Question 1 [Computing queries – 50 points]. Consider the following relations

```
R(A,B,C) = {<1,2,3>,<1,0,3>}
S(B,C,D) = {<2,3,7>,<1,1,4>,<2,2,4>,<2,3,5>}
```

(1.a)

Padding

R' = {<1,2,3>,<1,0,3>,<null,1,1><null,2,2>}

S' = {<2,3,7>,<1,1,4>,<2,2,4>,<2,3,5>,<0,3,null>}

Join

$R \bowtie_{FULL} S$ = {<1,2,3,7>,<1,2,3,5>,<1,0,3,null>,<null,1,1,4>
<null,2,2,4>}

Projection

$\pi_{A,D}(R \bowtie_{FULL} S)$ = {<1,7>,<1,5>,<1,null>,<null,4>}

(1.b)

Answer

{<2>}

(1.c)

Cartesian product with selection

{<1,2,3,2,3,7>,<1,2,3,2,2,4>,<1,2,3,2,3,5>}

Group by S.C … two groups

{{<1,2,3,2,3,7>,<1,2,3,2,3,5>}{<1,2,3,2,2,4>}}

Select groups having COUNT > 1

{<1,2,3,2,3,7>,<1,2,3,2,3,5>}

Answer

{<3,6>}

Question 2

Answer:

```
CREATE TABLE Course
(
  cid      CHAR(6)  PRIMARY KEY,
  ctitle   CHAR(16),
  dept     CHAR(3)
)

CREATE TABLE Student
(
  sid      NUMERIC(8)  PRIMARY KEY,
  sname    VARCHAR(16),
  dg       VARCHAR(4),
  city     VARCHAR(10),
  CHECK ((dg = 'BSC') OR (dg = 'BENG') OR (dg = 'BA'))
  )

CREATE TABLE S_Take_C
(
  sid      NUMERIC(8),
  cid      CHAR(6),
  yr       NUMERIC(4),
  trm      CHAR(1),
  mark     INTEGER,
  CHECK ((trm = 'W') OR (trm = 'S') OR (trm = 'F')),
  CHECK ((mark ≥ 0) AND (mark ≤ 100)),
  UNIQUE(sid,cid),
  FOREIGN KEY(sid)
      REFERENCES Student(sid)
          ON DELETE CASCADE,
  FOREIGN KEY(cid)
      REFERENCES Course(cid)
          ON DELETE CASCADE
          ON UPDATE CASCADE
)
```

Question 3 [Relational Algebra – 50 points].

(3.a)

$\pi_{sid,sname}(\sigma_{cid=\text{“csc343”}}((Student \bowtie S\_Take\_C)))$

or

$\pi_{sid,sname}((\sigma_{cid=\text{“csc343”}}(Student) \bowtie S\_Take\_C)))$

(3.b)

$R1 = \pi_{yr,trm,mark}(\sigma_{cid=\text{”csc343”}}S\_Take\_C)$

$R2 = \pi_{yr,trm,mark}(\sigma_{mark1>mark}(R1 \bowtie \rho_{mark\text{->}mark1}R1))$

$Answer = R1 - R2$

(3.c) [20 points] “List student ids for students who have taken every computer science course”

Answer:

$R1 = \pi_{cid}(\sigma_{dept=\text{“CSC”}}(Course))$

$Answer = (\pi_{sid,cid}(S\_Take\_C) / R1)$

Question 4 [SQL – 50 points].

(4.a)

```
SELECT    cid,cname
FROM      Course
WHERE     cid not in (SELECT cid FROM S_Take_C)
```

(4.b)

Answer:

```
SELECT    COUNT (DISTINCT T.sid)
FROM      Course C, S_Take_C T
WHERE     C.cid = T.cid AND C.dept = ‘CSC’
```

(4.c)

Answer:

```
CREATE VIEW SA (sid,savg) AS
SELECT    T.sid, AVG (T.mark)
FROM      S_Take_C T
GROUP BY  T.sid

SELECT    S.sid, S.sname
FROM      Student S, SA
WHERE     S.sid = SA.sid AND savg ≥ ALL
(SELECT savg FROM SA)
```

Question 5 [True/False questions – 70 points].

(5.a) [_F_] The equality $((\mathbf{S} \bowtie \mathbf{R}) \bowtie \mathbf{Q}) = (\mathbf{S} \bowtie (\mathbf{R} \bowtie \mathbf{Q}))$ holds if and only if **S** and **R**, **S** and **Q**, **R** and **Q** share respectively at least one attribute;

(5.b) [_F_] Pointers are better than value-based references in a database because the latter are hardware-dependent;

(5.c) [_T_] Every relation in the Relational Model has at least one superkey consisting of all its attributes;

(5.d) **[_F_]** A trigger is an event that triggers an SQL statement to execute whenever the event occurs;

(5.e) **[__F_]** In embedded SQL, cursors can only be declared for database tables, and not for results of queries;

(5.f) [_T_] A DBMS supports mechanisms that allow multiple transactions to execute concurrently against a single database without interfering with each other;

(5.g) [_T_] If query `Q` evaluates to the empty table, then the condition `0 > ANY (Q)` is false;

(5.h) [_F_] If **S** is empty, then $\mathbf{S} \bowtie \mathbf{R} = \mathbf{R}$;

(5.i) [_F_] In embedded SQL, `SQLSTATUS` is a special programming language variable whose value describes the execution status of the application;

(5.j) [_F_] In SQL, all views can be updated just like database relations;

(5.k) [_T_] If relation **R** consists of a single attribute, then $\mathbf{R} \bowtie \mathbf{R} = \mathbf{R}$;

(5.l) [_F_] In SQL-DDL, one can define new aggregate functions that can then be used in SQL queries;

(5.m) [_F_] JDBC is a statement-level interface for executing SQL within a Java program;

(5.n) [_F_] Object-oriented databases make it possible for Java programs to access relational databases.