Zhimin Wang

zhimanw2

CS 411: Database Systems

Fall 2021

Homework 3 (Due by 23:59 CT on Oct 27, 2021)

Logistics

1. This homework is due on Oct 27st, 2021 at 23:59 CT. **We DO NOT accept late homework submissions.**
2. You will be using Gradescope to submit your solutions. Answer each sub-question (e.g. “a.”, “b.” etc.) on a **new page** and submit your solution as a single PDF file on Gradescope. All registered students should have received an email invitation to Gradescope. Please submit the PDF to "Homework 2".
3. ***IMPORTANT*: Please make sure to link PDF pages with the corresponding question outline on Gradescope.**
4. The answers can be written electronically or they can be hand-written, but if we cannot read your submissions, we won’t be able to grade them.
5. Please write down any intermediate steps to receive full credit.
6. Keep your solutions brief and clear.
7. Please use Campuswire if you have questions about the homework but **do not post answers.** Feel free to use private posts or come to office hours.
8. DO NOT PLAGIARIZE. The following are written in the course syllabus:

> 1. **Assignments are individual work.**
> 2. Collaboration is NOT allowed when working on the assignments.
> 3. Discussions are allowed if and only if these discussions regard only high-level concepts and general ideas. Discussion cannot involve answers to the questions on the homework. Checking answers/part of the solutions among peers are **not** allowed. Sharing answers on any public/private electronic platform, including but not limited to email, messenger, Facebook groups, discord chat, etc., are **not** allowed.
> 4. If you discussed questions with your classmates, you **must** include their names and the questions you discussed. Not including students' names will be considered a violation of the course's academic integrity policy. This rule applies to all individual homework assignments, including MPs.
> 5. You should reference (in your code as comments) any code or concepts copied from StackOverflow or any other online resources. However, 80% of the code you turn in must be your own code.
> 6. You are allowed to submit regrade requests within the time frame listed on Campuswire. Typically we allow up to one week after the HW grades are released if not explicitly mentioned.
> 7. Uploading your assignment questions to public platforms (i.e., shared drive, course hero, etc.) is prohibited. Such violations are copyright infringements and possible violations of academic honesty. We will process these strictly.

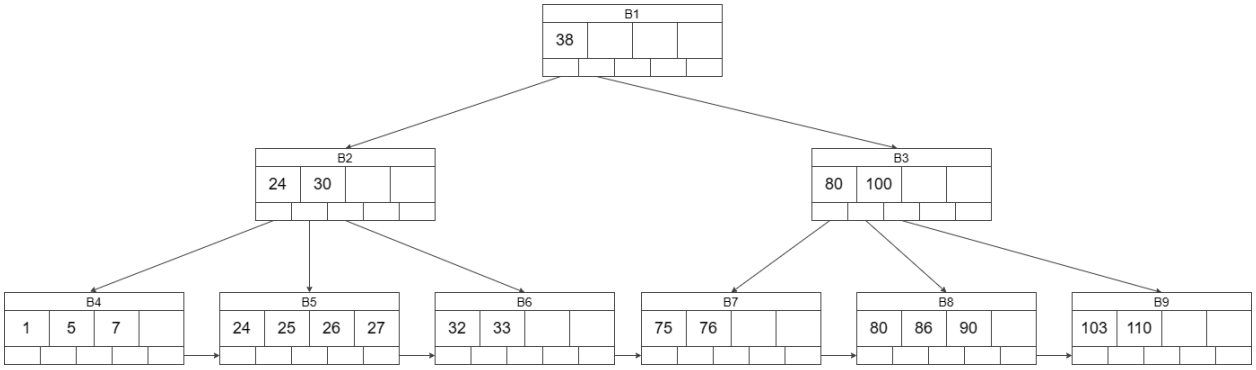
Rubric (IMPORTANT!)

1. Always write intermediate steps.
2. Explain your answer with as much detail as possible

Problem 1 B+ Tree

Consider the following B+ tree with $d = 2$. For each part, execute the operation on the initial tree and answer the related question.

- All the numbers in the B+ Tree represent IDs.
- Each sub-question is independent. For each of the insert and delete operations, apply all the operations to the **initial B+ tree** and draw the final resulting tree.
- For grading purposes, assume a cell prefers to merge with its **RIGHT** sibling instead of the left sibling when applicable. When a cell splits, assume the middle key goes to the **LEFT** resulting cell instead of the right one.
- You can duplicate the B+ Tree draw.io file from here



1.1 (1 point) Please write down the blocks visited for the following query:

```
Select * from table where ID = 35
```

1.2 (1 point) Please write down the blocks visited for the following query:

```
Select * from table where ID = 27
```

1.3 (2 point) Please write down the blocks visited for the following query:

```
Select * from table where ID > 25 AND ID < 79
```

1.4 (1 point) Insert 35 into the tree, you must show all intermediate steps.

1.5 (1 point) Remove 90 from the tree, you must show all intermediate steps.

1.6 (4 point) Insert 28 into the tree, you must show all intermediate steps.

1.7 (3 point) Remove 75 from the tree, you must show all intermediate steps.

1.8 (7 point) Remove 75, 76, 80 sequentially from the tree, you must show all intermediate steps.

