STUDENT NAME:
STUDENT ID:

Write your name and id first. Then answer all questions.

Functional Dependencies and Normalization (14 points)

(1) (14 points) Consider a relation $R$ with five attributes $ABCDE$. You are given the following dependencies: $A \rightarrow B$, $BC \rightarrow E$, and $ED \rightarrow A$.

1. (6 points) List all keys for $R$.

2. (4 points) Is $R$ in 3NF? Explain briefly.

SQL Queries (18 points)

(2) (18 points) Consider the following schema:

Suppliers(sid: integer, sname: string, address: string)
Parts(pid: integer, pname: string, color: string)
Catalog(sid: integer, pid: integer, cost: real)

The Catalog relation lists the prices charged for parts by suppliers. Write the following queries in SQL. Results should not contain duplicates; however, don’t use DISTINCT unless you have to:

1. Find the snames of suppliers who supply every red part.

2. Find the sids of suppliers who supply a red part and a green part.
3. Find the *sids* of suppliers who charge more for some part than the average cost of that part (averaged over all the suppliers who supply that part).

**Sorting and Operator Implementation (33 points)**

(3) **(15 points)** Consider processing the following SQL projection query:

```
SELECT DISTINCT E.title, E.ename FROM Executives E
```

You are given the following information:

- Executives has attributes *ename*, *title*, *dname*, and *address*; all are string fields of the same length.
- The *ename* attribute is a candidate key.
- The relation contains 10,000 pages.
- There are 10 buffer pages.

Consider the optimized version of the sorting-based projection algorithm: The initial sorting pass reads the input relation and creates sorted runs of tuples containing only attributes *ename* and *title*. (Assume that memory is utilized well and that any available optimization to increase run size is used.) Subsequent merging passes eliminate duplicates while merging the initial runs to obtain a single sorted result (as opposed to doing a separate pass to eliminate duplicates from a sorted result containing duplicates). The cost metric is the number of page I/Os.
1. How many sorted runs are produced in the first pass?

2. What is the average length of these runs?

3. What is the I/O cost of this sorting pass?

4. How many additional merge passes will be required to compute the final result of the projection query?

5. What is the I/O cost of these additional passes?
Consider the join $R \bowtie_{R.a = S.b} S$, given the following information about the relations to be joined. The cost metric is the number of page I/Os, and the cost of writing out the result should be uniformly ignored.

Relation $R$ contains 10,000 tuples and has 10 tuples per page.
Relation $S$ contains 2,000 tuples and also has 10 tuples per page.
Attribute $b$ of relation $S$ is the primary key for $S$.
Both relations are stored as simple heap files.
Neither relation has any indexes built on it.
52 buffer pages are available.

1. What is the cost of joining $R$ and $S$ using a page-oriented simple nested loops join?

2. What is the cost of joining $R$ and $S$ using a block nested loops join?

3. What is the cost of joining $R$ and $S$ using a sort-merge join?
Query Optimization and Index Selection (23 points)

(5) (18 points) Consider the following relations: Consider the following relation:

\[ \text{Emp}(eid: \text{integer}, \text{sal: integer}, \text{age: real}, \text{did: integer}) ]

There is a clustered index on \( \text{eid} \).

1. (3 points) Give an example of an update that is definitely speeded up because of the available indexes. (English description is sufficient.)

2. (3 points) Give an example of an update that is definitely slowed down because of the indexes. (English description is sufficient.)

3. Suppose that the following query is very common: For each department, find the \text{eid} of the highest-paid employee.

(a) (4 points) What index would you create to speed up this query?
(b) **(4 points)** Describe how you expect the system to use this index to evaluate the query.

(c) **(4 points)** Suppose that you also have the common query: *For each employee, find the highest salary paid in that employee’s department.* Given the indexes that you already have (the one on *eid* and the one that you created in response to the previous query), describe how the system could best evaluate this new query (assuming that your optimizer was ”perfect” and always found the best plan).

4. **(5 points)** Briefly discuss the clustered index on *eid*. (What kind of queries could the clustering be useful for? Are any of these likely to be common? Would you have made this index clustered?)
Transactions, Concurrency, Recovery (11 points)

(6) (11 points) Give brief answers to the following questions:

1. Define these terms: transaction, schedule, serializability.

2. What is the database log used for? How is the log created/maintained?

Free Point (1 point)

Make sure you wrote your name and id! Happy holidays.