CS 3743
Introduction to Database Systems
Midterm 1 Solutions

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1</td>
<td></td>
</tr>
<tr>
<td>Question 2</td>
<td></td>
</tr>
<tr>
<td>Question 3</td>
<td></td>
</tr>
<tr>
<td>Question 4</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
</tr>
</tbody>
</table>

NAME:____________________________________

Instructions
1. Do all of the problems
2. You have 70 minutes for the exam
3. Show all your work
4. Do not separate exam papers

<table>
<thead>
<tr>
<th>Easy</th>
<th>Difficulty Level</th>
<th>Difficult</th>
</tr>
</thead>
<tbody>
<tr>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. **E-R Diagram (28 pts):** Draw the E-R diagram for a sports organization. The following describes this organization.

- Each league has a league name and sport. League names are unique.
- Leagues consist of teams. Each team has a name, hometown, division and conference. No league has two teams with the same name. However, there can be teams with the same name in different leagues.
- Teams consist of players. Each player has a name, jersey number, college and rookie year. No team has two players with the same name. However, there can be players with the same number on different teams.

You can draw ER diagram here.

**Solution:**

![ER Diagram](image)

**Figure 1: ER Diagram**
2. *SQL (24 pts):* Consider the following relations for keeping track of students and the jobs they are seeking

\[
\begin{align*}
\text{Student}(sno, \text{ sname, major, GPA}) \\
\text{Appliedfor}(asno, \text{ ajno}) \\
\text{Job}(jno, \text{ jname, company, salary})
\end{align*}
\]

Write the following queries in *SQL*

(a) Find the average gpa of students who have applied for jobs

\[
\begin{align*}
\text{select avg(gpa)} \\
\text{from student} \\
\text{where sno in (select asno} \\
\text{from Appliedfor);}
\end{align*}
\]

(b) For each company, print the number of job openings at that company

\[
\begin{align*}
\text{select company, count(*)} \\
\text{from Job} \\
\text{group by company;}
\end{align*}
\]

(c) Find the names of jobs which has at least 10 applications

\[
\begin{align*}
\text{select jname, count(*)} \\
\text{from Job, Appliedfor} \\
\text{where ajno = jno} \\
\text{group by jname} \\
\text{having count(*)} \geq 10;
\end{align*}
\]

(d) Find the number and names of jobs whose salary is higher than avg salary of jobs offered by IBM

\[
\begin{align*}
\text{select jno, jname} \\
\text{from Job} \\
\text{where salary} > (\text{select avg(salary)} \\
\text{from Job} \\
\text{where company} = 'IBM');
\end{align*}
\]
3. **Relational Algebra (24 pts):** Consider the following database schema that keeps track of computers sold at an online store. The product relation gives the manufacturer, model number and type (PC or Laptop) of various products. We assume that model numbers are unique over all manufacturers and product types. The PC relation gives for each model number that is a PC the speed (of the processor, in Gigahertz), the amount of RAM (in gigabytes), the size of the hard disk (in gigabytes), the speed and type of removable disk (CD or DVD) and the price. The laptop relation is similar, except that the screen size (in inches) is recorded in place of information about the removable disk.

\[
\text{Product}(\text{maker}, \text{model}, \text{type})
\]
\[
\text{PC}(\text{pmodel}, \text{pspeed}, \text{pram}, \text{phd}, \text{rd}, \text{pprice})
\]
\[
\text{Laptop}(\text{lmodel}, \text{lspeed}, \text{lram}, \text{lhd}, \text{screen}, \text{lprice})
\]

Specify the following queries in **Relational Algebra** using above database schema.

(a) Which PC models have a speed of at least 2Ghz?

\[
\Pi_{\text{model}}(\sigma_{\text{pspeed} \geq 2}(\text{PC}))
\]

(b) What is the average price of a laptop?

\[
F_{\text{avg lprice}}(\text{Laptop})
\]

(c) Which manufacturers make laptops with a hard disk of at least one gigabyte?

\[
\Pi_{\text{maker}}(\sigma_{\text{lhd} \geq 1}(\text{Product } \bowtie_{\text{model} = \text{lmodel}} \text{Laptop}))
\]

(d) Find the manufacturers that sell Laptops but not PCs?

\[
(\Pi_{\text{maker}}(\text{Product } \bowtie_{\text{model} = \text{lmodel}} \text{Laptop})) - (\Pi_{\text{maker}}(\text{Product } \bowtie_{\text{model} = \text{lmodel}} \text{PC}))
\]
4. **SQL statements (24 pts):** Consider the following database scheme that keeps track of World War II capital ships. It involves the following relations:

- **Classes**(cclass, type, country, numguns, bore, displacement)
- **Ships**(sname, sclass, launched)
- **Battles**(sname, date)
- **Outcomes**(ship, battle, result)

Ships are built in classes from the same design and the class is usually named for the first ship of that class. The relation classes records the name of the class, the type (bb for battleship or bc for battlecruiser), the country that built the ship, the number of main guns, the bore (diameter of the gun barrel) of the main guns, and the displacement (weight, in tons). Relation ships records the name of the ship, the name of its class and the year in which the ship was launched. Relation battles gives the name and date of battles involving the ships. The relation Outcomes gives the result (sunk, damaged or ok) for each ship in the battle.

Specify the following queries in **SQL**

(a) Find the name and country for all ships with at least 10 guns?

```sql
select sname, county
from ships, classes
where sclass = cclass and numguns >=10;
```

(b) Find all ships that have the same name as their class?

```sql
select sname
from ships
where sname = sclass;
```

(c) Find the classes of ships at least one of which was sunk in a battle?

```sql
select class
from ships
where name in (select ship
                    from outcomes
                    where result = 'sunk');
```

(d) Find those countries that have both battleships and battlecruizers?

```sql
(select country
from classes
where type = 'bb') intersection (select country
                                    from classes
                                    where type = 'bc');
```