1.1 35 is smaller than 38

$\therefore B_1 \rightarrow B_2$

35 is greater than 30

$\therefore B_2 \rightarrow B_6$

$\therefore B_1 \rightarrow B_2 \rightarrow B_6$

1.2 : $\because 27 < 38$

$\therefore B_1 \rightarrow B_2$

$\because 27 > 24$

$\therefore B_2 \rightarrow B_5$

$\therefore B_1 \rightarrow B_2 \rightarrow B_5$

1.3 : Start from ZD > ZS

∵ 25 < 38

∴ $B_1 \to B_2$

∵ 25 > 24

∴ $B_2 \to B_5$

∵ 79 > 27

∴ $B_5 \to B_6$

∵ 79 > 73

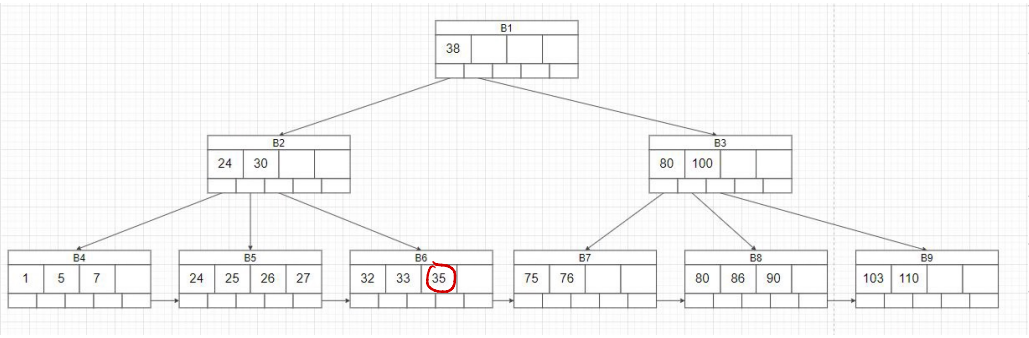
∴ $B_6 \to B_7$

∵ 80 > 79

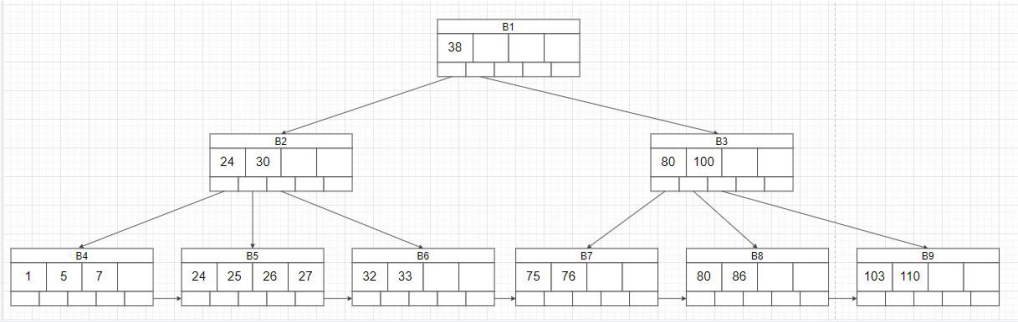
∴ the end

∴ $B_1 \to B_2 \to B_5 \to B_6 \to B_7$

1.4

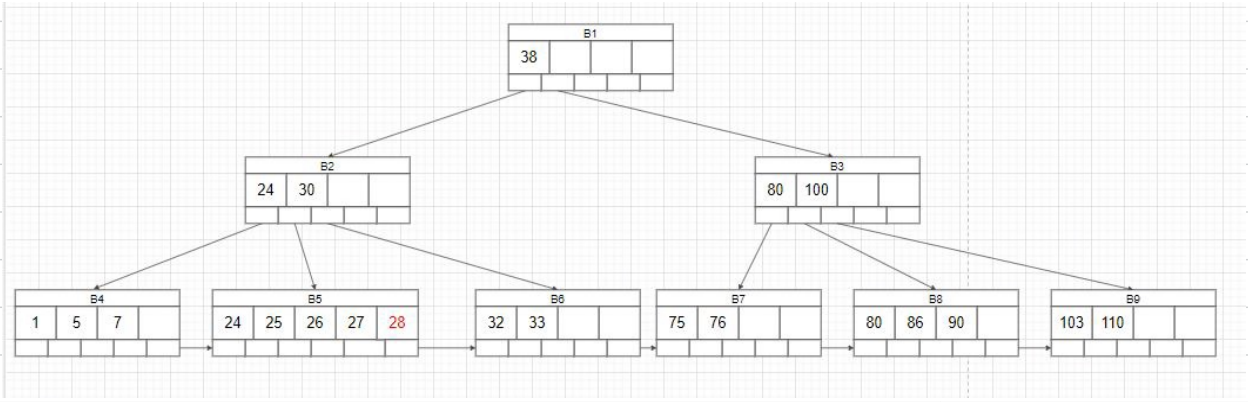

B1
38
B2
24
30
B3
80
100
B4
1
5
7
B5
24
25
26
27
B6
32
33
35
B7
75
76
B8
80
86
90
B9
103
110

1.5

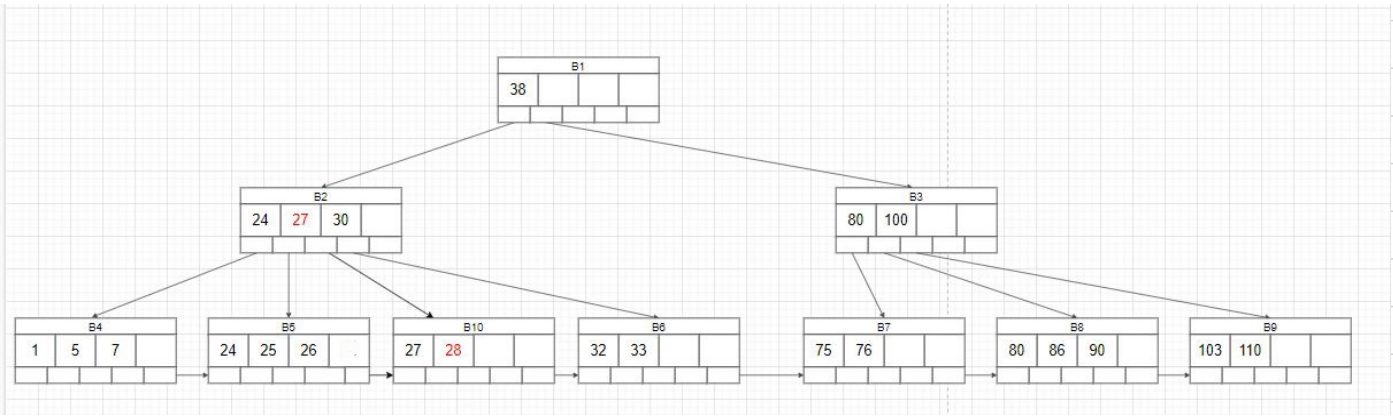

B1
38
B2
24
30
B3
80
100
B4
1
5
7
B5
24
25
26
27
B6
32
33
B7
75
76
B8
80
86
B9
103
110

1.b

step 1:

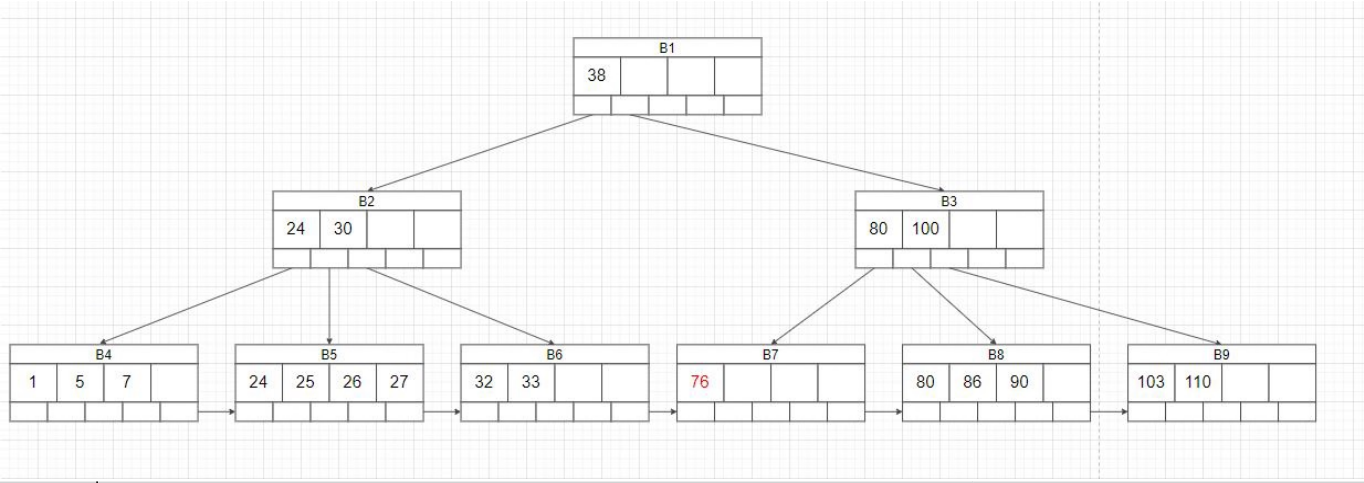


step2 :

