1) 20 pts 2pts/ea.
Indicate whether each of the following statements is true or false (T/F):

F Views can only be used to read data from the database but not write to it

F ER diagram is not useful when designing an object relational database.

T Candidate key is a minimal superkey that uniquely identifies an entity.

F If we compare two null values using <, >, =, and so on, the result is always true.

F Triggers are useful tools to maintain data integrity

F Any ternary relationship can be reduced to two or three binary relationships

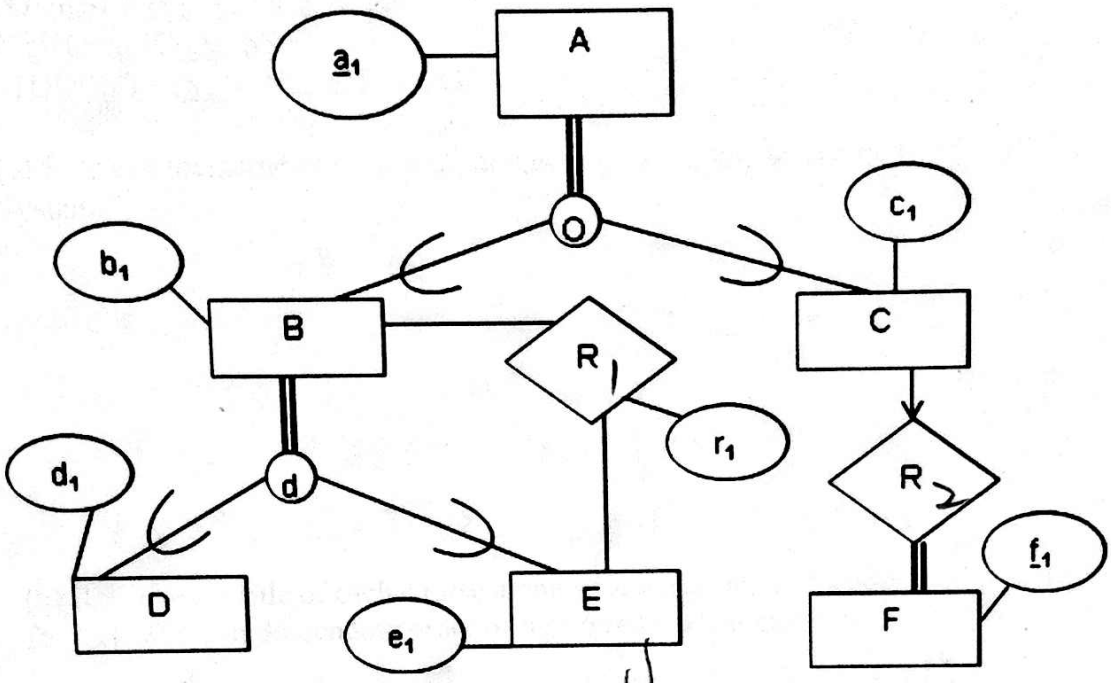
T Stored procedures can provide logical data independence

F JDBC drivers are DBMS independent

F View is a mechanism that provides support for physical data independence

T Code written in SQLJ is platform independent

2) 15 pts
Reduce the given EER diagram to relations using pure relational model (i.e., No Object Oriented or Object Relational). Be sure to identify all integrity constraints.



A($\underline{a_1}$)

B($\underline{a_1}$, b_1) a_1 FK to A

C($\underline{a_1}$, c_1) a_1 FK to A

D($\underline{a_1}$, d_1) a_1 FK to B

E($\underline{a_1}$, e_1) a_1 FK to B

F($\underline{f_1}$, a_1) a_1 FK to C

R_1($\underline{a_1}$, $\underline{a'_1}$, r_1) a_1 FK to B, a'_1 FK to E

assertion:
$B.a_1 \cup C.a_1 = A.a_1$ (1 Pt)

assertions:
$D.a_1 \cup E.a_1 = B.a_1$
$D.a_1 \cap E.a_1 = \phi$ (2 pts)

a_1 NOT NULL (1 Pt)

5 pts for tables

5 pts for FKs

3 pts: idea
2 pts: syntax
5 pts/ea. multiple sol. possible.

