BS-CS Fall 2012
Mid Term Examination – December 2012

Student ID:  
Student Name:  
Shift / Section: Morning / Afternoon  
Student Signature:  

Course Code: CMP-370  
Course Title: Database Management System  
Maximum Time: 90 Minutes  
Maximum Marks: 70  

Instructions

• Read questions carefully before solving.
• Solve the questions on provided blank space under each question.
• Understanding of the questions is part of examination. So, none of your queries will be entertained in the examinations hall.
• If there is any ambiguity in the paper, benefit will be given to the students.
• Paper has 8 pages including title page.

<table>
<thead>
<tr>
<th>Question #</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Marks</td>
<td>20</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>20</td>
<td></td>
<td>70</td>
</tr>
<tr>
<td>Marks Obtained</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

 Examiner Signature:  
 Marks Out of 35:  

DO NOT OPEN UNTIL YOU ARE TOLD TO DO SO
Question 1:

The boat reservations database has the following schema:

- **sailor**: sname (string), rating (integer)
- **boat**: bname (string), color (string), rating (integer)
- **reservation**: sname (string), bname (string), weekday (string)

The rating attribute for boats indicates the minimum rating required of a sailor reserving the boat. In addition, the following hold:

- *sname* is the primary key of *sailor* (so every sailor has just one rating);
- *bname* is the primary key of *boat* (so every boat has just one color and one rating);
- *bname* is a foreign key in the *reservation* relation, referencing relation *boat* (so every *bname* in *reservation* occurs in *boat*, but the converse need not be true);
- *sname* is a foreign key in the *reservation* relation, referencing relation *sailor* (so every *sname* in *reservation* occurs in *sailor*, but the converse need not be true).

Here is one example instance given in Figure 1 over the above schema:

<table>
<thead>
<tr>
<th>sailor</th>
<th>sname</th>
<th>rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brutus</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Andy</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Horatio</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Rusty</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boat</th>
<th>bname</th>
<th>color</th>
<th>rating</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SpeedQueen</td>
<td>white</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Interlake</td>
<td>red</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Marine</td>
<td>blue</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Bay</td>
<td>red</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>reservation</th>
<th>sname</th>
<th>bname</th>
<th>weekday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andy</td>
<td>Interlake</td>
<td>Monday</td>
<td></td>
</tr>
<tr>
<td>Andy</td>
<td>Bay</td>
<td>Wednesday</td>
<td></td>
</tr>
<tr>
<td>Andy</td>
<td>Marine</td>
<td>Saturday</td>
<td></td>
</tr>
<tr>
<td>Rusty</td>
<td>Bay</td>
<td>Sunday</td>
<td></td>
</tr>
<tr>
<td>Rusty</td>
<td>Interlake</td>
<td>Wednesday</td>
<td></td>
</tr>
<tr>
<td>Rusty</td>
<td>Marine</td>
<td>Wednesday</td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td>Bay</td>
<td>Monday</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Database instance for Question 1

Write the following queries in SQL (of course, the queries must work on all data, not just the sample one):

a) **(3 points)** List all boats reserved on Wednesday and their color.

```sql
select distinct reservation.bname, color from reservation, boat
where reservation.bname = boat.bname and reservation.weekday = 'Wednesday'
```

b) **(3 points)** List all pairs of sailors who have reserved boats on the same day.

```sql
select distinct x.sname, y.sname from reservation x, reservation y
where x.weekday = y.weekday
```
c) **(5 points)** List the days appearing in the reservation relation for which only red boats are reserved.

```
select distinct weekday from reservation
where weekday not in
  (select r.weekday from reservation r, boat b
  where r.bname = b.bname and b.color <> 'red')
```

d) **(4 points)** For each day, list the number of red boats reserved on that day.

```
select r.weekday, count(distinct r.bname) as number from reservation r, boat b
where r.bname = b.bname and b.color = 'red'
group by weekday
```

e) **(5 points)** For each day of the week occurring in the reservation relation, list the average rating of sailors having reserved boats that day.

```
select weekday, avg(rating) from sailor, reservation
where sailor.sname = reservation.sname
group by weekday
```

**Question 2:** (Marks 10)

Consider the following relations:

- **Emp(eno, ename, title, city)**
- **Proj(pno, pname, budget, city)**
- **Works(eno, pno, resp, dur)**
- **Pay(title, salary)**

where the primary keys are underlined, and Emp.title is a foreign key to Pay.title, Works.eno is a foreign key to Emp.eno, and Works.pno is a foreign key to Proj.pno.

