Midterm Exam #2 (Version A)
CS 122A
Spring 2018

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>True, False, or Null?</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>Name That Query</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>How Queryous</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>Never Ending SQL</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>And Briefly</td>
<td>10</td>
</tr>
<tr>
<td>TOTAL</td>
<td>All</td>
<td>100</td>
</tr>
</tbody>
</table>
Question 1: True, False, or Null? (20 points)

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

- Enforcing the uniqueness of a table’s column (e.g., User.email) would be a good use of a trigger.
  
  **TRUE**  
  **FALSE**

- Most real SQL implementations disallow CHECK constraints that involve correlated subqueries.
  
  **TRUE**  
  **FALSE**

- Two tables (or queries) are considered by SQL to be UNION-compatible if they have the same number of columns and the same column names in the same order.
  
  **TRUE**  
  **FALSE**

- Consider the query “SELECT * FROM Orders o INNER JOIN Lineitems l ON (o.oid = l.oid)”. The result tuple count for this query could be larger if “INNER” were “LEFT OUTER” instead.
  
  **TRUE**  
  **FALSE**

- It’s possible to write a tuple relational calculus query involving Sailor, Boat, and Reservation relations containing less than 20 tuples each whose result tuple count is infinite.
  
  **TRUE**  
  **FALSE**

- It’s possible to write a relational algebra query involving small Sailor, Boat, and Reservation relations whose result tuple count is infinite.
  
  **TRUE**  
  **FALSE**

- The use of triggers and stored procedures is advisable when writing database applications that need to be easily run on different relational DBMS choices (e.g., MySQL, Oracle, DB2, …).
  
  **TRUE**  
  **FALSE**

- If “CREATE TABLE Shirts (name VARCHAR(40), size ENUM('small', 'medium', 'large'))” is the DDL for the table Shirts, “SELECT size FROM Shirts” can have at most 3 result rows.
  
  **TRUE**  
  **FALSE**

- The SQL set operation EXCEPT is not actually necessary because it is always possible to express an equivalent query by appropriately using a subquery instead.
  
  **TRUE**  
  **FALSE**

- The query “(SELECT * FROM Boat B WHERE B.color = ‘yellow’) UNION ALL (SELECT * FROM Boat B WHERE B.color != ‘yellow’) is a way to (inefficiently ⊕) return all Boat tuples.
  
  **TRUE**  
  **FALSE**

**SCORE: __________**
Question 2: Name That Query (30 points)

(30 pts) Consider the Hoofers Driving Club database:

Drivers(did, dname, rating, age)  Reserves(did, cid, rdate)  Cars(cid, cname, color)

What follows is a list of queries in various languages. For the first five queries, Q1-Q5, list all of the equivalent queries from the remainder of the list (Q6-Q16). Note that there may be several, or perhaps none, and that two queries are equivalent if their results are identical under all database states.

Q1 (6 pts) – SELECT cid FROM Cars WHERE color = 'red' AND color = 'purple';

Equivalent queries:

Q2 (6 pts) – SELECT MAX(age) FROM Drivers;

Equivalent queries:

Q3 (6 pts) – \( \pi_{cid,cname,color} ((\sigma_{age=22} Drivers) \bowtie (Reserves \bowtie Cars)) \)

Equivalent queries:

Q4 (6 pts) – \{ t(age) | \exists d \in Drivers (t.age = d.age \land (d.rating < 3)) \}

Equivalent queries:

Q5 (6 pts) – Select the ids and names of drivers to whom alcohol can't be legally served at a UCI party; (Proof of age 21 or over required)

Equivalent queries:

Q6 – SELECT * FROM Cars C WHERE C.cid IN
   (SELECT R.cid FROM Reserves R WHERE R.did IN
   (SELECT D.did FROM Drivers D WHERE D.age = 22));

Q7 – SELECT R.cid FROM Reserves R WHERE FALSE;

Q8 – SELECT DISTINCT C.cid, C.cname, C.color FROM Reserves R, Drivers D, Cars C
   WHERE D.age = 22 AND D.did = R.did AND R.cid = C.cid;

Q9 – SELECT age FROM Drivers WHERE rating < 3;

Q10 – SELECT D.age FROM Drivers D ORDER BY D.age \text{ASC LIMIT 1};

Q11 – SELECT did, dname FROM Drivers WHERE age < 21;

Q12 – \pi_{age (\sigma_{rating < 3} Drivers)}

Q13 – \{ t(age) | \exists d \in Drivers (t.age = d.age \land (d.rating >= 3)) \}

Q14 – SELECT R.cid FROM Reserves R, Cars C1, Cars C2
   WHERE (C1.color = 'red' OR R.cid = C1.cid) \land (C2.color = 'purple' OR R.cid = C2.cid);

Q15 – \{ t(age) | \exists d \in Drivers (t.age = d.age \land \nexists d2 \in Drivers (d2.age > d.age)) \}

Q16 – SELECT did, dname FROM Drivers WHERE age < 21 OR age IS NULL;

SCORE: _________
Question 3: How Queryous (20 points)

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

Drivers(did, dname, rating, age)
Reserves(did, cid, date)
Cars(cid, cname, color)

```
<table>
<thead>
<tr>
<th>did</th>
<th>dname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bob</td>
<td>5</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td>Abby</td>
<td>3</td>
<td>null</td>
</tr>
<tr>
<td>2</td>
<td>Sally</td>
<td>9</td>
<td>22</td>
</tr>
<tr>
<td>5</td>
<td>Joe</td>
<td>null</td>
<td>35</td>
</tr>
<tr>
<td>3</td>
<td>Zack</td>
<td>10</td>
<td>19</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>did</th>
<th>cid</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>2017-03-15</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2018-01-01</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>2017-04-15</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2017-12-25</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>cid</th>
<th>cname</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Porsche</td>
<td>yellow</td>
</tr>
<tr>
<td>1</td>
<td>Tesla</td>
<td>red</td>
</tr>
<tr>
<td>3</td>
<td>Mercedez</td>
<td>green</td>
</tr>
<tr>
<td>4</td>
<td>BMW</td>
<td>green</td>
</tr>
<tr>
<td>2</td>
<td>Mustang</td>
<td>yellow</td>
</tr>
<tr>
<td>6</td>
<td>Land Rover</td>
<td>red</td>
</tr>
</tbody>
</table>
```

Write the following queries in the indicated language:

(10 pts) **Relational Algebra:**
Print the cids of cars that are reserved by a driver with a rating of at least 5 as well as a driver named Abby.

