Task#1 (Recursive Query)
CLO 2.6

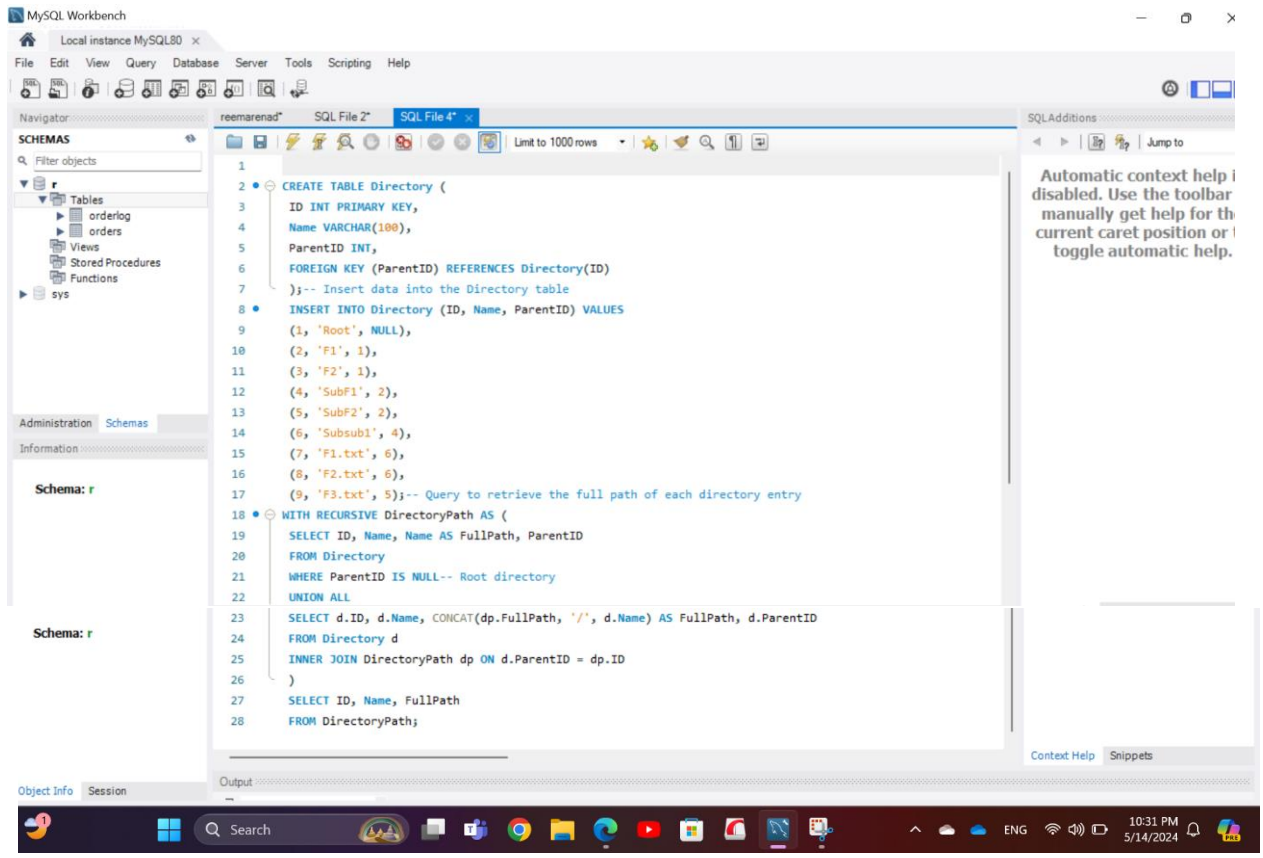
Q 1: Consider a table named Directory with the following columns:

ID (integer, primary key): Unique identifier for each directory entry.
Name (text): Name of the directory entry.
ParentID (integer, foreign key): ID of the parent directory. Null if the entry is at the root level.

