SID:_____

Question 1 –SQL [7 parts, 40 points total]

Consider the following schema for a library. Primary keys are underlined.

```
Author(name, citizenship, birthYear, birthCity)
Book (isbn, title, book_author)
Library (lname, city)
Subject (isbn, subject)
Inventory (isbn, lib_name, edition, quantity)
```

Notes:

- "book_author" in **Book** is a foreign key referencing "name" in **Author**
- "lib_name" in **Inventory** is a foreign key referencing "lname" in **Library**.

Write **SQL** queries for the following. All of these are doable *without creating views or temporary tables*. You may do so if you feel it is necessary but we'll take off some points if you do. We will also take off points for doing unnecessary joins, distincts, etc.:

a) [5 points] For all authors who have written more than 10 books, print their name and the count of the number of books they have written. The list should be output in DESCENDING order of the number of books written.

```
SELECT book_author, count(*)
FROM Book
GROUP BY book_author
HAVING count(*) > 10
ORDER BY count(*) DESC;
```

b) [5 points] Print the titles of all books that are in the inventory of a library located in the city where the author of that book was born. DO NOT PRINT DUPLICATE TITLES.

```
SELECT DISTINCT b.title
FROM Book b, Author a, Library l, Inventory i
WHERE b.book_author = a.name AND
      a.birthCity = l.city AND
      l.lname = i.lib_name AND
      i.isbn = b.isbn;
```

c)[5 points] Consider the following SQL query:

```
select b.isbn, b.title
from book b, inventory i
where b.isbn = i.isbn and i.lib_name = 'Cao Library' and
      i.quantity > ALL (select quantity
                        from inventory
                        where lib_name = 'Evans Library');
```

Explain concisely (in English) what this query produces:

The isbn and title of all books that are present at the 'Cao Library' in greater quantity than any single book at the "Evans Library'.

d)[5 points] Which of the following queries could produce a different result set than the query in part "c"above? **(Circle one answer only)**

a)
```
select b.isbn, b.title
from book b, inventory i
where b.isbn = i.isbn and i.lib_name = 'Cao Library' and
      i.quantity > (select MAX(quantity)
                    from inventory
                    where lib_name = 'Evans Library');
```

b)
```
select b.isbn, b.title
from book b, inventory i
where b.isbn = i.isbn and i.lib_name = 'Cao Library' and
      i.quantity NOT IN (select distinct quantity
                         from inventory
                         where lib_name = 'Evans Library');
```

c)
```
select b.isbn, b.title
from book b, inventory i
where b.isbn = i.isbn and i.lib_name = 'Cao Library' and
      NOT EXISTS (select *
                  from inventory
                  where lib_name = 'Evans Library'
                        and quantity >= i.quantity );
```

d) None of the above.

e) [5 points] Under what condition(s) would the two queries below produce different results?

`SELECT a.name, b.title`
`FROM author a LEFT OUTER JOIN book b ON`
`a.name = b.book_author;`
	`SELECT a.name, b.title`
	`FROM author a, book b`
	`WHERE a.name = b.book_author;`

There are authors in 'a' with no books in 'b'

SID:_____

f) [10 points] For each author who has written about two or more different subjects (in the same or in different books), print his/her name and the titles of all his/her books that can be found in the 'Doe' Library.

Two examples:

```
SELECT B.book_author, B.title
FROM Book B, Inventory I
WHERE I.isbn = B.isbn
AND I.lib_name = "DOE"
AND 2 <= (SELECT COUNT(DISTINCT S.subject)
          FROM Subject S, Book B2
          WHERE S.isbn = B.isbn
          AND B2.book_author = B.book_author);
```

```
SELECT B.book_author, B.title
FROM Book B, Inventory I
WHERE I.isbn = B.isbn
AND I.lib_name = "DOE"
AND B.book_author IN
  (SELECT B1.book_author
   FROM Book B1, Book B2, Subject S1
        Subject S2
   WHERE B1.isbn = S1.isbn
   AND B2.isbn = S2.isbn
   AND S1.subject <> S2.subject
   AND B1.book_author = B2.bookauthor
  );
```

g) [5 points] Consider the following (somewhat ugly) SQL query:

```
select b1.title, b2.title
from Book b1, Book b2, BSubject s11, BSubject s12, BSubject s21, BSubject s22
where b1.author = b2.author
      and b1.isbn = s11.isbn and b1.isbn = s12.isbn
      and s11.subject != s12.subject
      and b2.isbn = s21.isbn and b2.isbn = s22.isbn
      and s21.subject != s22.subject
      and s11.subject = s21.subject and s12.subject = s22.subject
      and b1.isbn > b2.isbn
group by b1.title, b2.title;
```