3) 15 pts

Consider the ~~relational~~ conceptual database schema below for keeping track of course registration of students:

COURSES (Code, Title, Dept)
Registered (Code, SSN)
STUDENTS (SSN, Name, Dept, GPA)

(a) Retrieve the name of each student who registered for the course titled "Database Systems".

```
SELECT DISTINCT S.Name
FROM STUDENTS S, Registered R, COURSES C
WHERE C.Code = R.code
AND   S.SSN = R.SSN
AND   C.Title = 'Database Systems';
```

(b) Retrieve the title of each course along with the number of students who registered for this course in descending order of registered student numbers.

```
SELECT C.title, COUNT(R.SSN)
FROM COURSES C, Registered R
WHERE C.code = R.code
GROUP BY C.code, C.title
ORDER BY COUNT(R.SSN) DESC;
```

(c) Retrieve the name of student(s) who have earned maximum GPA in every department that provides more than 30 courses.

```
SELECT S.Name
FROM STUDENTS S
WHERE S.Dept IN ( SELECT C.Dept
                  FROM COURSES C
                  GROUP BY C.Dept
                  HAVING COUNT(C.Code) > 30)
AND S.GPA >= ALL ( SELECT S2.GPA
                   FROM STUDENTS S2
                   WHERE S2.Dept = S.Dept)
```

4) 15 pts

Consider the following relational schema:

Emp (*eid:* integer, *ename:* string, *age:* integer, *salary:* real)
Works (*eid:* integer, *did:* integer)
Dept (*did:* integer, *dname:*, *managerid:* integer)

Write SQL code to create tables for Emp, Works, Dept such that Works has two foreign keys referring to Emp (*eid*) and Dept (*did*) respectively, and Dept has a foreign key (*managerid*) referring to Emp respectively. In addition, when a Dept tuple is deleted, all Works tuples referring to it should be deleted. When an Emp tuple is deleted, for all Dept tuples referring to it, the *managerid* should be set to null. Note *eid* is primary key for Emp, and *eid* and *did* together are primary key for Works. And *did* is the primary key for Dept.

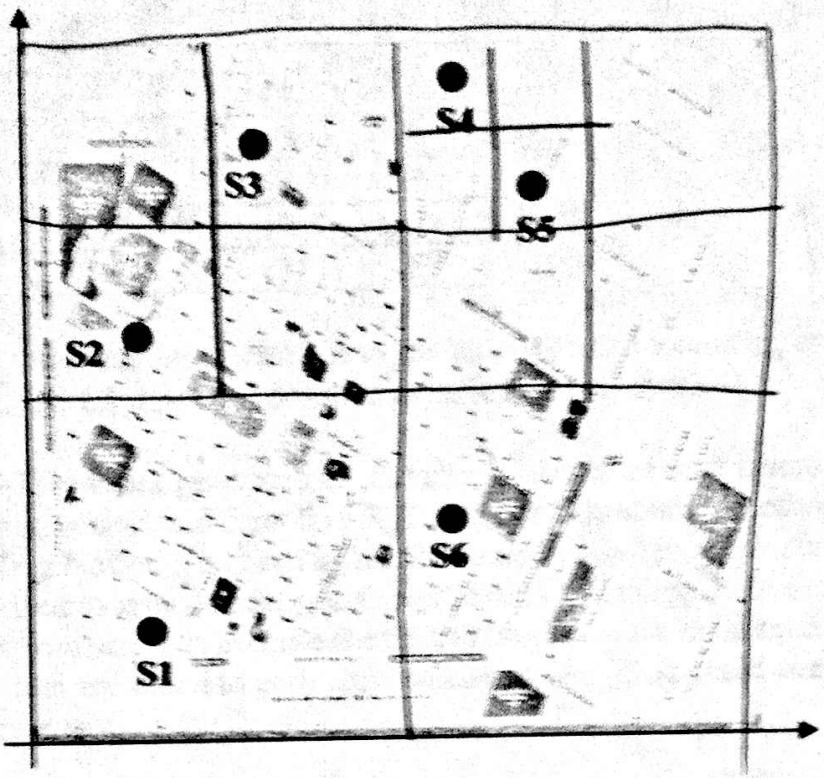
```
CREATE TABLE Emp (eid INTEGER,
                  ename CHAR(100),
                  age INTEGER,
                  salary REAL,
                  PRIMARY KEY (eid));

CREATE TABLE Dept (did INTEGER,
                   budget REAL,
                   managerid INTEGER,
                   PRIMARY KEY (did),
                   FOREIGN KEY (managerid)
                   REFERENCES Emp,
                   ON DELETE SET NULL);
```

```
CREATE TABLE Works (eid INTEGER,                    (1 pt)
                    did INTEGER,                    (1 pt)
                    PRIMARY KEY (eid, did),         (1 pt)
                    FOREIGN KEY (did)               (1 pt)
                    REFERENCES Dept,
                    FOREIGN KEY (eid) REFERENCES    (1 pt)
                    Emp,
                    ON DELETE CASCADE);             (1 pt)
```

