NAME:

There are 110 points on this exam. Look over the entire exam before starting and make sure you have 8 pages. Use your time accordingly with the number of points, i.e., 1 minute per point.

A) SQL and Relational Algebra (55 points)

You are to use the following relational database schema, primary keys are indicated in bold face.

\[
\begin{align*}
\text{ProductInfo} & \quad \text{(model, maker, type)} \\
\text{PC} & \quad \text{(model, speed, ram, numGB, typeRemovable, price)} \\
\text{Laptop} & \quad \text{(model, speed, ram, numGB, typeRemovable, screenSize, price)} \\
\text{Printer} & \quad \text{(model, color, type, price)}
\end{align*}
\]

First, assume that model is a unique id for all computer products over all manufactures. Each of the attributes is explained as follows:

- Product.model: The number of the computer product
- Product.maker: The manufacturer of the given product
- Product.type: The type of the product (PC, Laptop, or Printer)
- PC.model: The number of the computer product
- PC.speed: The speed (in Mhz) of the processor
- PC.ram: The amount of memory (in MB)
- PC.numGB: The disk size (in GB)
- PC.typeRemovable: Type of removable drive (CD or DVD)
- PC.price: Price of the product
- Laptop.model: The number of the computer product
- Laptop.speed: The speed (in Mhz) of the processor
- Laptop.ram: The amount of memory (in MB)
- Laptop.numGB: The disk size (in GB)
- Laptop.typeRemovable: Type of removable drive (CD or DVD)
- Laptop.price: Price of the product
- Laptop.screenSize: Size of screen (in inches)
- Printer.model: The number of the computer product
- Printer.color: Indicator if color, 1 equals color, 0 equals not
- Printer.type: Laser or inkjet
- Printer.price: Price of the product
Answer the following four query questions.

1. (8 points) Answer using relational algebra

Find all manufacturers who make pcs and laptops but no printers.

Solution.

\[
\left( \pi_{\text{maker}}(\sigma_{\text{type}="\text{PC"}}(\text{ProductInfo})) \cap \pi_{\text{maker}}(\sigma_{\text{type}="\text{Laptop"}}(\text{ProductInfo})) \right) - \pi_{\text{maker}}(\sigma_{\text{type}="\text{Printer"}}(\text{ProductInfo})).
\]

2. (8 points) Answer using relational algebra

Find all manufacturers who make at least 2 models of pcs but no laptops.

Solution.

\[
\text{MakersOfTwoModels} = \pi_{\text{maker1}}(\sigma_{\text{maker1} = \text{maker2} \land \text{type}="\text{PC"}} \land \text{model1} \neq \text{model2})(\text{ProductInfo} \times \text{ProductInfo}).
\]

\[
\text{MakersOfTwoModels} - \pi_{\text{maker}}(\sigma_{\text{type}="\text{Laptop"}}(\text{ProductInfo})).
\]

In the relational algebra queries you could have done joins of ProductInfo with the corresponding tables on Laptops, PCs, and Printers, instead of sigmas on the “type” attribute of ProductInfo. While this is less efficient, it is another correct (and also longer) way of writing these queries and were not deemed wrong. Also, in the query above, I did not use a renaming operator explicitly assuming the meaning is clear from context - again, as I mentioned before, the explicit use of the renaming operator is not something I require.

3. (8 points) Answer using SQL

For each manufacturer who makes at least three models of laptops (different model numbers), list the maximum, average, and minimum laptop price.

Solution.

\[
\text{SELECT MIN(L.price), MAX(L.price), AVG(L.price) FROM ProductInfo P, Laptops L WHERE P.model = L.model GROUP BY P.maker HAVING COUNT(*) >= 3.}
\]

Here, it is clear how COUNT(*) is used since model is the primary key for all tables and hence each tuple in ProductInfo joins with exactly one tuple in Laptops. Recall that aggregates are computed separately (and once) for each group.
4. **(9 points)** Answer using **SQL**

Find the model number and maker of the lowest priced PC that has 512MB or more memory.

**Solution.**

```sql
SELECT P.maker, C.model
FROM ProductInfo P, PC C
WHERE P.model = C.model AND C.ram >= 512
AND C.price = (SELECT MIN(C1.price)
   FROM PC C1
   WHERE C1.ram > 512)
```

5. **(12 points)** Answer using **SQL**

Find the model number and maker of the lowest priced PC that is more expensive than some printer but less expensive than all laptops (that is the PC that is priced lowest among the PCs in this category).

**Hint:** Try using nested queries and named intermediate relations.

**Solution.**

```sql
SELECT Temp.price, Temp.maker, Temp.model
FROM (SELECT PI.maker, C.model, C.price
   FROM PC C, ProductInfo PI
   WHERE C.model = PI.model AND
   C.price > ANY (SELECT P.price FROM Printer P) AND
   C.price < ALL (SELECT L.price FROM Laptop L) ) AS Temp
WHERE Temp.price = (SELECT MIN(price) FROM Temp)
```

A number of variations are possible. For example, I used ANY and ALL operators with Printer and Laptop subqueries when I could've used MIN and MAX instead (with slight additional changes to those lines).