Complete the following sentence to explain (in English) what this query does:

Print the titles of unique pairs of books that are by the same author and have at least two different subjects in common.

Question 2 –Sorting [4 parts, 15 points total]

Consider a relation containing information about university students:

```
Students (sid: integer, sname: varchar(50), street:
  varchar(50), city: varchar(30), age: integer)
```

Assume that the students file consists of 5,000 pages.

a) (5 points) Consider a simple multi-pass external merge sort algorithm that in the first pass, produces disk-based runs of "B" pages where B is the number of pages of memory available to the algorithm. If B = 50 pages, how many passes (including pass 1) of the algorithm will be required to sort the relation and how many runs will be produced by each of the passes?

Total Number of Passes = 3	**Pass #**	**Number of Runs Produced**
	1	100
	2	3
	3	1

b) (2 points) For part (a), how many I/Os (total of reads and writes) will be required? Assume that the relation is originally on disk and that the result of the sorting operation must also be written to disk.

(3)(2)(5000) = 30,000

c) (3 points) Suppose we use the in memory sort optimization that generates runs of (on average) length 2B during pass 1. If B = 50, how many passes are required (including pass 1) and how many runs will be produced by each of the passes?

Total Number of Passes = 3	**Pass #**	**Number of Runs Produced**
	1	50
	2	2
	3	1

d) (5 points) Now, consider the query "**SELECT sid, age FROM Students ORDER BY age**". **Briefly** describe how could you execute this query in a way that would require less I/O than even the lowest cost of sorting the Students relation above? Does your approach save disk reads, disk writes, or both?

Project out everything except sid and age during the first pass of the sort, this saves both disk reads and disk writes.

SID:_____

Question 3 – Query Optimization [10 parts, 45 points total]

Consider a slightly simplified Library schema from Question 1 with Primary Keys underlined:

```
Author(name, citizenship, birthYear, birthCity)
Book (isbn, title, book_author)
Library (lname, city)
Inventory (isbn, lib_name, edition, quantity)
```

Where:

- "book_author" in **Book** is a foreign key referencing "name" in **Author**
- "lib_name" in **Inventory** is a foreign key referencing "lname" in **Library**.
- "isbn" in **Inventory** is a foreign key referencing "isbn" in **Book**

Now consider the following statistics:

- The cardinalities (NTuples) and sizes (NPages) of the relations are:
  Author: NTuples = 10,000, NPages = 100
  Book: NTuples = 25,000, Npages = 500
  Library: NTuples = 500, Npages= 5
  Inventory: NTuples = 1,000,000, NPages = 10,000
- NKeys(birthCity) for **Author** is 1000 (i.e., there are 1000 distinct values for birthCity)
- NKeys(birthYear) for **Author** is 200
- ILow(birthYear) = 1801, IHigh(birthYear) = 2000
- NKeys(city) for **Library** is 400
- NKeys for the primary keysof each relation = NTuples for that relation

and the following indexes:

- A hash index is defined on the primary key for each relation.
- A *clustered* B+Tree index is defined on the *birthCity* attribute for **Author**
- An *unclustered* B+Tree index is defined on the *birthYear* attribute for **Author**
- An *unclustered* B+Tree index is defined on the *city* attribute for **Library**

a) [3 points] How many records do you estimate will be returned by the following query:

```
SELECT *
FROM Author
WHERE birthYear > 1950;
```

Answer to Part (a): ~2500

$RF = \frac{IHigh - value}{IHigh - ILow} = \frac{2000 - 1950}{2000 - 1801} = \frac{50}{199}$

answer = RF(10,000) ≈ 2500

b) [3 points] Under what condition(s) would a histogram of birthYear allow you to give a more accurate estimate than the answer you gave in part (a)?