Write a SQL query to retrieve the full path of each directory entry in the Directory table. The output should include the ID, Name, and FullPath columns, where FullPath represents the complete path of each directory entry, starting from the root directory and separated by slashes ("/").

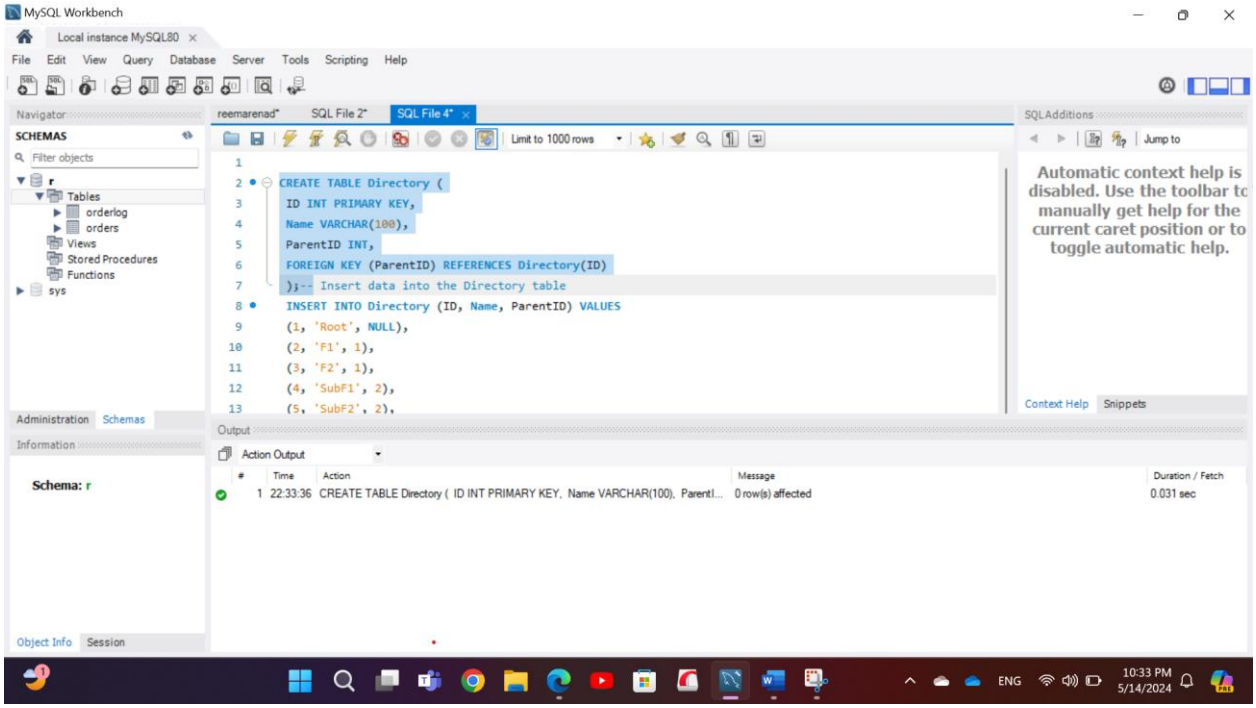
[3 Marks]

The code:

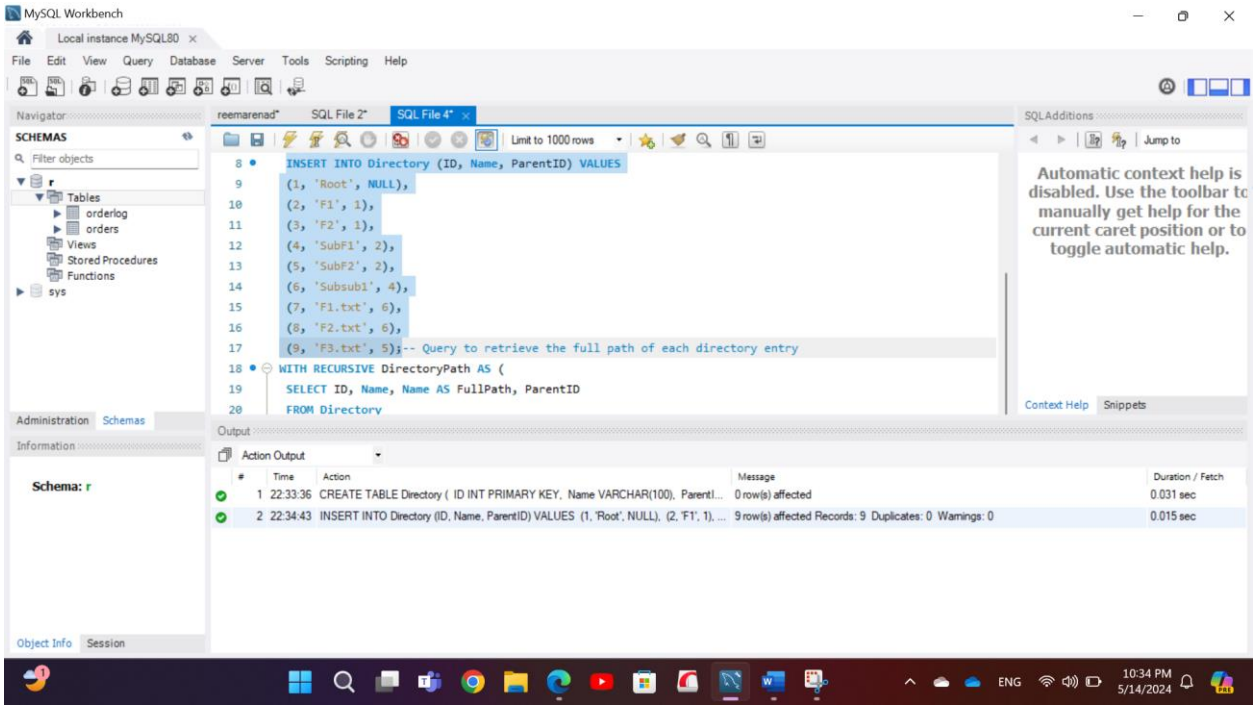


The execution:

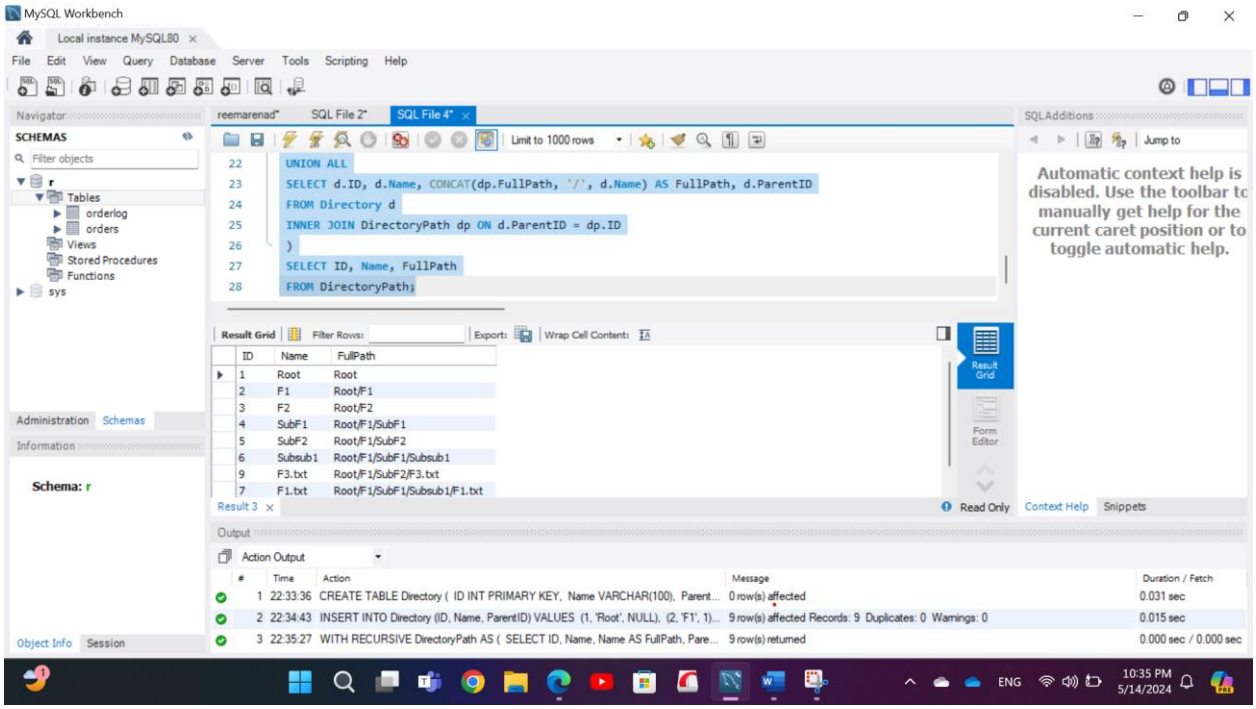
1:



2:



Output:


MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
r
Tables
orderlog
orders
Views
Stored Procedures
Functions
sys
Administration Schemas
Information
Schema: r
Object Info Session
reemarenad* SQL File 2* SQL File 4*
Limit to 1000 rows
22 UNION ALL
23 SELECT d.ID, d.Name, CONCAT(dp.FullPath, '/', d.Name) AS FullPath, d.ParentID
24 FROM Directory d
25 INNER JOIN DirectoryPath dp ON d.ParentID = dp.ID
26)
27 SELECT ID, Name, FullPath
28 FROM DirectoryPath;
Result Grid Filter Rows: Export: Wrap Cell Content:
ID Name FullPath
1 Root Root
2 F1 Root/F1
3 F2 Root/F2
4 SubF1 Root/F1/SubF1
5 SubF2 Root/F1/SubF2
6 Subsub1 Root/F1/SubF1/Subsub1
9 F3.txt Root/F1/SubF2/F3.txt
7 F1.txt Root/F1/SubF1/Subsub1/F1.txt
Result 3
Read Only
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 22:33:36 CREATE TABLE Directory (ID INT PRIMARY KEY, Name VARCHAR(100), Parent... 0 row(s) affected 0.031 sec
2 22:34:43 INSERT INTO Directory (ID, Name, ParentID) VALUES (1, 'Root', NULL), (2, 'F1', 1)... 9 row(s) affected Records: 9 Duplicates: 0 Warnings: 0 0.015 sec
3 22:35:27 WITH RECURSIVE DirectoryPath AS (SELECT ID, Name, Name AS FullPath, Pare... 9 row(s) returned 0.000 sec / 0.000 sec
ENG
10:35 PM
5/14/2024

Task#2 (B+ Tree Indexing)
CLO 2.2

Q.2: Consider the B+ tree shown in Figure 1, do the following:

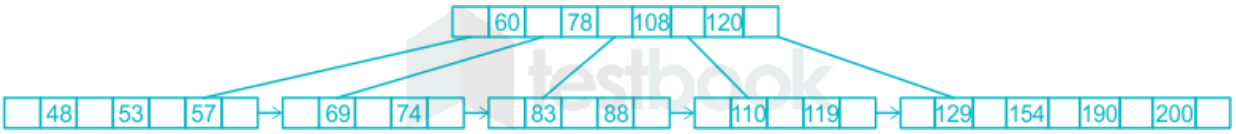
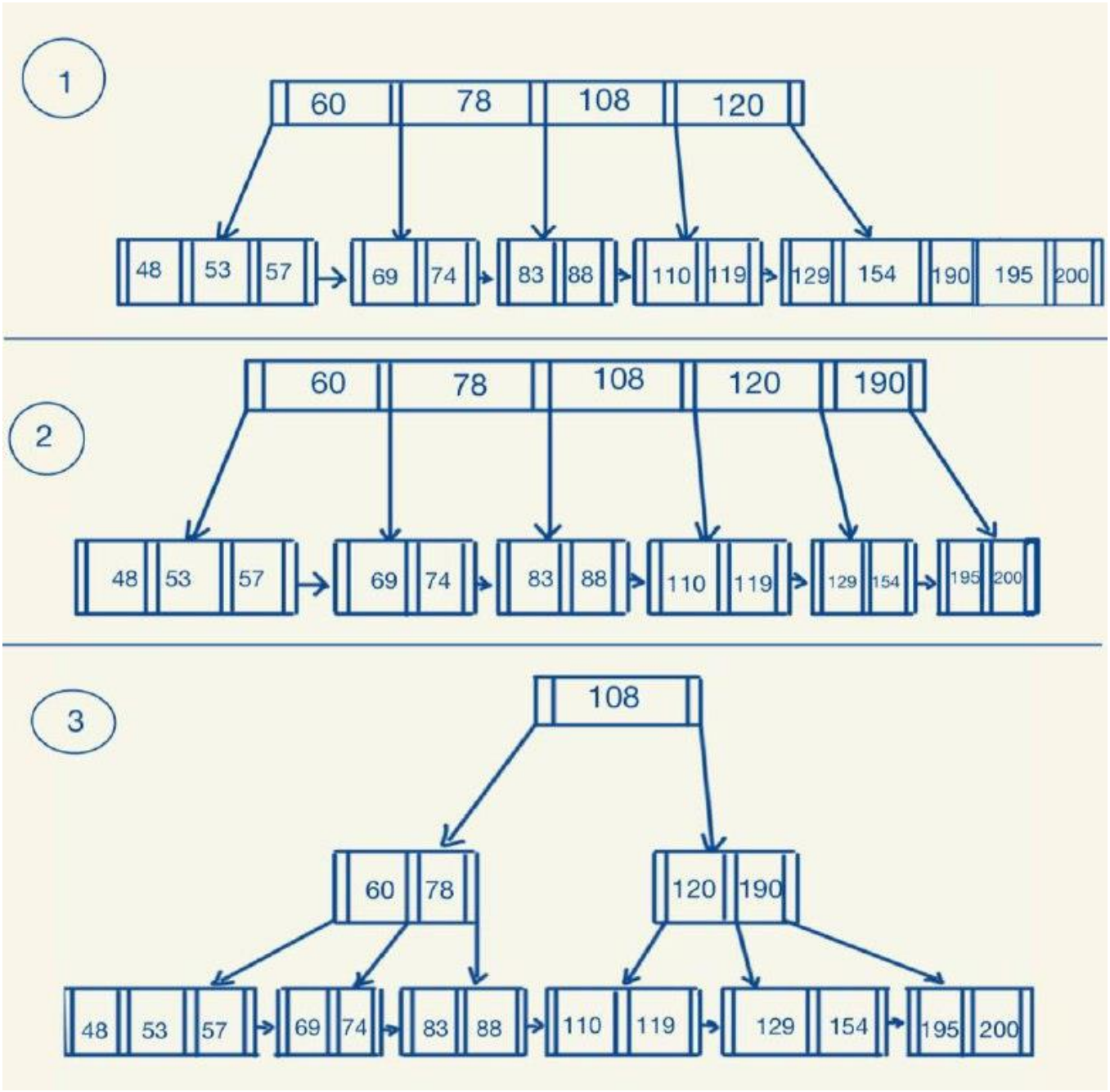