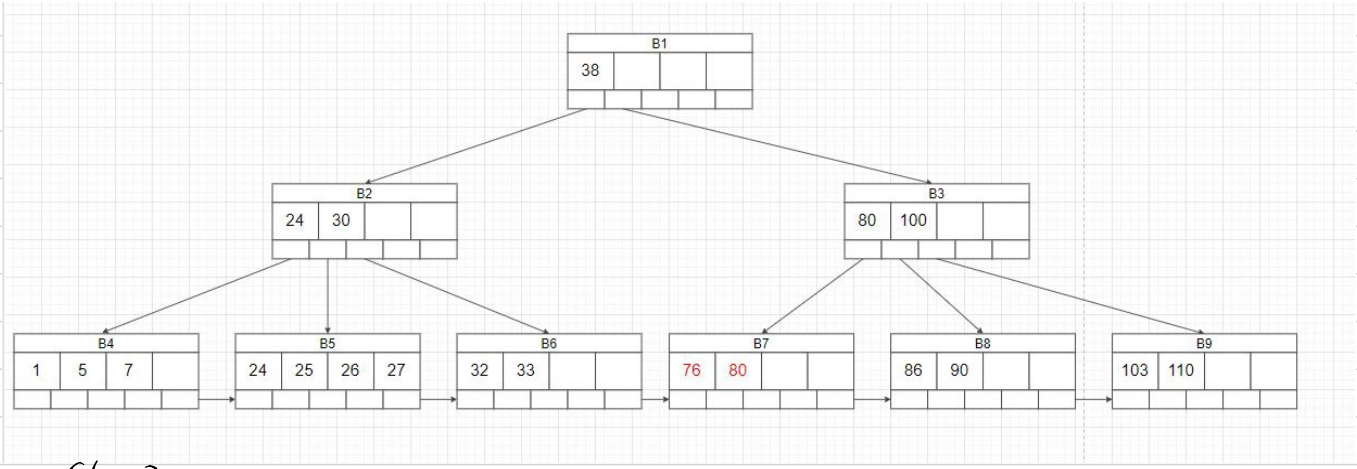


1.7

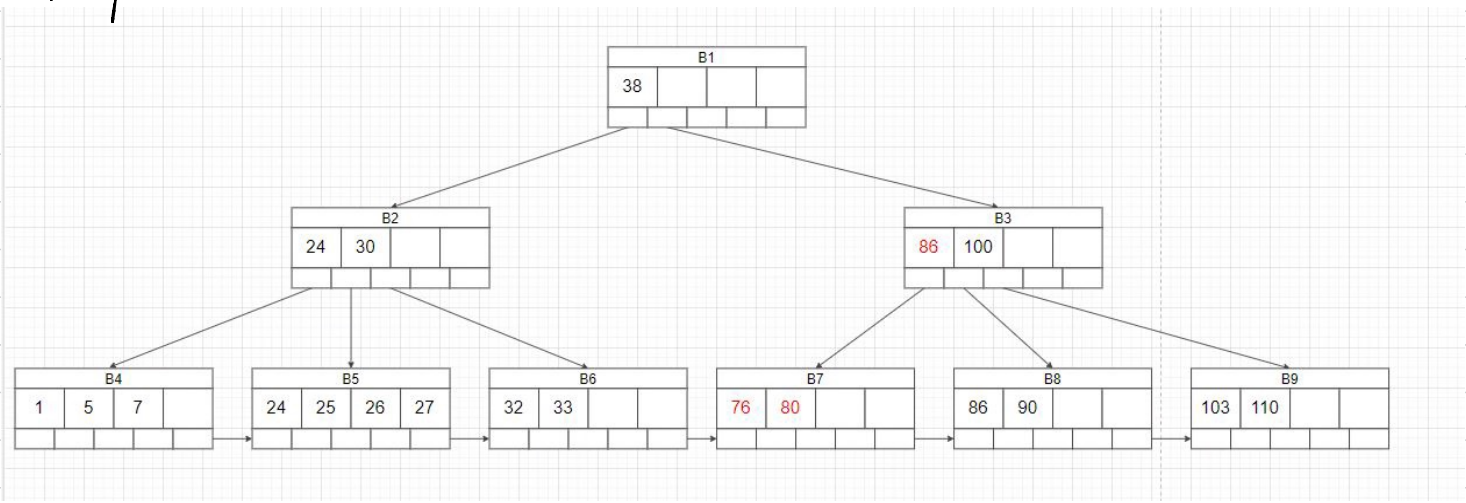
step 1:



step 2:

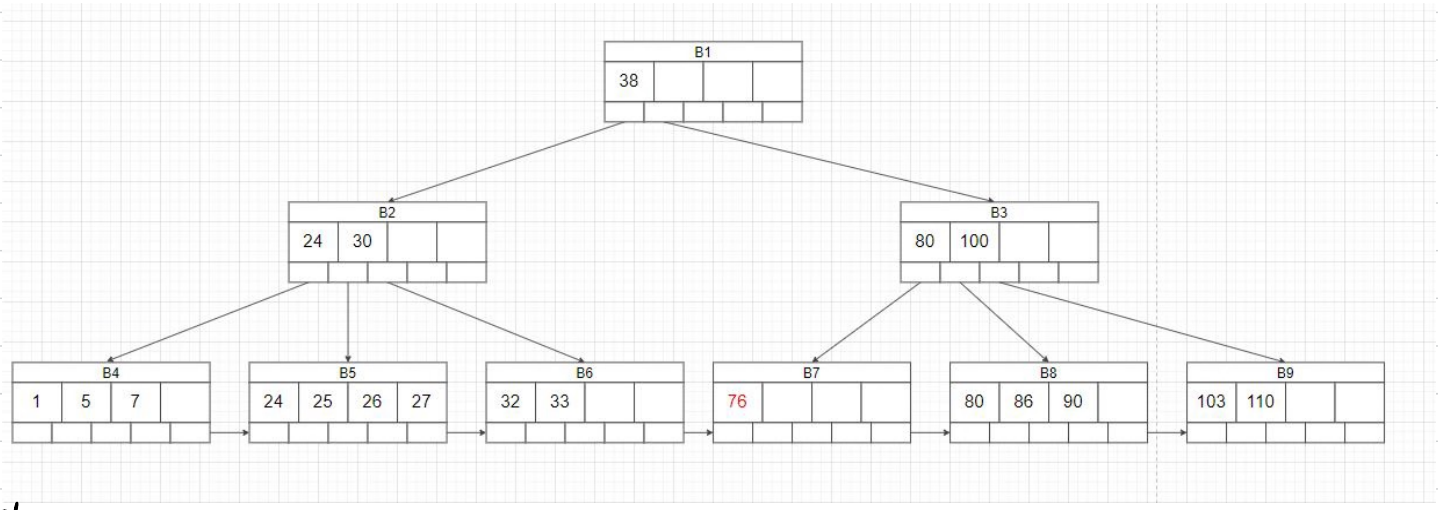


step3:

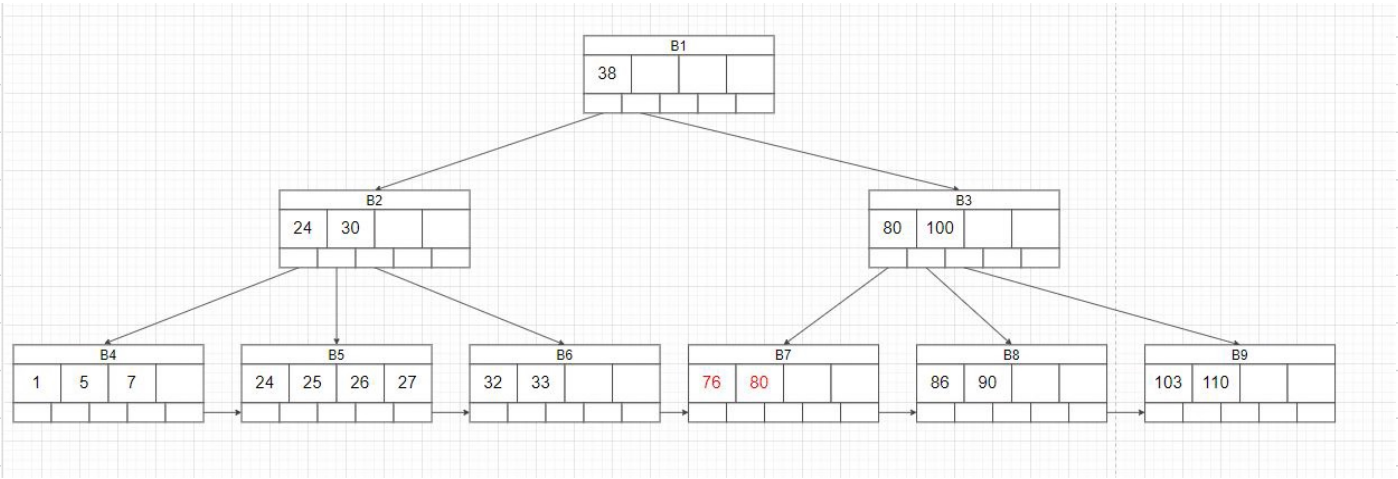


1.8 :

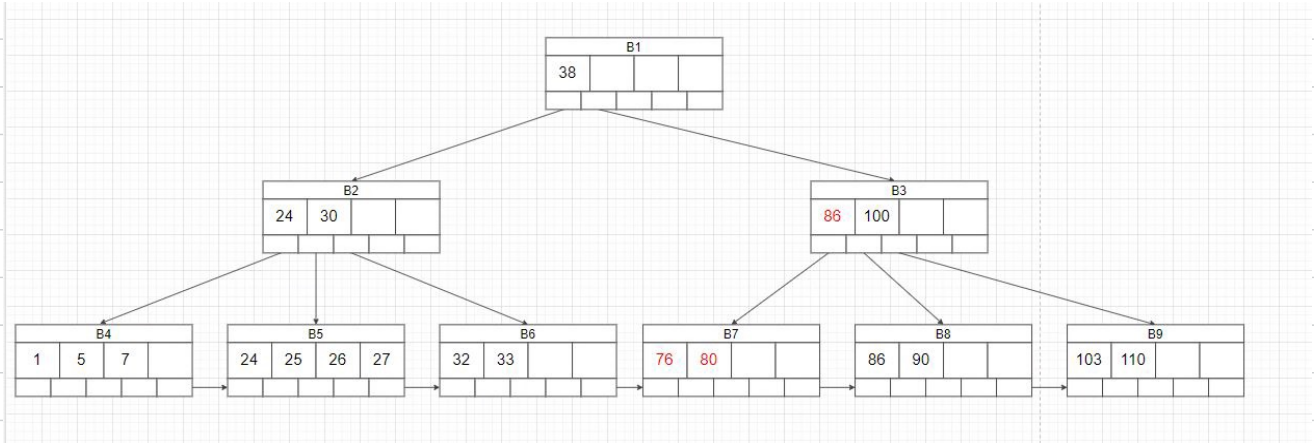
step 1 :



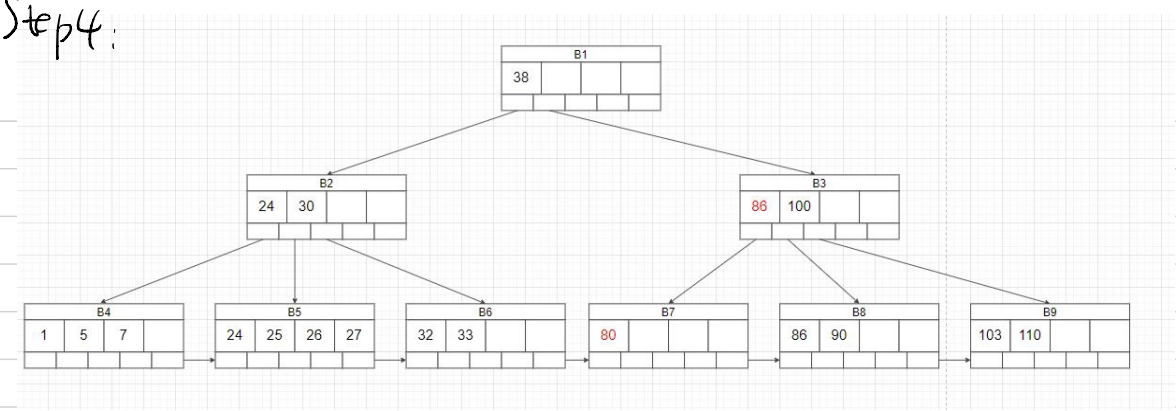
step2 :



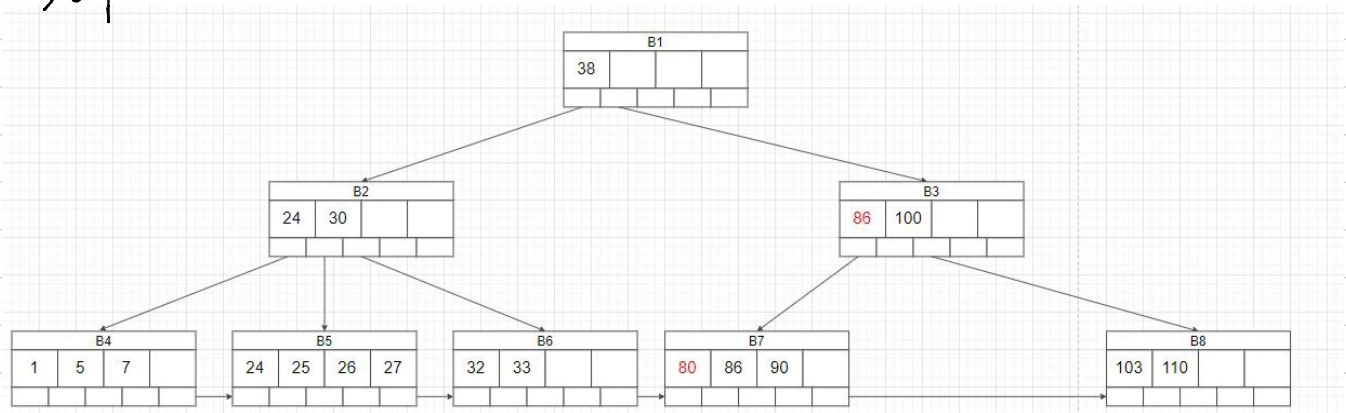
step3 :



