Midterm Exam #2 (Version B)
CS 122A
Winter 2017

Max. Points: 100
(Please read the instructions carefully)

Instructions:
- The total time for the exam is 50 minutes; be sure to budget your time accordingly.
- The exam is closed book and closed notes but “open cheat sheet”.
- Read each question first, in its entirety, and then carefully answer each part of the question.
- If you don’t understand something, ask one of the exam proctors for clarification.
- If you still find ambiguities in a question, note the interpretation you are taking.

<table>
<thead>
<tr>
<th>QUESTION</th>
<th>TOPIC</th>
<th>POINTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Short Answers</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Query Writing</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>Query Matching</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>SQL Execution</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>True or False?</td>
<td>20</td>
</tr>
<tr>
<td>TOTAL</td>
<td>All</td>
<td>100</td>
</tr>
</tbody>
</table>
Question 1: Short Answers (10 points)

(a) (1 pt each) Given two relations $R$ and $S$, where $R$ has $N_R$ tuples, $S$ has $N_S$ tuples, and $N_S > N_R > 0$, what are the maximum and minimum possible result cardinalities for the relational algebra queries $R \sqcap S$, $R \cup S$, and $R - S$ expressed in terms of $N_R$ and $N_S$?

$R \sqcap S$: max size: $N_S \times N_R$          min size: $N_S \times N_R$

$R \cup S$: max size: $N_S + N_R$          min size: $N_S$

$R - S$: max size: $N_R$          min size: $0$

(b) (4 points) Suppose that the table $\text{Emps}(\text{eid, name, salary, bonus})$ contains only two tuples:

(101, ’Sue’, NULL, 5000.0)
(102, ’Jeb’, 3000.0, NULL)

For each of the following SQL queries, show the output that it would produce if run against $\text{Emps}$:

(i) (2 pts) SELECT salary + bonus AS answer FROM Emps WHERE name = ’Sue;

    NULL

(ii) (2 pts) SELECT SUM(bonus) AS answer FROM Emps;

    5000.0

SCORE: _________
Question 2: Query Writing (20 points)

(20 pts) Consider again the Hoofers Sailing Club database, and suppose (if needed or helpful) that the tables contain the data shown below.

Sailors(sid, sname, rating, age)
Reserves(sid, bid, date)
Boats(bid, bname, color)

Write the following queries in the indicated language:

(10 pts) **SQL:**
For each favored boat color, print the color along with its total number of reservations and the number of different days on which boats of that color are reserved. (A color is said to be a favored boat color if 2 or more reservations exist for boats of that color.)

```
SELECT b.color, count(*) as cnt, count(DISTINCT r.date)
FROM Boats b, Reserves r
WHERE b.bid = r.bid
GROUP BY b.color
HAVING cnt >=2
```

(10 pts) **Relational Algebra:**
Print the colors of Boats that are reserved by both a sailor with a rating of less than 6 and a 35-year-old sailor with a rating of at least 4.

SCORE: _________
\[ \pi \text{ color } ((\pi \text{ bid } ((\sigma \text{ age } = 35 \land \text{ rating } \geq 4 (\text{Sailor})) \bowtie \text{ Reserves}) \cap \pi \text{ bid } ((\sigma \text{ rating } < 6 (\text{Sailor})) \bowtie \text{ Reserves})) \bowtie \text{ Boats}) \]

**Question 3: Query Matching (20 points)**

(20 pts) Consider our old favorite Hoofers Sailing Club example database, the schema for which is sketched below:

\[
\begin{align*}
\text{Sailors}(\text{sid}, \text{sname, rating, age}) \\
\text{Reserves}(\text{sid, bid, date}) \\
\text{Boats}(\text{bid, bname, colo})
\end{align*}
\]

Here is a list of query meanings to choose from when answering the following questions:

- **Q1** – Print the minimum age of sailors.
- **Q2** – Print the maximum age of sailors.
- **Q3** – Print the maximum age of sailors who have at least one boat reserved.
- **Q4** – Not a legal SQL query.
- **Q5** – None of the meanings listed above!