6. **(10 points)** Answer using **SQL**

You are given the following query in tuple-relational calculus. Write an equivalent query in SQL.

\[
\{P | P1 \in \text{Printer} \land (\forall L \in \text{Laptop} ((L.\text{screenSize} = 14) \rightarrow P1.\text{price} > L.\text{price})) \land
   P.\text{model} = P1.\text{model} \land P.\text{type} = P1.\text{type}\}
\]

**Solution.** Find printers that are more expensive than all 14-inch laptops. SQL:

```sql
SELECT P.model, P.type
FROM Printer P
WHERE P.price > ALL (SELECT L.price FROM Laptop L WHERE L.screenSize = 14)
```
B) Functional Dependencies and Normalization (35 points)

7. (a) (6 points) Candidate Keys
Assume schema (A, B, C, D, E, F) with functional dependencies (not necessarily a minimal cover):
A → B, B → C, B → F, BD → A, C → D, and C → E. Are any of the five single attributes (A, B, C, D, E, F) a candidate key? If so which one or ones?
Solution. The keys are A and B.

(b) (7 points) Decomposition
Assume schema (A, B, C, D, E, F) with functional dependencies: A → B, B → C, C → D, B → E, B → F, EF → A, and E → C. Which normal form does this relation belong to? Use the BCNF decomposition algorithm to decompose into a valid BCNF decomposition. Is the decomposition dependency preserving?
Solution. B → C can be removed to make a minimal cover. Then, decompose into ABCEF and CD since the only dependencies violating BCNF are C → D and E → C. Then, decompose into ABEF, EC, and CD. You’re done - it’s dependency preserving (if you started first with EC, then you need to add CD back since you would’ve not preserved E → C. But you don’t need to worry about that - just answer not dependency preserving in that case.
(c) (11 points) Dependency Preserving Decompositions
Assume schema (A, B, C, D, E, F) and functional dependencies: \( AB \to C, AB \to F, A \to D, \\
A \to E, CE \to F, D \to E, E \to B, F \to A \). Compute the minimal cover for this set of dependencies. Then, state which normal form the relation conforms to and explain why. Then, decompose into BCNF. If not dependency preserving, create a 3NF decomposition.

**Solution.** First, since dependencies are already in standard form, get rid of B in AB on the lefthand side. Then, remove \( A \to F \) and \( A \to E \) to make a minimal cover. This is because of \( A \to E, E \to B \). Hence, A is key, so is F, so are CE and CD. The violating dependencies are \( D \to E, E \to B \) (this last one violates 2NF since it’s a partial dependency - it goes to something not part of a key on the righthand side). Hence, it’s in 1NF. Decompose into ACDF, DE, and EB (I started with \( E \to B \)). Now, \( CE \to F \) is not preserved. Add CEF to the decomposition. The result is BCNF and dependency preserving.

(d) (11 points) Dependency Preserving Decompositions
Assume schema (A, B, C, D, E) and function dependencies: \( AC \to BDE, C \to D, C \to E, \\
and \( D \to A \). Compute the minimal cover for this set of dependencies. Then, state which normal form the relation conforms to and explain why. Then, decompose into BCNF. If not dependency preserving, create a 3NF decomposition.

**Solution.** Remove A from AC on the lefthand side because of \( C \to D \) and \( D \to A \). Now, you get two copies of \( C \to D \) and \( C \to E \) as a result. The only dependency causing a problem is \( D \to A \). This is 2NF. Decompose into CBDE and DA. The result is BCNF and dependency preserving.
D) B-trees (15 points)

8. (15 points) Assume max entries per node \( n = 2 \), left subtree is \(<\), right subtree is \( \geq \), and when you split the extra entry goes to the right. Show the tree at each stage when inserting in order 25, 34, 14, 47, 52 followed by deleting 25.

Solution. I’ll let you figure this one out yourself. Refer to the slides. You should get a three-level tree after all insertions and then deleting 25 causes a merge that propagates and results in a two-level tree.
E) Hashing vs B-trees (5 points)

9. (a) (3 points) Give a simple SQL query involving only the Sailors(sid, sname, rating, age) relation for which having a B-tree index on rating would result in a smaller query response time than a hashing index on rating. Assume the attribute rating can take on values between 0 and 10 rounded to the nearest 0.1, i.e. (0.0, 0.1, 0.2, ..., 9.8, 9.9, 10.0). Describe (in a few words) a particular situation with the data distribution when using a hashing index may be particularly inefficient.

Solution. Any query that looks for a range of values on this attribute such as WHERE rating > 5.7 AND rating < 9.4. With a hash index you are no better off than with a simple unsorted file and so will need to do individual look-ups on each value in the range. A B+ tree will go to 5.7 and then walk at the leaf level.

(b) (2 points) Give a simple query for which the hash index would result in a smaller query response.

Solution. Any query with an exact match, such as looking for sailors whose rating is 8.7.
Page left blank for scratch work.