(10 pts) **SQL:**
For each driver, print their did, name, rating, and – if they have any red cars reserved – the reservation details (date, cid, and cname).

SCORE: _________
Question 4: Never Ending SQL (20 points)

And again! Just when you thought it was safe. For this problem, you are to assume that the Reserves table has been created with did and cid each being defined as FOREIGN KEYs with the ON DELETE CASCADE option for their respective tables (Drivers and Cars). Note that a new table has just been added to the mix – CarPrefs – which has no primary or foreign keys. CarPrefs is initially empty and will soon be used by the club to assist them in advertising cars and other services to their club members.

Drivers(did, dname, rating, age) Reserves(did, cid, date) Cars(cid, cname, color) CarPrefs(did, date, cname, color)

<table>
<thead>
<tr>
<th>did</th>
<th>did</th>
<th>cid</th>
<th>date</th>
<th>Cars</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2017-03-15</td>
<td>Porsche</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>2018-01-01</td>
<td>Tesla</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>2017-04-15</td>
<td>Mercedez</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td>2017-04-15</td>
<td>Mustang</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>1</td>
<td>2017-12-25</td>
<td>Land Rover</td>
</tr>
</tbody>
</table>

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

```
DELIMITER $$
CREATE TRIGGER TrackActivity
AFTER INSERT ON Reserves FOR EACH ROW
BEGIN
  INSERT INTO CarPrefs
  (did, date, cname, color)
  VALUES
  (NEW.did, NEW.date, (SELECT cname FROM Cars WHERE cid = NEW.cid),
   (SELECT color FROM Cars WHERE cid = NEW.cid));
END$$
```

(a) (10 pts) Show below the result of running the following statement against the initial four tables above. (i.e., show the contents of the tables after execution. You may skip any tables that are unaffected.)

```
CALL DropCar(1);
```

SCORE: _________
Question 4: SQL Never Ending SQL (continued)

Here’s all of the database information once again. You are to assume that Reserves table was created with did and cid each being defined as FOREIGN KEYs with the ON DELETE CASCADE option for their respective tables (Drivers and Cars). Note the new table – CarPrefs – which has no primary or foreign keys. This new table, which they just added, is currently empty and will be used by the club to assist them in advertising cars and other services to their club members.

Drivers(did, dname, rating, age)  Reserves(did, cid, date)  Cars(cid, cname, color)
CarPrefs(did, date, cname, color)

<table>
<thead>
<tr>
<th>Driver</th>
<th>Reserves</th>
<th>Cars</th>
</tr>
</thead>
<tbody>
<tr>
<td>did</td>
<td>cid</td>
<td>cid</td>
</tr>
<tr>
<td>Bob</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Abby</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sally</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joe</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>null</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zack</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DELIMITER $$
CREATE TRIGGER TrackActivity
CREATE PROCEDURE DropCar(oldcar INT(11))
BEGIN
    DELETE FROM Cars WHERE cid = oldcar;
END $$
CREATE TRIGGER TrackActivity
CREATE PROCEDURE DropCar(oldcar INT(11))
AFTER INSERT ON Reserves FOR EACH ROW
BEGIN
    INSERT INTO CarPrefs(did, date, cname, color)
    VALUES( NEW.did, NEW.date,
            (SELECT cname FROM Cars WHERE cid = NEW.cid),
            (SELECT color FROM Cars WHERE cid = NEW.cid));
END; $$
DELIMITER ;

(b) (5 pts) Show below the result of running the following statement starting from the initial four tables above. (I.e., show the contents of the tables after execution. You may skip any tables that are unaffected.)

```
INSERT INTO Reserves(did, cid, date) VALUES (3,2,'2018-05-30'), (5,4,'2018-05-31');
```

(c) (5 pts) Show below the results of executing the following statement against the initial tables above.

```
CREATE VIEW AgeSummary(age, numdrivers) AS
    SELECT age, COUNT(*) as numdrivers FROM Drivers GROUP BY age;
SELECT * FROM AgeSummary A WHERE A.age >= 21;
```
Question 5: 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 $N_R > N_S > 0$, what are the minimum and maximum possible result cardinalities for the following relational algebra queries expressed in terms of $N_R$ and $N_S$?

- $S - R$: min size: \hspace{1cm} max size:
- $R \bowtie S$: min size: \hspace{1cm} max size:
- $\pi _A R$: min size: \hspace{1cm} max size:

(b) (4 points) Suppose that the table Products(sku, pname, price, shipping, tax) contains three tuples:

- (123, ‘Widget’, NULL, NULL, 1.10)
- (456, ‘Gadget’, 50.00, 3.00, 1.10)
- (789, ‘Whatzit’, 100.00, 5.00, NULL)

For each of the following SQL queries, show the output that it would produce if run against Products:

(i) (2 pts) SELECT sku, (price * tax) + shipping AS totalcost FROM Products;

(ii) (2 pts) SELECT SUM(price + shipping) AS taxfreecost FROM Products;

SCORE: _________