(4 pts each) For each of the following SQL queries, indicate the meaning of the query in the space to its right by choosing the appropriate answer from the list of meanings (one of Q1 through Q5) given above. Note that you can (and may need to) use the same answer multiple times!

**SQL QUERY:**

<table>
<thead>
<tr>
<th>SQL QUERY</th>
<th>MEANING:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT S.age</td>
<td><strong>Q1</strong></td>
</tr>
<tr>
<td>FROM Sailors S</td>
<td></td>
</tr>
<tr>
<td>ORDER BY S.age ASC</td>
<td></td>
</tr>
<tr>
<td>LIMIT 1;</td>
<td></td>
</tr>
<tr>
<td>SELECT MAX(age)</td>
<td><strong>Q3</strong></td>
</tr>
<tr>
<td>FROM Sailors S, Reserves R</td>
<td></td>
</tr>
<tr>
<td>WHERE S.sid = R.sid;</td>
<td></td>
</tr>
<tr>
<td>SELECT MAX(age)</td>
<td></td>
</tr>
<tr>
<td>FROM Sailors S</td>
<td></td>
</tr>
</tbody>
</table>

**SCORE:** _______
WHERE EXISTS
(SELECT * FROM Boat);

SELECT MAX(age)
FROM Sailors S
LEFT OUTER JOIN Reserves R
ON (S.sid = R.sid);

SELECT S.age
FROM Sailors S
WHERE MIN(S.age);

Question 4: SQL Execution (30 points)

Consider yet again the Hoofers Sailing Club database containing the data shown below:

<table>
<thead>
<tr>
<th>bid</th>
<th>bname</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interlake</td>
<td>maroon</td>
</tr>
<tr>
<td>2</td>
<td>Sailfish</td>
<td>yellow</td>
</tr>
<tr>
<td>3</td>
<td>Clipper</td>
<td>green</td>
</tr>
<tr>
<td>4</td>
<td>Yacht</td>
<td>green</td>
</tr>
<tr>
<td>5</td>
<td>Sailfish</td>
<td>yellow</td>
</tr>
<tr>
<td>6</td>
<td>Dinghy</td>
<td>maroon</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>bid</th>
<th>date</th>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4</td>
<td>2017-03-15</td>
<td>10</td>
<td>Robbie</td>
<td>5</td>
<td>35</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>2017-04-15</td>
<td>20</td>
<td>Sandra</td>
<td>9</td>
<td>22</td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>2017-04-15</td>
<td>30</td>
<td>Zachary</td>
<td>10</td>
<td>19</td>
</tr>
<tr>
<td>40</td>
<td>4</td>
<td>2018-01-01</td>
<td>40</td>
<td>Abigail</td>
<td>3</td>
<td>null</td>
</tr>
<tr>
<td>50</td>
<td>1</td>
<td>2017-12-25</td>
<td>50</td>
<td>Joseph</td>
<td>null</td>
<td>35</td>
</tr>
</tbody>
</table>

In addition, assume that the following additional SQL goodies have already been defined:

CREATE TRIGGER myTrigger
AFTER DELETE ON Sailors FOR EACH ROW
BEGIN
    UPDATE Reserves
    SET bid = NULL
    WHERE sid = OLD.sid;
END$$
DELIMITER ;

DEIMITER $$
CREATE PROCEDURE NewSailor(sname varchar(20), age float)
BEGIN
    DECLARE new_sid INT(11);
    SET new_sid = (SELECT MAX(sid)+1 FROM Sailors);
    INSERT INTO Sailors(sid, sname, rating, age)
    VALUES (new_sid, sname, 0, age);
END$$

SCORE: _________
(a) (10 pts) Show below the result of running the following query against the tables above:

```
SELECT S.sname, B.bname, S.age, B.color,
FROM (Sailors S LEFT OUTER JOIN Reserves R ON (S.sid = R.sid))
JOIN Boats B ON (R.bid = B.bid)
WHERE S.rating >= 3 AND S.age = 35.0;
```

