In this question, do NOT assume that strict 2 phase locking is used. Rather you can assume that locks are released as soon as they are no longer needed (after a read for shared, and after a write for exclusive)
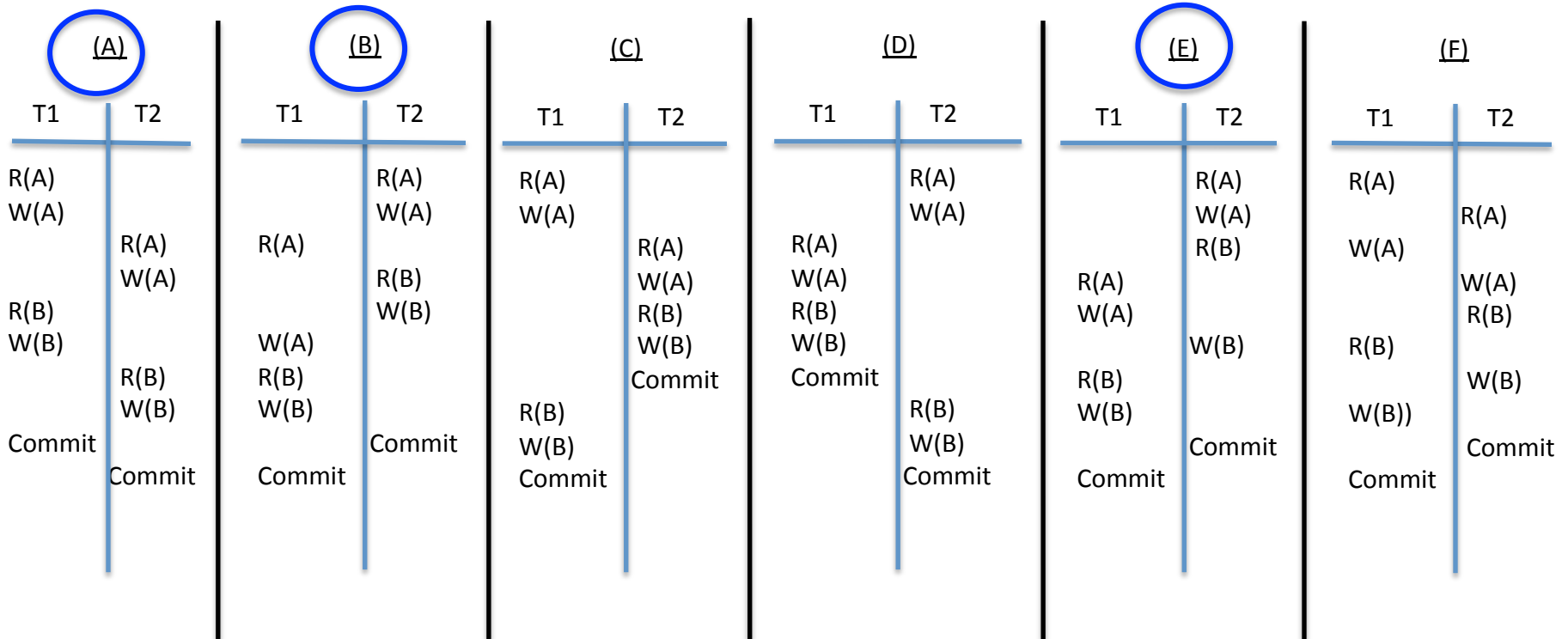
1. (5 points) Consider the following two transactions, T1 and T2:

T1: Read(A), $Op_{11}(A)$, Write(A), Read(B), $Op_{12}(B)$, Write(B), Commit

T2: Read(A), $Op_{21}(A)$, Write(A), Read(B), $Op_{22}(B)$, Write(B), Commit

+3 for one, + 4 for two, +5 for three

-1 for one incorrect circled, -2 for two, -5 for three

0 min, **5** max

Identify all schedules (just showing disk reads and writes) that clearly result in *serializable* behavior, under the assumption of no system failures and transaction rollbacks, even without knowing when the Ops are performed. **A and B are distinct objects, but are shared across transactions.**