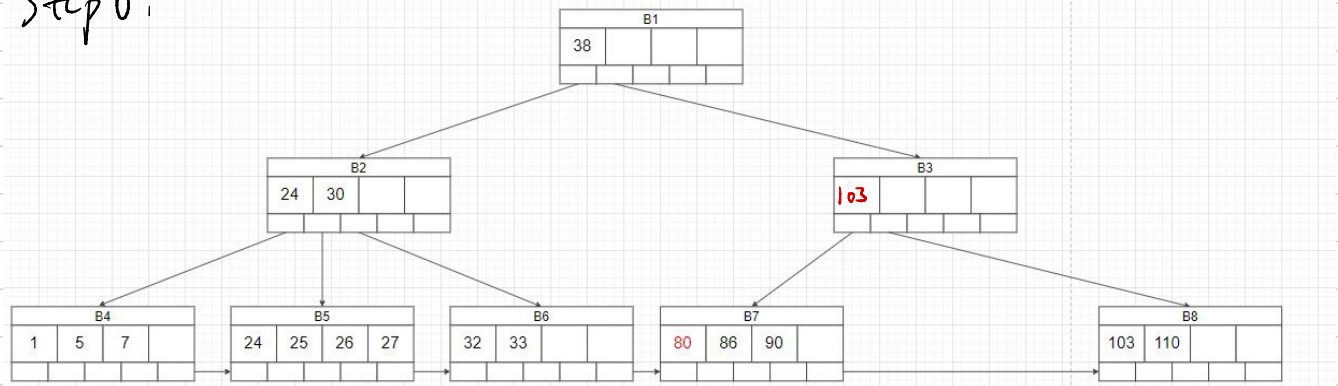
Step4:



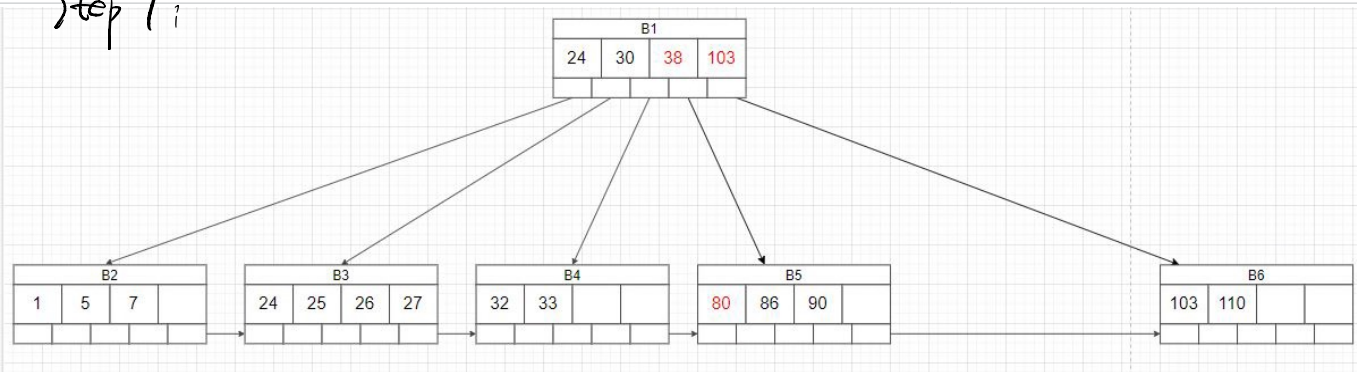
Step 5:



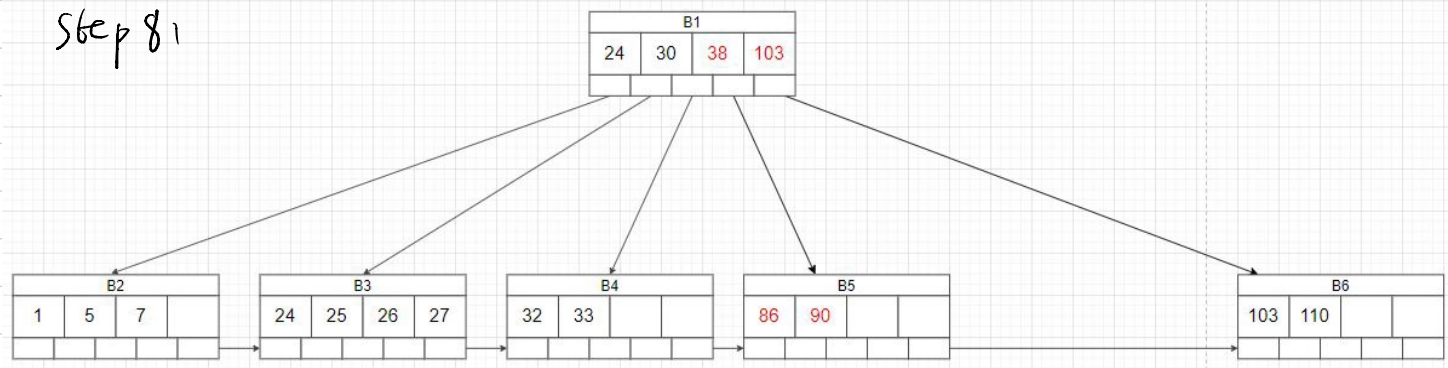
Step 6:



Step 7:



Step 8:

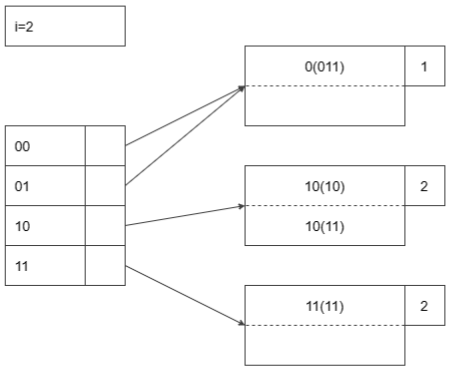


Problem 2 Hash tables

For this question, please complete the following questions:

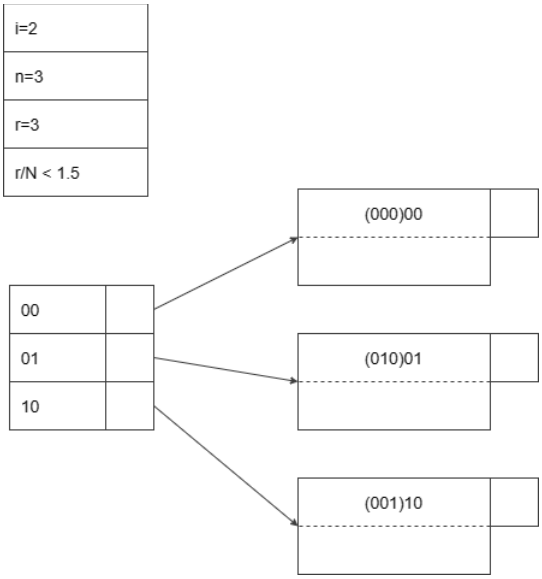
Q1. Extensible Hash Table (10 points)

Consider the following extensible hash table. Please insert the following values into the hash table in sequence: 0001, 0110, 0010, 1000. Please draw all intermediate steps.
You can duplicate the Hash Table draw.io file from here (select the Hash table tab)