Robbie, Yacht, 35, green
Robbie, Sailfish, 35, yellow

Question 4: SQL Execution (continued)

Here’s all of the database information once again:

<table>
<thead>
<tr>
<th>bid</th>
<th>bname</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interlake</td>
<td>maroon</td>
</tr>
<tr>
<td>2</td>
<td>Sailfish</td>
<td>yellow</td>
</tr>
<tr>
<td>3</td>
<td>Clipper</td>
<td>green</td>
</tr>
<tr>
<td>4</td>
<td>Yacht</td>
<td>green</td>
</tr>
<tr>
<td>5</td>
<td>Sailfish</td>
<td>yellow</td>
</tr>
<tr>
<td>6</td>
<td>Dinghy</td>
<td>maroon</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>bid</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4</td>
<td>2017-03-15</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>2017-04-15</td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>2017-04-15</td>
</tr>
<tr>
<td>40</td>
<td>4</td>
<td>2018-01-01</td>
</tr>
<tr>
<td>50</td>
<td>1</td>
<td>2017-12-25</td>
</tr>
</tbody>
</table>

DELMITER $$
CREATE PROCEDURE NewSailor(sname varchar(20), age float)
BEGIN
  DECLARE new_sid INT(11);
  SET new_sid = (SELECT MAX(sid)+1 FROM Sailors);
  INSERT INTO Sailors(sid, sname, rating, age)
    VALUES (new_sid, sname, 0, age);
END$$
DELMITER $$
DELIMITER $$
CREATE TRIGGER myTrigger
AFTER DELETE ON Sailors
FOR EACH ROW
BEGIN
  UPDATE Reserves
  SET bid = NULL
  WHERE sid = OLD.sid;
END$$
DELIMITER ;

(b) (10 pts) Show below the results of executing the following statement against the tables above. (I.e., show the contents of the above tables after execution; you may skip the tables that stay the same.)

```
CALL NewSailor('Sandra', 25.0);
```

51, Sandra, 0, 25.0
(c) (10 pts) Show below the results of executing the following statement against the initial tables above. (I.e., show the contents of the above tables after execution; you may skip the tables that stay the same.)

```
DELETE FROM Sailors S WHERE S.sname LIKE '%a%';
```

**Sailors: 4pts**
- Robbie
- Joseph

**Reserves: 6pts**
- 10, 4, ...
- 10, 5, ...
- 30, null, ...
- 40, null, ...
- 50, 1, ...

SCORE: _________
Question 5: True or False? (20 points)

(2 pts each) For each of the following statements, indicate whether the statement is true (circle TRUE) or false (circle FALSE):

- The relational algebra and the safe subset of the tuple relational calculus are equivalent in their expressive power.
  
  TRUE  FALSE

- Views based on joining two or more tables together are always updatable.
  
  TRUE  FALSE

- SQL is a more powerful language than the relational algebra.
  
  TRUE  FALSE

- SQL/PSM (the SQL standard’s stored procedure extension) is more powerful than SQL.
  
  TRUE  FALSE

- Views can be queried just as if they were stored tables.
  
  TRUE  FALSE

- Given a relational schema, there is only one way to translate a given English query against that schema’s data into a correct SQL query.
  
  TRUE  FALSE

- To express universal quantification (for all) queries, the SQL language includes a DIVISION operation modeled after the corresponding relational algebra operator.
  
  TRUE  FALSE

- Triggers in SQL are more powerful than foreign key constraints.
  
  TRUE  FALSE

- To be union-compatible (e.g., combinable with UNION or UNION ALL), two SQL tables or sub-queries must have the same number of columns with the same data types and the same column names.
  
  TRUE  FALSE

- The set operation INTERSECT is necessary in SQL because it is not always possible to express an equivalent query by appropriately using a JOIN operation instead.
  
  TRUE  FALSE

SCORE: __________