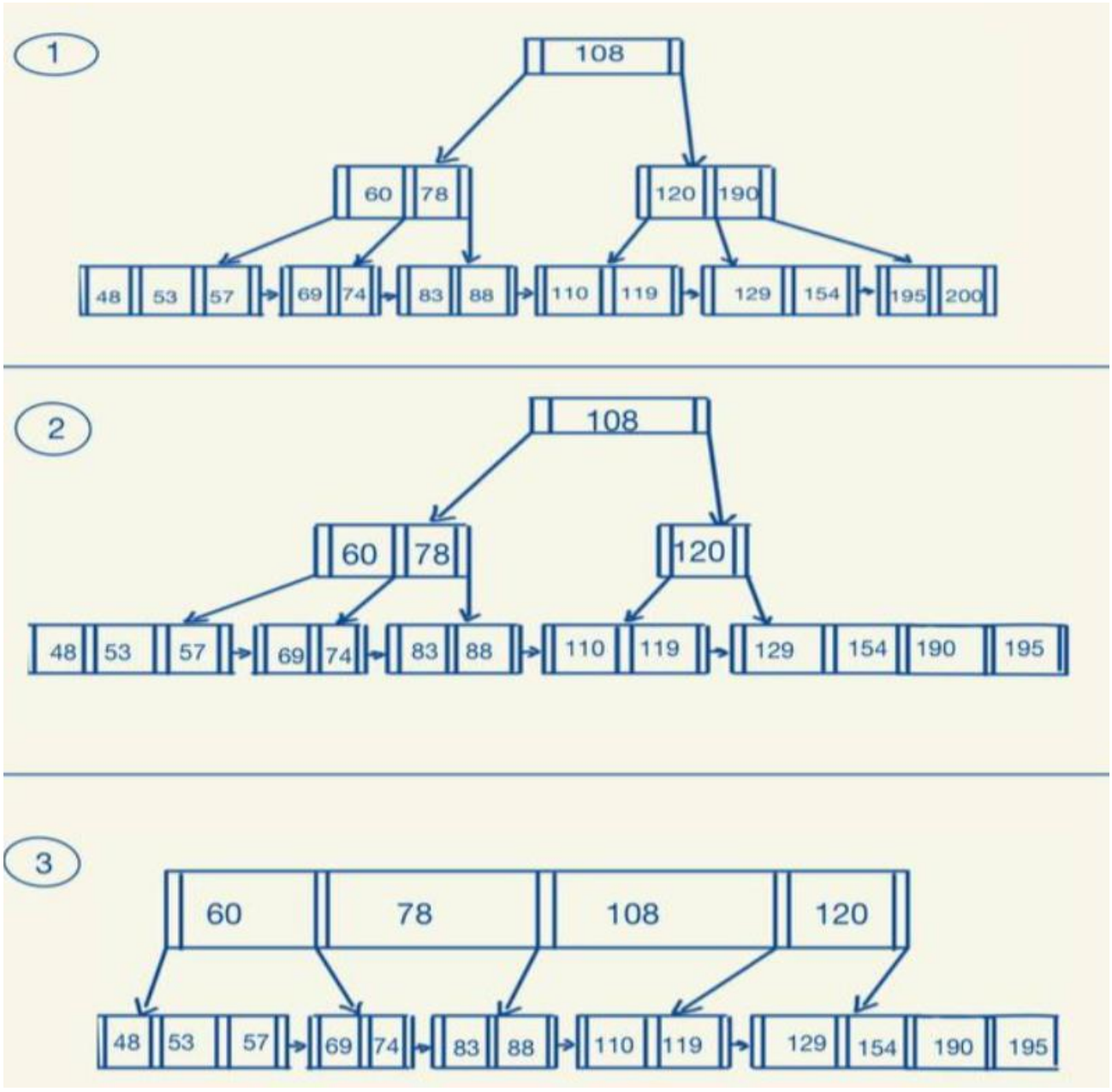


