Qtext:-

Use Linear Hashing technique to store the following customer data with Cid as Pk . Assume Linear Hashing and hash function h(k)= k mod 10. **[5]**

Customer:

CID	STATE	ZIP CODE	COUNTRY
14324742	CA	95123	USA
14569877	NC	27514	USA
14898029	MD	20850	USA
14324742	CA	95123	USA
14569877	NC	27514	USA
14898029	MD	20850	USA
19876078	CA	93274	USA

(A) For SQL Query ? Delete from customer where cid = 23328383? and Redraw the Hash. **[2.5]**

(B) For SQL Query ?Insert into student (cid, state , zipcode, country) values (54440383, AZ, 33332, NZ)?. Redraw Hash. **[2.5]**

Qtext:-

Consider the following classes of schedules: serializable, conflict-serializable, view-serializable, recoverable, avoids-cascading-aborts, and strict. For each of the following schedules, state which of the above classes it belongs to. If you cannot decide whether a schedule belongs in a certain class based on the listed actions, explain briefly. The actions are listed in the order they are scheduled, and prefixed with the transaction name. If a commit or abort is not shown, the schedule is incomplete; assume that abort/commit must follow all the listed actions:

(A) T1:R(X), T1:R(Y), T1:W(X), T2:R(Y), T3:W(Y), T1:W(X), T2:R(Y) **[1]**
(B) T1:R(X), T2:W(X), T1:W(X), T2:Abort, T1:Commit **[1]**
(C) T1:W(X), T2:R(X), T1:W(X), T2:Abort, T1:Commit **[1]**
(D) T1:W(X), T2:R(X), T1:W(X), T2:Commit, T1:Commit **[1]**
(E) T1:W(X), T2:R(X), T1:W(X), T2:Commit, T1:Abort **[1]**
(F) T2: R(X), T3:W(X), T3:Commit, T1:W(Y), T1:Commit, T2:R(Y), T2:W(Z), T2:Commit **[2]**
(G) R(X), T2:W(X), T2:Commit, T1:W(X), T1:Commit, T3:R(X), T3:Commit **[1]**
(H) T1:R(X), T2:W(X), T1:W(X), T3:R(X), T1:Commit, T2:Commit, T3:Commit **[1]**

Qtext:-

(a) Consider the following code what happens if the two Transactions run concurrently **[3]**

T1

```
BEGIN TRANSACTION
UPDATE works_on
   SET job = 'Manager'
   WHERE emp_no = 18316
   AND project_no = 'p2'
WAITFOR DELAY '00:00:10'
UPDATE employee
   SET emp_lname = 'Green'
   WHERE emp_no = 9031
COMMIT
```

T2

```
BEGIN TRANSACTION
UPDATE employee
   SET dept_no = 'd2'
   WHERE emp_no = 9031
WAITFOR DELAY '00:00:10'
DELETE FROM works_on
  WHERE emp_no = 18316
  AND project_no = 'p2'
COMMIT
```

(b) Assume that a transaction T1 would do a read and write on certain data items like d1,d2,..,dk. Write down the pseudo code or algorithm for T1 as to how to acquire the locks so that serializability and deadlock-freedom is achieved. **[3]**

(c) Draw ?wait for graph? for the following and determine if it would lead to deadlock or not? **[3]**

T1	T2
Lock_X(P) Read(P) P=P-100 Write(P) Lock_X(Q) Read(Q) Q=Q+100 Write(Q) UnLock(P) UnLock(Q)	Lock_S(Q) Read(Q) Lock_S(P) Read(P) Display(P+Q) UnLock(Q) UnLock(P)

Qtext:-

Suppose that all the relations were created by (and hence are owned by) user X, who wants to grant the following privileges to user accounts A, B, C, D, and E:

(A) Account A can retrieve or modify any relation except DEPENDENT and can grant any of these privileges to other users. **[1]**

(B) Account B can retrieve all the attributes of EMPLOYEE and DEPARTMENT except for Salary, Mgr_ssn, and Mgr_start_date. **[1]**

(C) Account C can retrieve or modify WORKS_ON but can only retrieve the Fname, Minit, Lname, and Ssn attributes of EMPLOYEE and the Pname and Pnumber attributes of PROJECT. **[2]**

(D) Account D can retrieve any attribute of EMPLOYEE or DEPENDENT and can modify DEPENDENT. **[1]**

(E) Account E can retrieve any attribute of EMPLOYEE but only for EMPLOYEE tuples that have Dno = 3**[1]**

(F) Write SQL statements to grant these privileges. Use views where appropriate. **[1]**

Qtext:-

Compare the cost of two different query plans for the following query: [10]

$$\pi_{sname}\left(\pi_{sid}\left(\left(\pi_{bid}\sigma_{color='red'}Boats\right)\bowtie Reserves\right)\bowtie Sailors\right)$$