(A)		(B)		(C)		(D)		(E)		(F)	
T1	T2	T1	T2	T1	T2	T1	T2	T1	T2	T1	T2
R(A)			R(A)	R(A)			R(A)		R(A)	R(A)	
W(A)			W(A)	W(A)			W(A)		W(A)		R(A)
	R(A)	R(A)			R(A)	R(A)			R(B)		W(A)
	W(A)		R(B)		W(A)	W(A)		R(A)			R(B)
			W(B)		R(B)	R(B)		W(A)			
R(B)					W(B)	W(B)			W(B)	R(B)	
W(B)		W(A)			Commit	Commit		R(B)			W(B)
	R(B)	R(B)		R(B)			R(B)	W(B)		W(B))	
	W(B)	W(B)		W(B)			W(B)		Commit		Commit
Commit			Commit	Commit			W(B)	Commit		Commit	
	Commit	Commit				Commit		Commit			

2. (5 points) Consider the following two transactions, T1 and T2:

T1: Read(A), $Op_{11}(A)$, Write(A), Read(B), $Op_{12}(B)$, Write(B), Commit

T2: Read(A), $Op_{21}(A)$, Write(A), Read(B), $Op_{22}(B)$, Write(B), Commit

Suppose that A=5 and B=2 just before T1 and T2 are initiated.

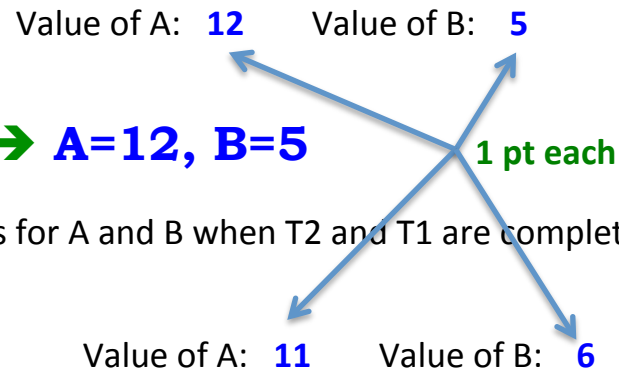Suppose Op11(A) = A+1; Op12(B)=B*2; Op21(A)=2*A; Op22(B)=1+B

So, we have

T1: Read(A), **A+1**, Write(A), Read(B), **B*2**, Write(B), Commit

T2: Read(A), **2*A**, Write(A), Read(B), **1+B**, Write(B), Commit

a) Under the conditions above, what are the resulting values for A and B when T1 and T2 are completed, with the entirety of T1 completed before T2 initiated.

All of T1 followed by all of T2:   Value of A: **12**   Value of B: **5**

A=5, B=2 ➔ T1 ➔ A=6, B=4 ➔ T2 ➔ **A=12, B=5**   **1 pt each**

b) Under the conditions above, what are the resulting values for A and B when T2 and T1 are completed, with the entirety of T2 completed before T1 initiated.

All of T2 followed by all of T1:   Value of A: **11**   Value of B: **6**

A=5, B=2 ➔ T2 ➔ A=10, B=3 ➔ T1 ➔ **A=11, B=6**

Suppose that A=5 and B=2 before T1 and T2; suppose Op11(A) = A+1; Op12(B)=B*2; Op21(A)=2*A; Op22(B)=1+B

2c Consider the following two transactions, T1 and T2:

T1: Read(A), $Op_{11}(A)$, Write(A), Read(B), $Op_{12}(B)$, Write(B), Commit

T2: Read(A), $Op_{21}(A)$, Write(A), Read(B), $Op_{22}(B)$, Write(B), Commit

What are the values of a and b under the the schedule below

Value of A: **12**

1 pt each

(A)		
T1	T2	
R(A) W(A)		
	R(A) W(A)	
R(B) W(B)		
	R(B) W(B)	
Commit		
	Commit	
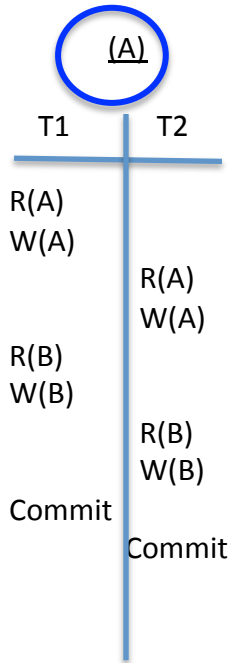
A=5, B=2