Figure 1: A B+ Tree

a) Add key 195, and update the B+ tree. **[2 Marks]**



b) Delete key 200 from the tree obtained from Q1 (a) **[3 Marks]**



Task#3 (Transactions and Schedules in Databases)

CLO 2.4

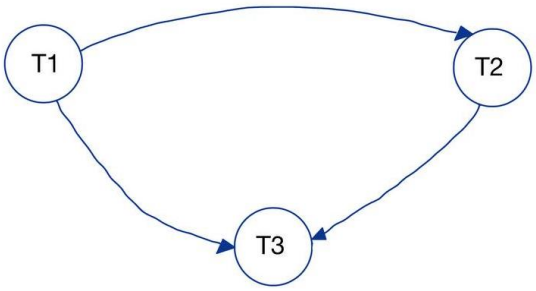
Q.3(a): You are given the following schedules involving three transactions (T1, T2, T3) performing operations on database items (A, B, C, D): **[2 Marks]**

Schedule S1:

T1: R(A), R(B), W(C)
T2: W(A), R(B), R(C)
T3: W(A), W(B), R(D)

T1 → T2: Conflict 5
T1 → T3: Conflict R(A) in T1, W(A) T3
T2 → T3: Conflict 7

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



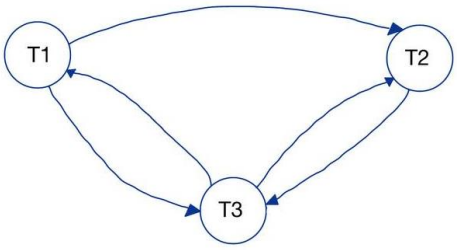
T1 R(A) → T2 W(A)
T1 R(A) → T3 W(A)
T1 R(B) → T3 W(B)
T1 W(C) → T2 R(C)
T2 W(A) → T3 W(A)
T2 R(B) → T3 W(B)

S1: Conflict-serializable because there is no cycle.

Schedule S2:

T1: R(A), R(C), W(C), R(D)
T2: R(A), W(B), R(C), W(D)
T3: W(B), W(A), W(D), R(C)

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



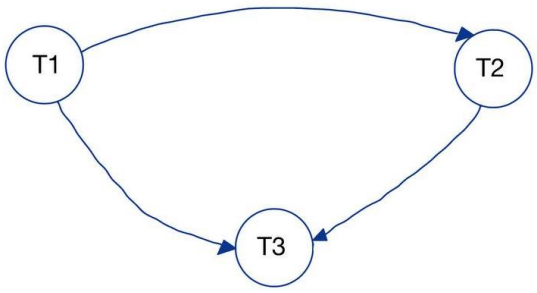
T1 R(A) → T3 W(A)
T1 W(C) → T2 R(C)
T1 R(D) → T2 W(D)
T2 R(A) → T3 W(A)
T3 W(B) → T2 W(B)
T3 W(D) → T2 W(D)
T3 W(D) → T1 R(D)

S2: Not conflict-serializable because there is a cycle.

Schedule S3:

T1: R(A), R(B), R(C), W(D)
T2: W(A), W(B), W(C), R(D)
T3: W(A), R(B), W(C), W(D)

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



T1 R(A) → T2 W(A)
T1 R(A) → T3 W(A)
T1 R(B) → T2 W(B)
T1 R(C) → T2 W(C)
T1 R(C) → T3 W(C)
T1 W(D) → T2 R(D)
T1 W(D) → T3 W(D)

S3: Conflict-serializable because there is no cycle.

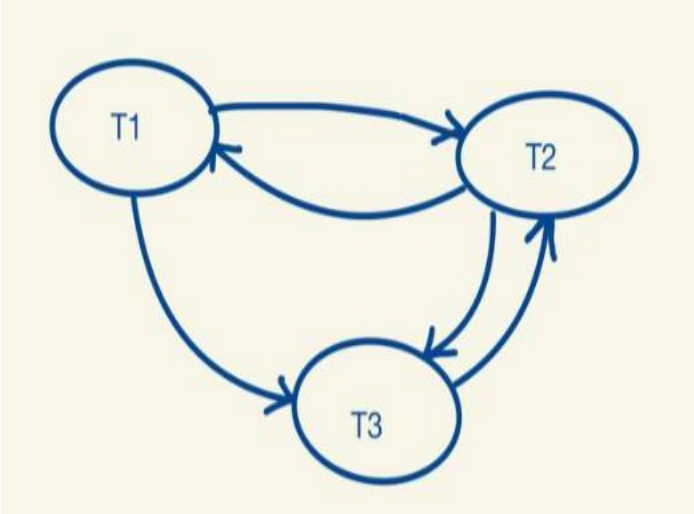
Your task is to verify the conflict serializability of each schedule and determine whether they are conflict-serializable or not. If it's not, explain the conflicting operations that violate conflict-serializability.

