Name: ___________________________________

Class Account:______________________________

UNIVERSITY OF CALIFORNIA
Department of EECS, Computer Science Division

CS186 Hellerstein

Fall 2007 Midterm Exam

Midterm Exam: Introduction to Database Systems

This exam has five problems, worth 20 points each. Each problem is made up of multiple questions. You should read through the exam quickly and plan your time-management accordingly. Before beginning to answer a problem, be sure to read it carefully and to *answer all parts of every problem!*

You **must** write your answers on the exam, in the spaces provided. ***Do not tear pages off of your exam!***

Good luck!

1. Sorting

Zombo.com has begun advertising “sorting as a service” for their website. Users can upload a table of numbers to Zombo.com, and the site will sort it for them and email it back. Unfortunately, after launching their intergalactic advertising campaign, the executives at Zombo.com realize that they have no idea how to deal with large tables. They hire you as a consultant to work this out for them.

a) **[6 points]** Zombo.com sorts one file at a time. You promise the executives that the I/O count for sorting the file (including reading it from disk and writing the result to disk) will be **4 times** that of simply reading it. To guarantee this, you suggest introducing a limit on the size of file uploads to some number **maxsize** bytes per file. The Zombo.com servers use 8K disk blocks, and have 128Kbytes of memory available for sorting. What value should you recommend for **maxsize** (in Kb!)?

(16*15 pages)*8KB/page
= 1920 KB

b) **[6 points]** The executives are unhappy with the idea of size limits. They remind you that “the unattainable is unknown at Zombo.com!” They ask you how many **disk I/Os** it would take to sort a file that is one Yottabyte big on a single server. Please state your answer as a function of the value Y (where Y = 1 Yottabyte).

$$2(Y/8K) * (1 + \log_{15}(Y/(8K*16)) \text{ I/Os}$$

3 points for using a log of any kind.
1 point for log base 15.
1 point for the 2Y factor.
1 point for dividing by 8K and 16 appropriately.

c) **[8 points]** A frequent user of the service uploads 256KB files that are often already sorted. Assume that the output must be written on the same disk as the input, but that you have another disk available to you for scratch space. Assume also that you choose to use QuickSort in memory. Fill in the following table describing the I/O behavior when they upload a file that happens to already be sorted:

	# of Random I/Os	**# of Sequential I/Os**
Pass 0 Read	1 (or 0)	31 (or 32)
Pass 0 Write	1 (or 0)	31 (or 32)
Pass 1 Read	3 (or 4)	29 (or 28)
Pass 1 Write	1	31

Sequential Mostly: 4 points
32 Blocks/Row: 1 points
Pass 1 Read: 2 points
Pass 1 Write: 1 point

2. Query Languages

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

```
Author(name, citizenship, birthYear, birthPlace)
Book (isbn, title ,author)
Library (lname, city)
Bindex (isbn, subject)
In_stock (isbn, lib_name, edition, quantity)
```

Based on the above schema, try to answer the following questions:

a) **[4 points]** Convert this SQL query to relational algebra:

```
SELECT *
  FROM Author a, Book b
 WHERE a.name = b.author
   AND b.title = 'Hopscotch';
```

Your answer:

($\sigma_{\text{ttitle='Hopscotch'}}$ Book) \join Author

```
-1 if no selection
-1 if projection was specified on the wrong set of fields
-1 if cross-product without selection predicate to account for the
join between Books and Authors
```

b) **[6 points]** Consider the following SQL query:

```
SELECT a.name
FROM author a
WHERE NOT EXISTS (SELECT b.isbn
                  FROM book b
                  WHERE b.author = a.name AND NOT EXISTS (
                     SELECT *
                     FROM in_stock i
                     WHERE i.isbn = b.isbn AND i.lib_name = ‘Evans’ AND
                            i.quantity > 2)) ;
```

Fill in the blanks in the following relational calculus query that makes it equivalent to the SQL above,. Note: the correct answer may not require all blanks to be filled in!

