In this question, do not assume that strict 2 phase locking is used. Rather you can assume that locks are released as when they are no longer absolutely needed (after a read for shared, and after a write for exclusive)

9. (4 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

+2 for one, + 3 for two, +4 for three
-1 for one incorrect circled, -2 for two,
-4 for three
0 min, 4 max

Circle all schedules (just showing disk reads and writes) that clearly result in *serializable* behavior even without knowing when the Ops are performed. **A and B are distinct (but are shared across transactions).**

[✓ (A)]

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

[✓ (B)]

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

(C)

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

(D)

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

[✓ (E)]

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

(F)

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

(G) NONE OF THE ABOVE

0 total for question if this circled

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

9. **(4 points)** Consider the following two transactions, T1 and T2:

+2 for one, + 3 for two, +4 for three
-1 for one incorrect circled, -2 for two,
-4 for three
0 min, 4 max

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

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

Circle all schedules (just showing disk reads and writes) that clearly result in *serializable* behavior even without knowing when the Ops are performed. **A and B are distinct.**

All of T1 followed by all of T2:

A=5, B=2 $\rightarrow$ T1 $\rightarrow$ A=6, B=4 $\rightarrow$ T2 $\rightarrow$ **A=12, B=5**

All of T2 followed by all of T1:

A=5, B=2 $\rightarrow$ T2 $\rightarrow$ A=10, B=3 $\rightarrow$ T1 $\rightarrow$ **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

9. (4 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

+2 for one, + 3 for two, +4 for three
-1 for one incorrect circled, -2 for two,
-4 for three
0 min, 4 max

Circle all schedules (just showing disk reads and writes) that clearly result in *serializable* behavior even without knowing when the Ops are performed. **A and B are distinct (but are shared across transactions).**

[✓ (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

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

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

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

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

Same as T1, T2

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 .

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

9. (4 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

+2 for one, + 3 for two, +4 for three
-1 for one incorrect circled, -2 for two,
-4 for three
0 min, 4 max

Circle all schedules (just showing disk reads and writes) that clearly result in *serializable* behavior even without knowing when the Ops are performed. **A and B are distinct (but are shared across transactions).**

(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

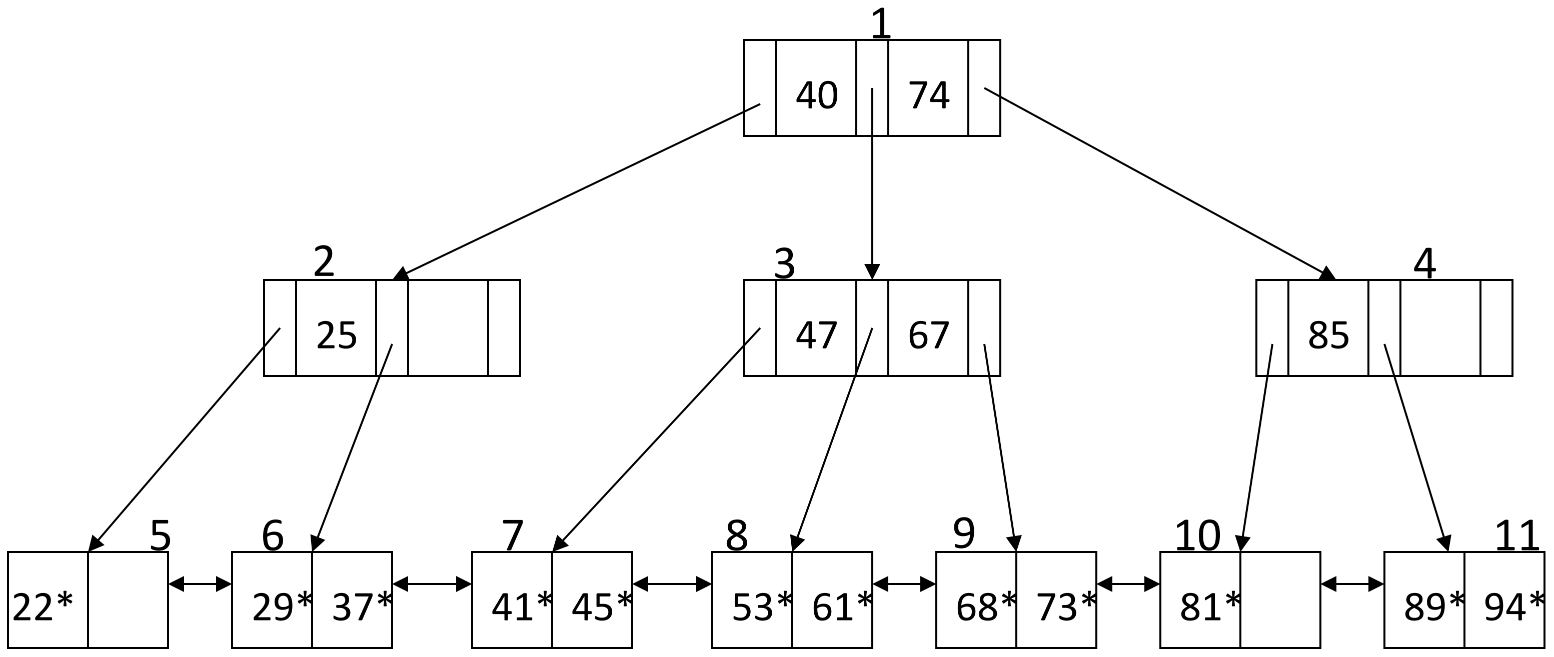
$\rightarrow$ T1(A) $\rightarrow$ A=6, B=2

$\rightarrow$ T2(A) $\rightarrow$ **A=12**

$\rightarrow$ T2(B) $\rightarrow$ B=3

$\rightarrow$ T1(B) $\rightarrow$ **B=6**

21. (4 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.



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)**

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

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

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

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