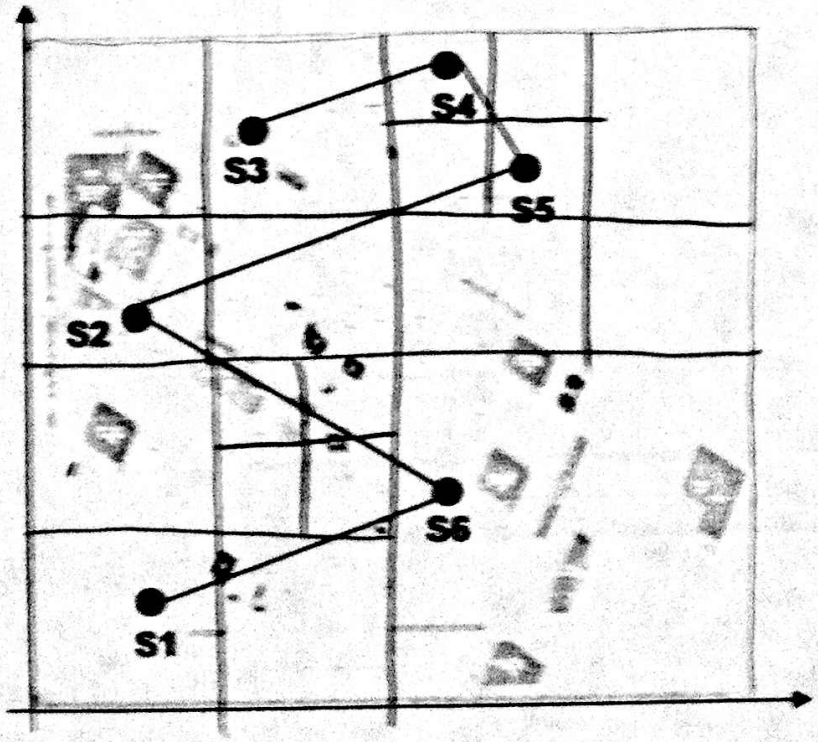
5) 10 pts

Consider the six Tram Stations shown in the following picture of USC campus.

(a) Build a PR Quadtree for these six points. (You do not need to draw the actual tree, just show the partitioning on the above figure)



(b) A Tram starts from the first station which is S1 and after passing through all the stations stops at the last one which is S3. Construct a PMI Quadtree for this path on the figure below.