$\{R \mid \underline{\ \exists\ }\ A \in \text{Authors} \ (\ A.\text{name} = R.\text{name} \wedge$

$\forall\ B \in \text{Book} \ (\ (\ B.\text{author} = A.\text{name}\ )$

$\underline{\ \ \rightarrow\ \ }(\exists\ I \in \text{InStock} \ (\ \ I.\text{isbn} = B.\text{isbn}\ \underline{\ \wedge\ \ \ }$

$I.\text{lib\_name}=\text{'Evans'}\ \underline{\ \wedge\ \ \ }$

$I.\text{quantity} > 2\ ))))\ \}$

Eirinaios Michelakis 11/6/07
1:02 AM
-2

Eirinaios Michelakis 11/6/07
1:02 AM
-2

Eirinaios Michelakis 11/6/07
1:02 AM
-1

Eirinaios Michelakis 11/6/07
1:02 AM
-1

c) **[5 points]** The following SQL query is given:

```
SELECT b.title, s.edition
FROM book b, in_stock s
WHERE b.isbn = s.isbn and s.lib_name = 'Cao Library' and
        NOT EXISTS (SELECT *
                    FROM in_stock i
                    WHERE i.lib_name = 'Evans Library'
                          and i.quantity > s.quantity );
```

Which of the following queries will produce a **different** result set?

```
i) SELECT b.title, s.edition
   FROM book b, in_stock s
   WHERE b.isbn = s.isbn and s.lib_name = 'Cao Library' and
         s.quantity NOT IN (SELECT distinct quantity
                            FROM in_stock i
                            WHERE i.lib_name = 'Evans Library');
```

```
ii) SELECT b.title, s.edition
   FROM book b, in_stock s
   WHERE b.isbn = s.isbn and s.lib_name = 'Cao Library' and
         s.quantity > (SELECT MAX(quantity)
                       FROM in_stock i
                       WHERE i.lib_name = 'Evans Library');
```

```
iii) SELECT b.title, s.edition
   FROM book b, in_stock s
   WHERE b.isbn = s.isbn and s.lib_name = 'Cao Library' and
         s.quantity > ALL (SELECT quantity
                           FROM in_stock i
                           WHERE i.lib_name = 'Evans Library');
```

iv) None of the above.

Your answer:
i

i : 5/5
i,ii,iii: 1.75/5
i,ii or i,iii: 3.25/5
ii: 1.75/5
iii: 1.75/5

d)**[5 points]** Given the following Relational Calculus statement:

$$\{B \mid \exists B \in Books(\exists A1 \in Authors(A1.birthYear > 1920) \wedge$$
$$(\exists A2 \in Authors(A2.birthYear > 1920 \wedge$$
$$A2.birthYear > A1.birthYear \wedge$$
$$B.author = A2.name))\}$$

circle the Relational Algebra expressions that compute the same result.

i) $\rho(A1, \sigma_{birthYear > 1920} Author)$
$\rho(A2(name2, c, birthYear, bp), \sigma_{birthYear > 1920} Author)$
$\pi_{isbn, title ,author} (Books \bowtie_{author=name2} (\sigma_{A2.birthYear > A1.birthYear}(A1 \times A2)))$

ii) $Books - (\quad (Books \bowtie \sigma_{birthYear < 1921} Author)$
$\cup (Books \bowtie \sigma_{birthYear < 1920} Author))$

iii) All the above

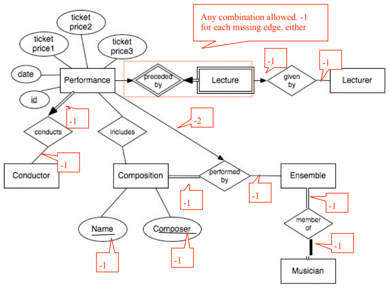
iv) None of the above

Your answer:
i

i: 5/5

3. Entity-Relationship Modeling

a. The following ER diagram models a database that might be used by Zellerbach Hall to store information about the performances it hosts. The diagram contains all the entity sets, their attributes, and relationship sets you need to consider, but is missing lines connecting the relationships with the corresponding entities, and is missing key constraints. Also note that some entities *might be* weak entities, although if there are any, they are not (yet) drawn accordingly. Finally, be sure to clearly differentiate any **THICK** lines from thin ones!



Express the following constraints, **by drawing on the above picture**:

i) [2 points] A composition is identified by both a name (e.g. Symphony in C major) and a composer (e.g. Mozart).

ii) [4 points] A *performance* may be preceded by a single *lecture* analyzing the works that are going to be performed. A lecture must be given by one *lecturer*.

iii) [2 points] A *performance* is always conducted by one *conductor*.

iv) [6 points] A *performance* includes a (positive) number of *compositions*. For a particular *performance*, each *composition* is performed by one *ensemble* (orchestra). An *ensemble* is comprised of a positive number of *musicians*. Every *musician* may participate in one *ensemble*.

b. **[2 points]** According to the above ER diagram, Zellerbach Hall is divided by default in three seating sections, each of which is associated with a particular price. The management decides to introduce a more flexible pricing policy, according to which the number of seating sections (and their pricing) may vary. What changes would you recommend to the above ER diagram to accommodate the management's wish to make more money?