Value of B: **5**

→ T1(A) → Read(A), A+1, Write(A) → A=6

→ T2(A) → Read(A), 2*A, Write(A) → **A=12**

Same as T1, T2

→ T1(B) → Read(B), B*2, Write(B) → B=4

→ T2(B) → Read(B), 1+B, Write(B) → **B=5**

Generally, consider T1 and T2 simplified into two transactions, T(A) and T(B), based on each shared object, A and B. If simplified, but still dependent, transactions follow same serial order, such as T1(A),T2(A) and T1(B),T2(B), then the schedule is serializable (assuming no system failures and rollbacks).

Suppose that A=5 and B=2 before T1 and T2; suppose Op11(A) = A+1; Op12(B)=B*2; Op21(A)=2*A; Op22(B)=1+B

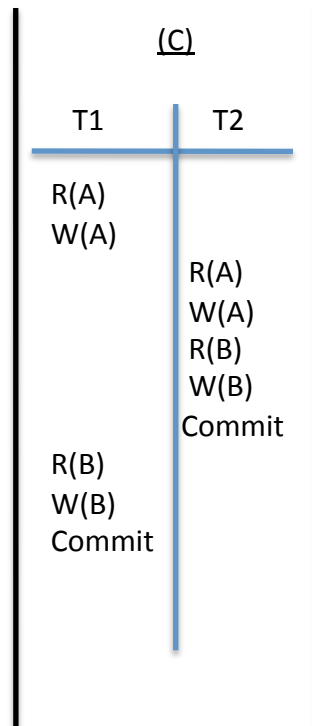**2d** Consider the following two transactions, T1 and T2:

T1: Read(A), $Op_{11}(A)$, Write(A), Read(B), $Op_{12}(B)$, Write(B), Commit

T2: Read(A), $Op_{21}(A)$, Write(A), Read(B), $Op_{22}(B)$, Write(B), Commit

What are the values of a and b under the the schedule below
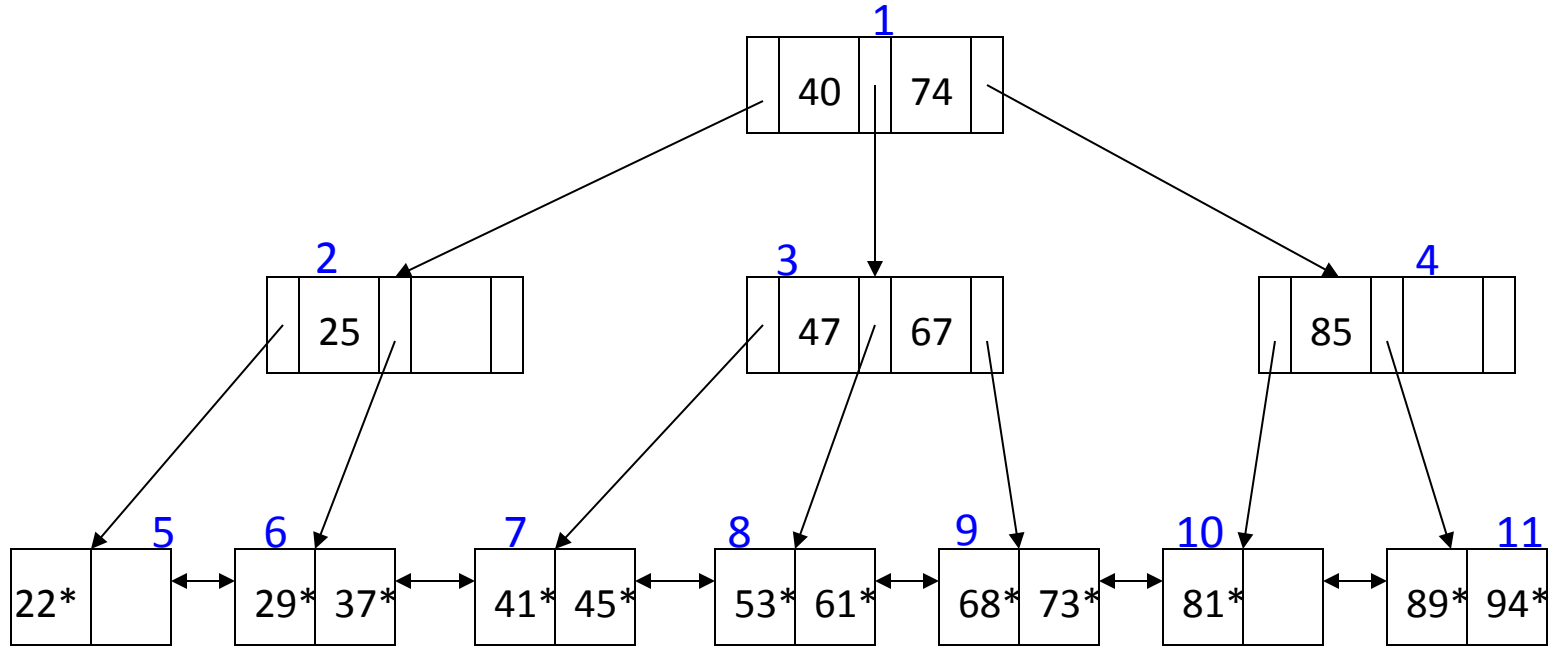
Value of A: **12**

Value of B: **6**

1 pt each

(C)

T1	T2
R(A)	
W(A)	
	R(A)
	W(A)
	R(B)
	W(B)
	Commit
R(B)	
W(B)	
Commit	

A=5, B=2

➔ T1(A) ➔ A=6, B=2

➔ T2(A) ➔ **A=12**
➔ T2(B) ➔ B=3

➔ T1(B) ➔ **B=6**

3. (10 pts) Consider the B+ tree index for attribute A of table T. Above each node is a numeric label for the node (1 through 11), which you will use in answering this question.

1

| | 40 | | 74 | |

2

| | 25 | | |

3

| | 47 | | 67 | |

4

| | 85 | | |

5

| 22* | |

6

| 29* | 37* |

7

| 41* | 45* |

8

| 53* | 61* |

9

| 68* | 73* |

10

| 81* | |

11

| 89* | 94* |

For each of the following operations, list the *nodes* (by label), in proper order, that would be locked (shared or exclusive) in strict two-phase locking (2PL) when performing the respective operation. If no nodes need be locked, then write *None*. Ignore data nodes, which are not shown, and do not list new nodes that might be introduced. Assume that this index for attribute A is used in evaluating each operation below. Write S(label) for a shared lock, and X(label) for an exclusive lock. Note that the same node, A, may be listed twice, first as S(A) then as X(A), for the same operation. Do not show order of lock release (we'll assume all locks released at end of operation, upon commit). Treat each operation as independent, and not as a sequence of actions. Assume that redistribution is never used.

(a) SELECT T.A FROM T WHERE T.A > 75 : **S(1), S(4), S(10), S(11)** **2 pts for (a)**

(b) UPDATE T SET T.B = T.B + 100 WHERE T.A = 37 : **S(1), S(2), S(6)** **2 pts for (b)**

(c) UPDATE T SET T.A = T.A + 5 WHERE T.A = 29 : **S(1), S(2), S(6), X(6)** **3 pts for (c)**

(d) INSERT INTO T (A, B, C) VALUES (70, 20, 10) : **S(1), S(3), S(9), X(9), X(3), X(1)** **3 pts for (d)**
subtle: X(8)and/or X(10) too so as to update ptr to 9 (ok if missing)

In most cases, probably full credit for selected variations, notably skipping over S(k) straight to X(k)