6) 15 pts

A road network is maintained in a spatial database as following:

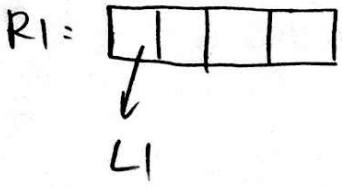
Road ID	Road Segment
L1	(4, 7), (6, 8)
L2	(6,4), (7, 7)
L3	(2,1), (3, 2)
L4	(0, 2), (1, 5)
L5	(3, 3), (5, 6)

1-2 : 0
1-3 : $\sqrt{26}$ ✓
1-4 : $\sqrt{13}$
2-3 : $\sqrt{13}$
2-4 : 5
3-4 : 1

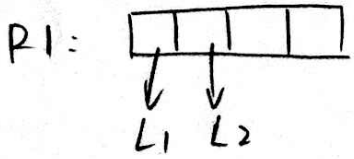
Assume that the roads are inserted in to the table with the ascending order of RoadID (i.e., L1, L2, L3, L4, L5). Also assume that (m,M)=(2,4).

Draw the R-Tree index generated for the above table after each insertion. In other words, you should draw five R-Trees. Use "Quadratic" method to split the R-Tree Nodes. You need to briefly describe what happens after each step. If you need to split a node, you should clearly and completely describe which line(s) you select to become the first element of each child node, and then which lines are added to each child node and why. You could use the following chart to draw the lines.

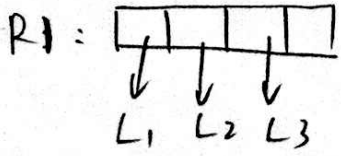
① Insert L1 :

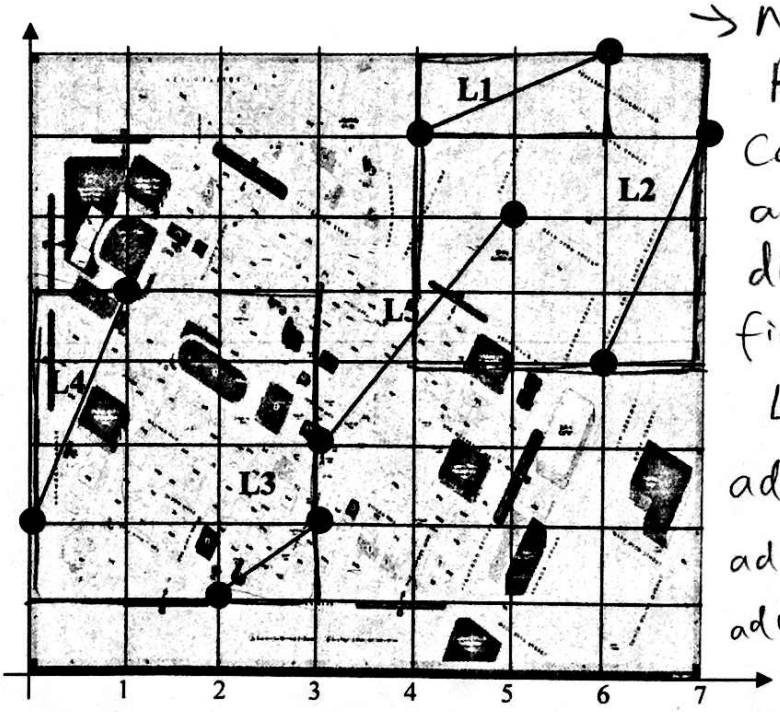


② Insert L2 :

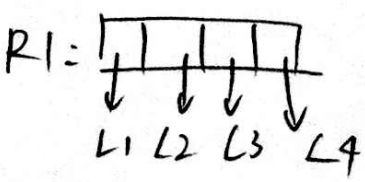


③ Insert L3 :





④ Insert L4 :



⑤ Insert L5 :

R1 overflow.
→ need to split R1.
Calculate all pairwise distances and find the seeds
L1 & L3.
add L2 : → L1
add L4 : → L3
add L5 : → L1