Any solution capturing the need to create a separate entity set for the seating sections, which was associated with the Performance entity set via some relationship set, was given full score.

c. **[4 points]** The following DDL SQL statement creates the table to store the "includes" relationship from the original ER diagram. Fill in the missing details, so that it captures the constraints that the ER diagram represents.

```
CREATE TABLE includes (
   id INTEGER,
   name VARCHAR(20),
   composer VARCHAR(20) -1


   PRIMARY KEY (id, name, composer      ), -1
   FOREIGN KEY(id) REFERENCES Performance -1
   FOREIGN KEY(name, composer) REFERENCES Composition -1

);
```

4. Buffer Management and Spatial Indexing

Suppose the following sequence of calls is presented to the Buffer Manager of a database:

1. get(1);
2. get(7);
3. pin(7);
4. get(3);
5. pin(3);
6. get(4);
7. get(5);
8. get(1);
9. get(4);
10. unpin(7);
11. get(3);
12. get(6);
13. pin(6);
14. get(2);
15. get(1);
16. get(1);
17. unpin(3);
18. get(2);
19. get(6);
20. get(2);
21. get(7);

The calls above have the following behavior:

- `get(RID)`: fetches the record identified by `RID` from the buffer, potentially retrieving it from disk as well if it is not already in the buffer.
- `pin(RID)`: ensures that the record `RID` stays in the buffer.
- `unpin(RID)`: permits the record `RID` to be evicted from the buffer.

Additionally, assume the following:

- The Buffer Manager has 4 buffers A, B, C and D and they are all initially empty.
- The calls `pin(RID)` and `unpin(RID)` cause the Buffer Manager to access the buffer containing `RID`.
- If there are multiple free buffers for Buffer Manager to choose to assign to an `RID`, Buffer Manager chooses the first free buffer alphabetically (for example, A before B if both are free).

For each of LRU, MRU and CLOCK, please indicate the **final buffer contents** and the **number of buffer misses** that occur in the table provided below:

	Final buffer contents (RIDs)	**# of buffer misses**
LRU	A B C D 2 7 3 6	11
MRU	A B C D 1 6 7 5	11
CLOCK	A B C D 1 or 7 7 or 1 3 6	11

1 point for each of the boxes A,B,C,D.
2 points for each correct no. buffer misses.
2 points free.
Note that CLOCK has two possible correct entries for A,B depending upon how clock hand is moved.

Name______________________________________

Text Search
Suppose we were to build an inverted index on the following set of documents using a B+ tree:

ford.html
mustang expedition focus fonefiftey

honda.html
accord civic

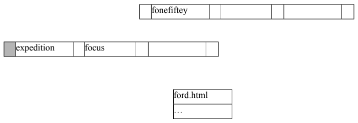
Additionally, assume that:

- The index is constructed by sequentially adding documents one at a time in this order (no bulk loading):
  (1)ford.html, (2) toyota.html, (3) honda.html , (4) wanted.html

- When considering x.html, its contents are considered sequentially. For example, for ford.html, the words are added one at a time in this order (no bulk loading):
  (a) mustang, (b) expedition, (c) focus, (d) fonefiftey

- The B+ Tree structure follows Alternative 2 where data entries are <key, rid> pairs. Note that each document is 1 block.

After ford.html's contents are added, the inverted index looks like this:



a) **[8 points]** Fill in the following table:

After processing html page:	Contents of root index node	Maximum number of keys that can be inserted without splitting any nodes
ford.html	fonefiftey	2
toyota.html	expedition fonefiftey prius	4
honda.html	fonefiftey	5
wanted.html	fonefiftey	2

b) **[4 points]** After **toyota.html** is processed, how many disk blocks are accessed to answer each of the following queries? Assume a buffer size big enough to hold 1000 nodes, and that the buffer is empty at the start of each query.

Keywords in query	Number of nodes accessed
fonefiftey	3
prius	3

a) **[4 points]** After **wanted.html** is processed, how many disk blocks are accessed to answer each of the following queries? Assume a buffer size big enough to hold 1000 nodes, and that the buffer is empty at the start of each query.

Keywords in query	Number of nodes accessed
eclass	3

toyota.html
camry prius corolla tacoma

impala AND fonefiftey	5

2 points (all or nothing) for each correct root block.
1 point for max num keys insertable.
2 points for each keyword query.
2 points free

wanted.html
impala corolla zfour

▨	fonefiftey		mustang			