Q.3(b): Consider the following two schedules; Schedule A and Schedule B, as given below:
[2 Marks]

Schedule A:

T1: R1(X); W2(Y); W1(Y); R3(Z); R1(Y); W2(X); W2(Z); W3(X)
T2: R1(Y); W1(X); R2(Y); R2(X)
T3: W1(Z); W3(Y); R3(X); R2(Z); R1(Z)

T1	T2	T3
R(X)		
	W(Y)	
W(Y)		
		R(Z)
R(Y)		
	W(X)	
	W(Z)	
		W(X)
R(Y)		
W(X)		
	R(Y)	
	R(X)	
W(Z)		
		W(Y)
		R(X)
	R(Z)	
R(Z)		

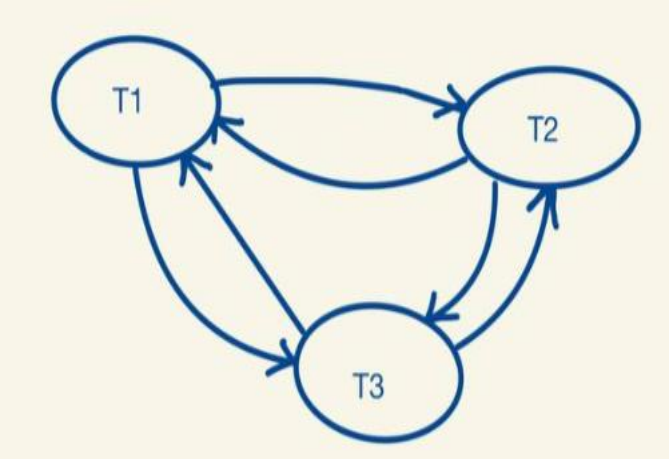


Schedule "A" is not conflict-serializable.

Schedule B:

T1: R1(X); R2(Y); W1(X); R2(Z); W2(X); W3(Z); R3(X)
T2: R1(Y); R3(Z); W1(Y); R2(X); R2(Z)
T3: R1(Z); W2(Y); R3(X); W1(Z)

T1	T2	T3
R(X)		
	R(Y)	
W(X)		
	R(Z)	
	W(X)	
		W(Z)
		R(X)
R(Y)		
		R(Z)
W(Y)		
	R(X)	
	R(Z)	
R(Z)		
	W(Y)	
		R(X)
W(Z)		



Schedule "B" is not conflict-serializable.

Draw the precedence graph for Schedule A and Schedule B. State whether Schedules are conflict-serializable or not, and explain your reasoning.

Q.3(c): Consider the following two transactions, T1 and T2, involved in a multi-user database environment:

Transaction#1

```
read(X);
read(Y);
if X = 0 then Y := Y + 1;
write(Y).
```

Answer Transaction#1:
lock-S(X)
read (X)
lock-X(Y)
read (Y)
if X = 0 then Y := Y + 1
write(Y)
unlock(X)
unlock(Y)

Tranascation#2

read(Y);
read(X);
if Y = 0 then X := X + 1;
write(X).

Answer Tranascation#2:
lock-S(Y)
read(Y)
lock-X(X)
read (X)
if Y = 0 then X := X+1
write(X)
unlock(Y)
unlock(X)

Modify transactions T1 and T2 by adding appropriate lock and unlock instructions to ensure they adhere to the two-phase locking protocol.