For each part of this question (considered independently of the other parts), write a single SQL statement that accomplishes the given requirements.

a) **(3 points)** For each city, how many projects are located in that city and what is the total budget over all projects in the city?

```
SELECT city, count(pno) AS totproj, sum(budget) AS totbudget
FROM Proj
GROUP BY city
```

b) **(3 points)** List all projects located in “Lahore” and include for each one the number of persons working on the project.

```
```
SELECT Proj.pno, count(*) FROM Proj LEFT OUTER JOIN Works ON (Proj.pno=Works.pno) WHERE city=`Lahore` GROUP BY Proj.pno, pname

c) (4 points) Formulate the following SQL query in relational algebra. This is expressed over the same schema as above.

\[
\text{SELECT} \ \text{pname, budget} \\
\text{FROM} \ \text{Proj, Works, Emp} \\
\text{WHERE title =`Programmer`} \\
\text{AND Works.eno = Emp.eno} \\
\text{AND Works.pno = Proj.pno} \\
\]

*pname,budget (Proj ./ Works ./ (-- title=`Programmer' Emp))

You can put the selection outside all of the joins if you wish; that would work too:
*pname,budget (--title=`Programmer'(Proj ./ Works ./ Emp))

Question 3: (Marks 10)

Consider the instance of the Sailors relation given in Figure 2 where the key is underlined. Show the results of the following query.

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>Dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>29</td>
<td>Brutus</td>
<td>1</td>
<td>33.0</td>
</tr>
<tr>
<td>31</td>
<td>Lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>32</td>
<td>Andy</td>
<td>8</td>
<td>25.5</td>
</tr>
<tr>
<td>58</td>
<td>Rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
<tr>
<td>64</td>
<td>Horatio</td>
<td>7</td>
<td>35.0</td>
</tr>
<tr>
<td>71</td>
<td>Zorba</td>
<td>10</td>
<td>16.0</td>
</tr>
<tr>
<td>74</td>
<td>Horatio</td>
<td>9</td>
<td>35.0</td>
</tr>
<tr>
<td>85</td>
<td>Art</td>
<td>3</td>
<td>25.5</td>
</tr>
<tr>
<td>95</td>
<td>Bob</td>
<td>3</td>
<td>63.5</td>
</tr>
</tbody>
</table>

Figure 2: Database instance for Question 3

a) (5 points)

\[
\text{SELECT} \ \text{S.sid} \\
\text{FROM} \ \text{Sailors S} \\
\text{WHERE S.rating >= ALL} \\
\text{(SELECT S2.rating} \\
\text{FROM Sailors S2 )} \\
\]

Note: The result that is returned is a table. You need to specify the column title.

Sid
58
71

b) (5 points)


SELECT S.rating, AVG(S.age) AS average
FROM Sailors S
GROUP BY S.rating
HAVING 1 <
(SELECT COUNT(*)
FROM SAILORS S2
WHERE S.rating = S2.rating)

Note: The result that is returned is a table. You need to specify the column title.

<table>
<thead>
<tr>
<th>Rating</th>
<th>average</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>44.5</td>
</tr>
<tr>
<td>7</td>
<td>40</td>
</tr>
<tr>
<td>8</td>
<td>40.5</td>
</tr>
<tr>
<td>10</td>
<td>25.5</td>
</tr>
</tbody>
</table>

Question 4: (Marks 10)

Consider the database consisting of one relation:

movie: title, director, actor.

Express the following queries in SQL.

a) (5 points) List the actors cast only in movies by Berto.

```
SELECT actor FROM movie
WHERE actor NOT IN
  (SELECT actor FROM movie
   WHERE director != Berto)
```

b) (5 points) List all pairs of distinct actors who act together in at least one movie.

```
SELECT m1.actor as actor1, m2.actor as actor2
FROM movie m1, movie m2
WHERE m1.title = m2.title
```