If the values of birthYear follow a non-uniform distribution

c) [3 points] How many records do you estimate will be returned by the following query:

```
SELECT *
FROM Author, Book
WHERE name = book_author;
```

Answer to Part (c): 25,000

Recall:
1) name is primary key of Author
2) book-author is foreign key to name

$\min\left(25{,}000 * \frac{10{,}000}{10{,}000},\ 10{,}000 * \frac{25{,}000}{10{,}000}\right) = \min(25{,}000,\ 25000) = 25{,}000$

d) [3 points] How many records do you estimate will be returned by the following query:

```
SELECT *
FROM Author, Library
WHERE birthCity = city;
```

Answer to Part (d): 5000

$\min\left(10{,}000 * \frac{500}{400},\ 500 * \frac{10{,}000}{1000}\right)$
$= \min(12{,}500,\ 5000) = 5000$

e) [3 points] How many records do you estimate will be returned by the following query:

```
SELECT *
FROM Inventory, Library;
```

Answer to Part (e): 500,000,000

Note: join key unspecified
$(1{,}000{,}000) \times (500) = 500{,}000{,}000$

For parts **f,g, and h,** state an *efficient method* for answering this query and state how many disk accesses will have to be made in order to answer the query using this method. Be sure to state which index(es) if any you are using. *State any assumptions you are making. Note, you cannot assume that any of the relations fit in memory.* Part of your grade will depend on how efficient a solution you choose.

f) [5 points]

```
SELECT *
FROM Author
WHERE birthCity = 'Berkeley' and birthYear = 1972;
```

Assume Alternative 1.
First use clustered index on birthCity to find where it is 'Berkeley'.
This results in $10{,}000\left(\frac{1}{1000}\right) = 10$ records, which can fit on one page.
Next scan that page for records where birthYear = 1972.

I/O Analysis:
Assume 2-4 I/Os for B+tree traversal
Index lookup for birthCity: 2-4 I/Os
Scan of page for birthYear: 1 I/O (if not in buffer)
=> we accepted ~2-5 I/Os, or ~12-15 if used different Alternative

Question 3 – Query Optimization (continued)

g) [5 points]
```
SELECT *
FROM Author
WHERE birthCity = 'Berkeley' and name = 'Alan Ginsberg';
```

First use Hash index on name to find 'Alan Ginsberg'.
Since name is primary key, there will be 1 record returned (on 1 page)
Then check that record for birthCity
I/Os: 1-2 for Hash lookup + 1 I/O to get whole record (if Alt. 2 or 3)

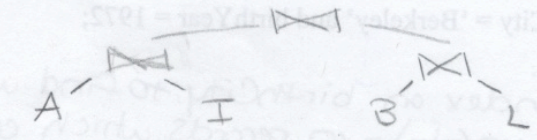
h) [10 points] Assuming there are ten (10) pages of memory available for use in by the join, choose an efficient method for computing the join between **Library** and **Inventory** and estimate the number of disk I/Os it would incur. Be sure to state the join method you have chosen, and which relation is inner or outer if appropriate.

Method: Block nested loop join
NPages(Library) = 5, NPages(Inventory) = 10,000

outer: Library
inner: Inventory
blocksize: 5
cost = $5 + 10{,}000\left(\lceil \frac{5}{8} \rceil\right) = 5 + 10{,}000 = 10{,}005$ I/Os

i) [6 points] Consider the four-way join of the relations in this question by their key/foreign key relationships. Give one join ordering that a System-R optimizer would **NOT** consider, and briefly state why it would not consider it. Use a query plan tree or Relational Algebra (your choice) to answer this question.

Examples:
1) non-left deep plan
2) performing cross-products first



j) [4 points] Write a SQL SELECT statement that could take advantage of a B+Tree index on a composite key (isbn, lib_name) of **Inventory** if one were defined, but could not use the existing Hash index on that key.

Examples:
1) range query using isbn

WHERE isbn > 42

2) equality query using only isbn

WHERE isbn = 42