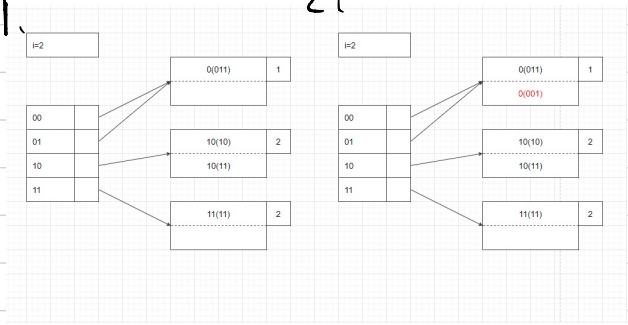
Q2. Linear Hash Table (10 points)

Consider the following Linear hash table. Please insert the following values into the hash table in sequence: 00011, 01101, 10001, 11101, 11111. Please draw all intermediate steps.
You can duplicate the Hash Table draw.io file from here (select the Hash table tab)



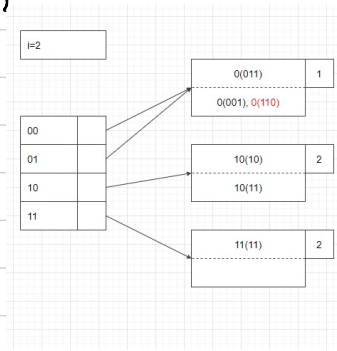
Q1:

1.

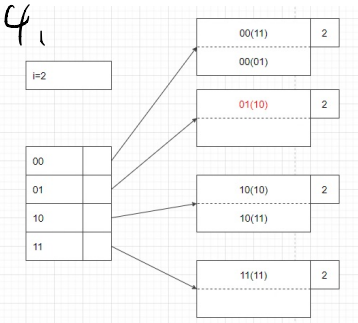


2.

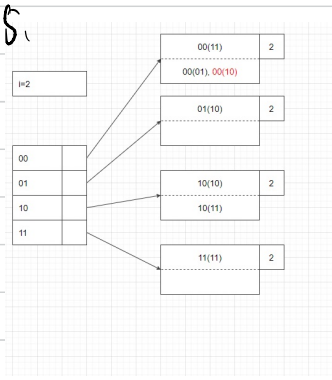
3.



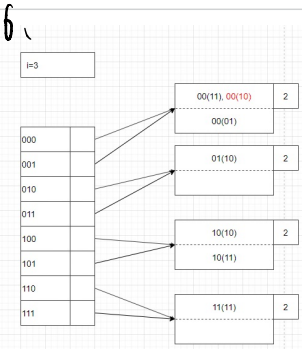
4.



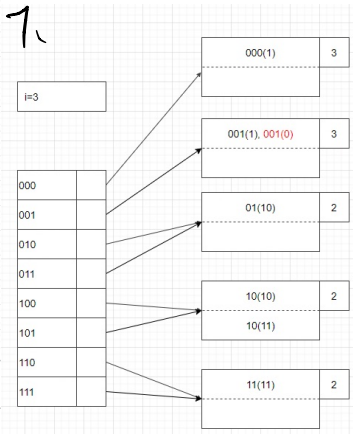
5.



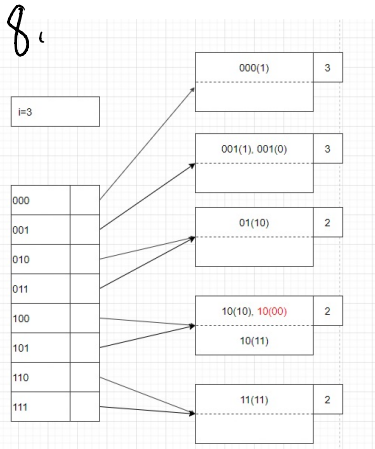
6.



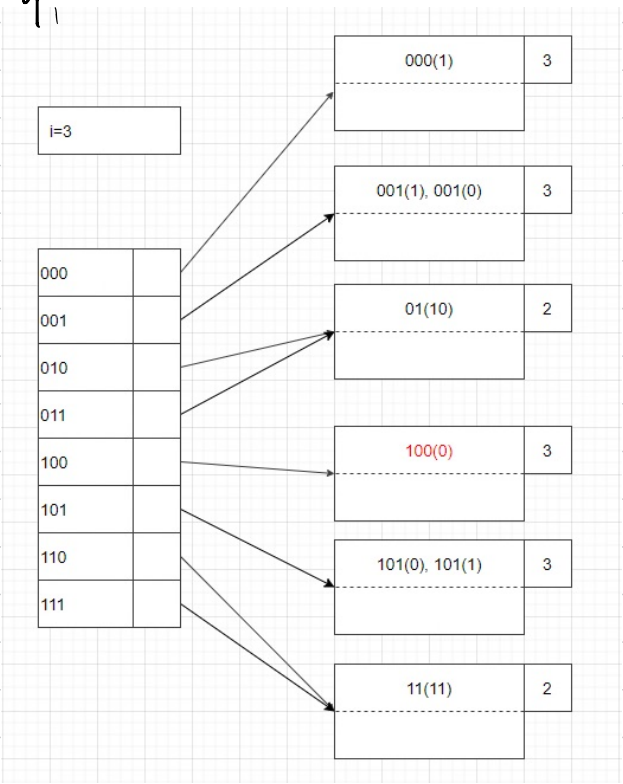
7.



8.

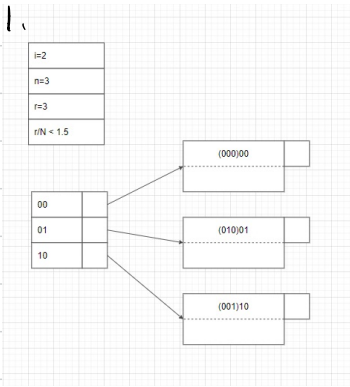


9.

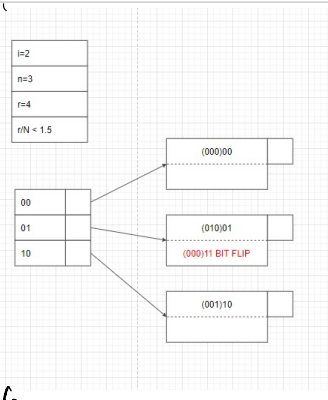


Q2

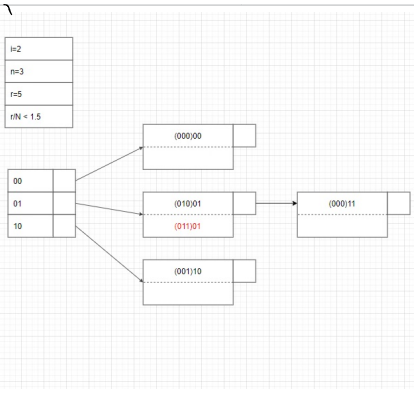
1.



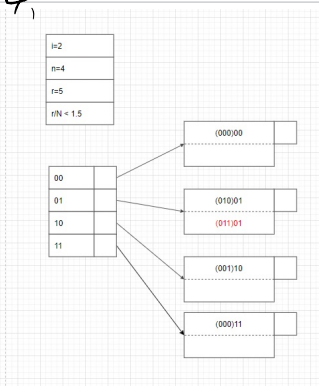
2.



