1. [10pts] For all students majoring in ‘ART’ and having taken at least 4 classes offered by ‘CS’ dept, list their user_id, first_name and last_name. Expected result row(s): 1

a) [7pts] SQL Query:

```
select User.user_id, User.first_name, User.last_name
from User, Student, Takes
where User.user_id = Student.user_id
AND Student.user_id = Takes.user_id
AND Student.major = 'ART'
AND Takes.dept = 'CS'
group by User.user_id, User.first_name, User.last_name
HAVING count(*) >=4;
```

b) [3pts] Result:

```
user_id, first_name, last_name
'13','Maureen','Wolff'
```

2. [10pts] For all assignments for class CS122A (cno ‘122A’ dept ‘CS’) in year 2017 and term Fall, list their assign_id, name and maximum scores completed by students. Expected result row(s): 4

a) [7pts] SQL Query:

```
select distinct Assignment.assign_id, Assignment.name, max(Does.grade)
from User, Student, Class, Assignment, Does
where User.user_id = Student.user_id
AND Class.cno = '122A' AND Class.dept = 'CS' AND Class.year = '2017' AND Class.term = 'Fall'
AND Assignment.class_no = Class.class_no AND Assignment.cno = Class.cno
AND Assignment.dept = Class.dept
AND User.user_id = Does.user_id AND Does.assign_id = Assignment.assign_id
group by Assignment.assign_id, Assignment.name;
```

b) [3pts] Result:

```
assign_id, name, max_grade
'152','Assignment 0','93.00'
'153','Assignment 1','84.00'
'154','Assignment 2','89.00'
'155','Assignment 3','90.00'
```

3. [10pts] For all assignments which belong to class CS223 (cno '223' and dept 'CS') that have been completed by *at least 90% of* the students taking that class, list their assign_id, name, and how many students have completed it. Expected result row(s): 4

a) [7pts] SQL Query:

```
select Assignment.assign_id, Assignment.name, count(*)
from User u1, Student s1, Takes t1, Class c1, Assignment, Does
where u1.user_id = s1.user_id
AND u1.user_id = t1.user_id
AND t1.class_no = c1.class_no AND t1.dept = c1.dept AND t1.cno = c1.cno
AND Assignment.class_no = c1.class_no AND Assignment.cno = c1.cno AND Assignment.dept = c1.dept
AND u1.user_id = Does.user_id AND Does.assign_id = Assignment.assign_id
AND c1.dept = 'CS' AND c1.cno = '223'
group by Assignment.assign_id, Assignment.name, c1.class_no, c1.dept, c1.cno
having count(*) >=
        (select count(*)*0.9
   from Student, Takes
   WHERE Student.user_id = Takes.user_id
   AND Takes.class_no = c1.class_no AND Takes.dept = c1.dept AND Takes.cno = c1.cno)
```

b) [3pts] Result:

```
assign_id, name, count
'434','Assignment 0','10'
'435','Assignment 1','9'
'441','Assignment 0','3'
'450','Assignment 0','10'
```

4. **Views** [20 pts]

a) [15 pts] Create the desired view (CSStudentView) by writing an appropriate CREATE VIEW statement.

CREATE VIEW CSStudentView …;

```
CREATE VIEW CSStudentView as

select User.user_id, User.first_name, User.last_name, Takes.class_no, Takes.dept, Takes.cno, Takes.grade,
Course.title, Course.level

from User, Student, Takes, Class, Course

where User.user_id = Student.user_id

AND Student.major = 'CS'

AND Takes.user_id = user.user_id

AND Takes.class_no = Class.class_no AND Takes.cno = Class.cno AND Takes.dept = Class.dept

AND Class.cno = Course.cno AND Class.dept = Course.dept;
```

b) [5 pts] Show the usefulness of your view by writing a SELECT query against the view that prints the user_id, first_name, last_name of the students who have the highest grade in the course titled 'Transaction Processing'. Expected result set: 1 row

```
select user_id, first_name, last_name from CSStudentView

where title = 'Transaction Processing' AND

grade = (select max(grade) from CSStudentView where title = 'Transaction Processing')
```

5. **Stored Procedures** [20 pts]

a) [15 pts] Create and exercise a SQL stored procedure called RegisterStudent(…) that the site's application can use to add a new student with a mobile phone to the database.

```
DELIMITER //
        CREATE PROCEDURE RegisterStudent(
            user_id varchar(20),
             email varchar(50),
             first_name varchar(30),
             last_name varchar(30),
             major varchar(50),
             mobile_number varchar(20))
        BEGIN
                INSERT INTO User VALUES (user_id, email, first_name, last_name);
                INSERT INTO Student VALUES(user_id, major);
                INSERT INTO Phone VALUES(user_id, 'Mobile', mobile_number);
```

```
        END;  //
DELIMITER ;
```

b) [5pts] Verify that your new stored procedure is working properly by calling it as follows to add a new student and then running a SELECT query to show the procedure's after-effects:

'12345','johndoe@email.com','CS','123-456-789','Mobile'

6. **Alter Table** [10 pts]

The CTO wishes to modify the original schema to accommodate a new feature. As the database already has valuable information that we need to retain, we just want to alter one of the existing tables to fulfill the CTO's request. The new feature is to force a user's phone data to be deleted when the user is deleted. Recall that the current Phone table definition is as follows:

```
CREATE TABLE Phone(
  user_id      VARCHAR(20),
  type         ENUM('Mobile','Office','Home') NOT NULL,
  number       VARCHAR(20),
  PRIMARY KEY (user_id,type,number),
  FOREIGN KEY (user_id) REFERENCES User (user_id)
);
```

a) [5 pts] Write and execute the ALTER TABLE statement(s) needed to modify the Phone table so that whenever the user associated with the phone information is deleted then the phone data is deleted too. (Note: the name of the existing foreign key constraint for user_id is phone_ibfk_1).

```
ALTER TABLE Phone DROP FOREIGN KEY phone_ibfk_1;

ALTER TABLE Phone ADD FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE;
```

b) [5 pts] Execute the following DELETE and SELECT statements to show the effect of your change:

empty result set

7. **Triggers** [20 pts]

a) [15 pt] Use the CREATE TRIGGER statement in MySQL to define a trigger that will do the desired job.

```
delimiter //

CREATE TRIGGER update_popularity AFTER INSERT ON Likes

FOR EACH ROW

BEGIN
```

```
        UPDATE Post SET popularity = (select count(*) from Likes where Likes.post_id = New.post_id)
        where Post.post_id = New.post_id;
END;//
delimiter ;
```

b) [5 pts] Execute the following INSERT and then SELECT statements to show the effect of your trigger. Write down the popularity returned by each SELECT statement.

```
SELECT popularity FROM Post where Post.post_id = '1'; => 26
INSERT INTO Likes VALUES('1', '1');
SELECT popularity FROM Post where Post.post_id = '1'; => 12
INSERT INTO Likes VALUES('5', '1');
SELECT popularity FROM Post where Post.post_id = '1'; => 13
```