncommitted data.

Q9

a)

i)

Operation	A			B			C		
	RTS	WTS	C	RTS	WTS	C	RTS	WTS	C
R1(A)	1	0	True	0	0	True	0	0	True
R2(B)	1	0	True	2	0	True	0	0	True
R3(B)	3	0	True	2	0	True	0	0	True
W1(A) Abort	3	0	True	2	0	True	0	0	True
R2(C)	3	0	True	2	0	True	2	0	True
W3(B)	3	0	True	2	3	False	2	0	True
W2(C)	3	0	True	2	3	False	2	2	False
C1	3	0	True	2	3	False	2	2	False
R2(A)	3	0	True	2	3	False	2	2	False
W3(C)	3	0	True	2	3	False	2	3	False
C3	3	0	True	2	3	True	2	3	False
W2(B) Ignore	3	0	True	2	3	True	2	3	True
C2	3	0	True	2	3	true	2	3	True

ii)

Operation	A			B			C		
	RTS	WTS	C	RTS	WTS	C	RTS	WTS	C
R1(A)	2	0	True	0	0	True	0	0	True
R2(B)	2	0	True	3	0	True	0	0	True
R3(B)	2	0	True	3	0	True	0	0	True
W1(A)	2	2	False	3	0	True	0	0	True
R2(C)	2	2	False	3	0	True	3	0	True
W3(B) rollback	2	2	False	3	0	True	3	0	True
W2(C)	2	2	False	3	0	True	3	3	False
C1	2	2	True	3	0	True	3	3	False
R2(A)	3	2	True	3	0	True	3	3	False
W3(C) reject	3	2	True	3	0	True	3	3	False
C3 reject	3	2	True	3	0	True	3	3	False

W2(B)	3	2	True	3	3	False	3	3	False
C2	3	2	True	3	3	True	3	3	true

b)Commit bit is used for preventing cascading aborts and dirty reads. Thanks to the commit bit, schedules can be recoverable.