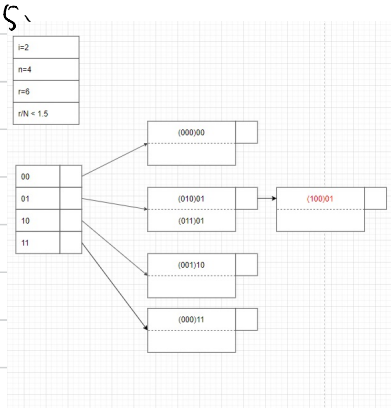
3.



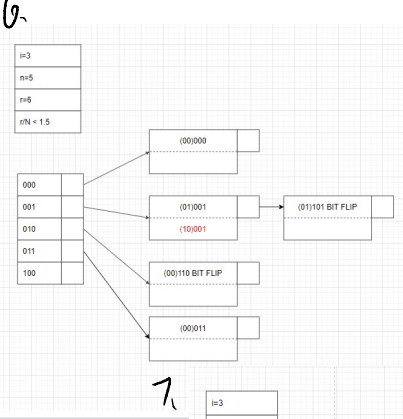
4.



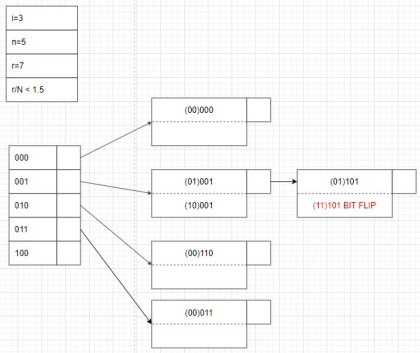
5.



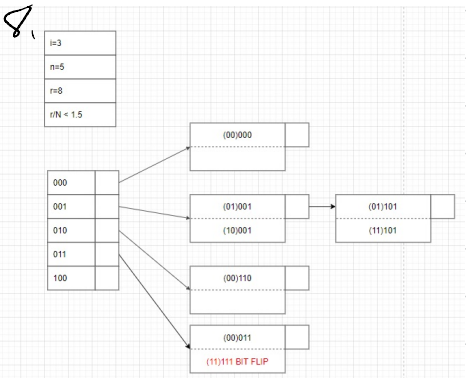
6.



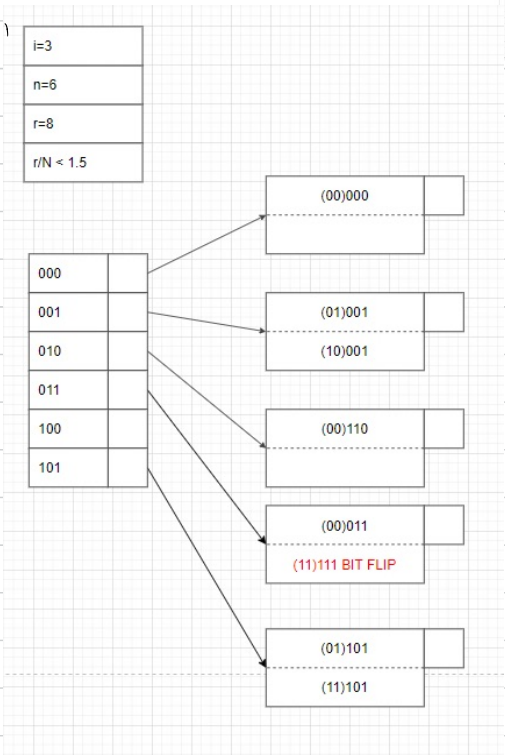
7.



8.



9.



Problem 3: Isolation Levels and Locking in Transactions (20 points)

Consider the following schedules S1 and S2 followed by transactions T1 and T2 on database objects A, B and C:

S1: R1(A), R1(B), W2(B), R2(C), R1(C), W2(A), W2(C), W1(B)
S2: R1(A), W2(B), R2(C), R1(B), W2(C), W2(A), R1(C), W1(B)

Determine whether the given schedules (S1 and S2) are feasible under the following isolation levels:

a. T1: Read Committed, T2: Repeatable Read (5×2=10 pts.)
b. T1: Repeatable Read, T2: Repeatable Read (5×2=10 pts.)

For each part (a and b), solve for both S1 and S2. For each part, explain your answer with a placement of locks to determine the feasibility (or infeasibility) under the assumed isolation levels. Note that the locks must be released as early as possible while respecting the isolation level.

Notation:

- Use SLOCKi(A) to indicate a shared lock on A by Transaction i.
- Use XLOCKi(A) to indicate an exclusive lock on A by Transaction i.
- Use RELi(A) to indicate the release of the lock on A by Transaction i.

Q3:

* I use $S_i(X)$ to represent SLOCKi(X)
  $X_i(X)$ to represent XLOCKi(X)

For S_1 : R1(A), R1(B), W2(B), R2(C), R1(C), W2(A), W2(C), W1(B)

in a's situation:

S1(A), R1(A), REL1(A), S1(B), R1(B), REL1(B), X2(B), W2(B), S2(C), R2(C), S1(C), R1(C), REL1(C), X2(A), W2(A), X2(C), W2(C), REL2(A,B,C), X1(B), W1(B), REL1(B)

No denied appears. So it's feasible.

in b's situation:

S1(A), R1(A), S1(B), R1(B), X2(B), DENIED2(B)

Since transaction 1 is still reading data B, while the share lock is not released. Transaction 2 can't add an exclusive lock on B.

So it's not feasible.

For S_2 : R1(A), W2(B), R2(C), R1(B), W2(C), W2(A), R1(C), W1(B)

in a's situation:

S1(A), R1(A), REL1(A), X2(B), W2(B), X2(C), R2(C), S1(B), DENIED1(B)

Since transaction 2 has already put an exclusive lock on B, so transaction 1 can't put a share block on B.

∴ it's not feasible.

in b's situation

S1(A), R1(A), X2(B), W2(B), X2(C), R2(C), S1(B), DENIED1(B)

It's similar to a's situation since B's exclusive lock makes transaction 1 can't add share lock.

Problem 4: Precedence Graph and Conflict-serializability (30 points)

Consider the following schedules S1, S2 and S3 followed by transactions T1, T2, T3, and T4 on database objects A, B and C:

SCHEDULE S1

S1: R1(B), W2(A), W1(C), R2(B), W3(C), W3(A), W4(B), W3(B), R4(A), W4(A)

T1	T2	T3	T4
R(B)			
	W(A)		
W(C)			
	R(B)		
		W(C)	
		W(A)	
			W(B)
		W(B)	
			R(A)
			W(A)

SCHEDULE S2

S2: R2(A), W2(B), R3(A), W4(A), R3(C), R2(B), R4(C), W1(C), R4(A), R1(C)

T1	T2	T3	T4
	R(A)		
	W(B)		
		R(A)	
			W(A)
		R(C)	
	R(B)		
			R(C)
W(C)			
			R(A)
R(C)			

SCHEDULE S3

S3: R4(C), R1(B), W1(C), R3(B), R4(A), R3(A), W2(B), R4(B), R2(C), R2(A)

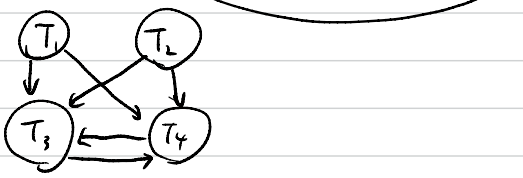
T1	T2	T3	T4
			R(C)
R(B)			
W(C)			
		R(B)	
			R(A)
		R(A)	
	W(B)		
			R(B)
	R(C)		
	R(A)		

a. Draw the precedence graph of S1, S2 and S3. (5×3=15 pts.)

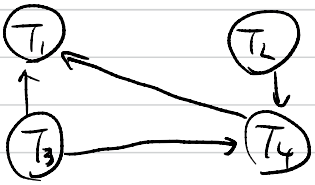
b. For schedules S1, S2, and S3, determine whether they are conflict-serializable or not. If they are, give the equivalent **serial schedule**. If not, briefly explain why. (5×3=15 pts.)

Q4 :

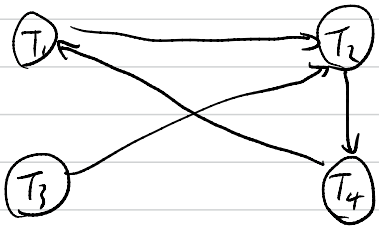
a. S_1 : S1: R1(B), W2(A), W1(C), R2(B), W3(C), W3(A), W4(B), W3(B), R4(A), W4(A)



S_2 : S2: R2(A), W2(B), R3(A), W4(A), R3(C), R2(B), R4(C), W1(C), R4(A), R1(C)



S_3 : S3: R4(C), R1(B), W1(C), R3(B), R4(A), R3(A), W2(B), R4(B), R2(C), R2(A)



Q4

b. for S_1, it's not conflict-serializable, there's a cycle between T_3 and T_4
for S_3, it's not conflict-serializable, there's a cycle between T_1, T_2 and T_4.

for S_2:

S2: R2(A), W2(B), R3(A), W4(A), R3(C), R2(B), R4(C), W1(C), R4(A), R1(C)

S2: R2(A), W2(B), R3(A), W4(A), R3(C), R2(B), R4(C), R4(A), W1(C), R1(C)

S2: R2(A), W2(B), R2(B), R3(A), W4(A), R3(C), R4(C), R4(A), W1(C), R1(C)

S2: R2(A), W2(B), R2(B), R3(A), R3(C), W4(A), R4(C), R4(A), W1(C), R1(C)

here, I swap $R_2(B)$ with $R_3(C)$, $W_4(A)$ $R_3(A)$ to move it forward

$T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow T_1$

Problem 5: Locking Protocols (2PL and Strict 2PL) (20 points)

Consider a set of schedules S1 and S2 by transactions T1, T2 and T3 on database objects A, B and C.

S1: R1(A), W3(C), R2(B), W1(A), R2(C), W1(B), W3(A), R1(B)
S2: W1(A), R3(B), W2(C), R3(B), W3(B), R1(C), R3(A), R2(C)

a. (10 pts.) Determine whether schedules S1 and S2 can be produced by a Two-Phase Lock (2PL) scheduler. For each, provide a table with placements of shared locks (SLOCK), exclusive locks (XLOCK) and lock releases (REL) that adhere to 2PL and lock-compatibility restrictions. (E.g. a shared lock by T1 on A is given by SLOCK1(A)). Each column must correspond to a single transaction with a single operation in each row. **In the case of infeasibility, mark the first point of the schedule where a transaction fails to get a required lock and you can stop there.** Note: Locks must be acquired only when necessary. You may assume actions are predictable within each transaction to determine lock releases.

b. (10 pts.) Determine whether schedules S1 and S2 can be produced by a STRICT Two-Phase Lock (2PL) scheduler. Similar to part (a), provide the corresponding lock placement table for each schedule. **If a schedule cannot be produced by Strict 2PL, enforce Strict 2PL to obtain a different schedule.** Reorder operations across transactions only when necessary; actions within each transaction should not be reordered. In case there is a deadlock (i.e. 2+ transactions depend on each other to release locks), you can stop there. Is the resulting schedule equivalent to the original schedule? Explain your answer.

Q5

a. S_1:

	T_1	T_2	T_3
	$S_1(A)$		
	$R_1(A)$		
			$X_3(C)$
			$W_3(C)$
		$S_2(B)$	
		$R_2(B)$	
	$X_1(A)$		
	$R_1(A)$		
		$S_2(C)$	

A	S1		X_1		X_3	
B		S_2		X_1		S_1
C	X_3		S_2			

$S1(A); R_1(A); X3(C); W3(C); S_2(B); R2(B); X1(A); W_1(A); S_2(C);$ DENIED2(C)

We can't release $X_3(C)$ here since transaction 3 still needs an exclusive lock later.

S_2:

A	X_1			S_3	
B		S_3	S_3	X_3	
C			X_2		S_1 S_2

$X_1(A); W_1(A); S_3(B); R_3(B); X_2(C); W_2(C); S_2(C); R_3(B); X_3(B); W_3(B); S_1(C); REL1(A,C)$
$S_3(A); R_3(A); REL3(A,B); R_2(C); REL2(C)$

T_1	T_2	T_3
$X_1(A)$		
$W_1(A)$		
		$S_3(B)$
		$R_3(B)$
	$X_2(C)$	
	$W_2(C)$	
	$S_2(C)$	
		$R_3(B)$
		$X_3(B)$
		$W_3(B)$
$S_1(C)$		
$REL1(A,C)$		
		$S_3(A)$
		$R_3(A)$
		$REL3(A,B)$
	$R_2(C)$	
	$REL2(C)$	

S_2 can be produced by 2PL.

b: S_1 : A S1 X_1 X_3

B S_2 X_1 S_1

C X_3 S_2

T_1	T_2	T_3
$S_1(A)$		
$R_1(A)$		
		$X_3(C)$
		$W_3(C)$
	$S_2(B)$	
	$R_2(B)$	
$X_1(A)$		
$W_1(A)$		
	$S_2(C)$ <Denied, wait>	
$X_1(B)$ <Denied, wait>		

We have a deadlock here since transaction 1 and 2 depend on each other to release locks.

S_2 :

A X_1 S_3

B S_3 S_3 X_3

C X_2 S_1 S_2

T_1	T_2	T_3
$X_1(A)$		
$W_1(A)$		
		$S_3(B)$
		$R_3(B)$
	$X_2(C)$	
	$W_2(C)$	
		$R_3(B)$
		$X_3(B)$
		$W_3(B)$
$S_1(C)$ <Denied, wait>		
		$S_3(A)$ <Denied, wait>
	$R_2(C)$	
	$REL_2(C)$	
$S_1(C)$ <Granted>		
$R_1(C)$		
$REL_1(A,C)$		
		$S_3(A)$ <Granted>
		$R_3(A)$
		$REL_3(A,B)$

It's like the schedule with swap operations

It's equivalent to the original schedule.

It's like:

S: $X_1(A), W_1(A), S_3(B), R_3(B), X_2(C), W_2(C)$ $R_3(B), X_3(B), W_3(B), R_2(C), REL_2(C), S_1(C)$ $R_1(C), REL_1(A,C), S_3(A), R_3(A), REL_3(A,B)$