Spring 2013
CS 122C & CS 222
Midterm Exam (and Comprehensive Exam, Part I)
(Max. Points: 100)

Instructions:
- This exam is closed book and closed notes but open “cheat sheet”.
- The total time for the exam is 80 minutes, so budget your time accordingly.
- Be sure to answer each part of each question after reading the whole question carefully.
- If you don’t understand something, you may ask for clarification.
- If you still find ambiguities in a question, write down the interpretation you are taking and then answer the question based on that interpretation.
- The last two page of this exam is blank; you can use it as scratch paper.

STUDENT NAME:IMA WRIGHT
STUDENT ID: #1

<table>
<thead>
<tr>
<th>QUESTION</th>
<th>POINTS</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>TOTAL</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>
Question 1: Storage Formats (15 points)

Consider the problem of persistently storing a table with the following schema for keeping track of the employees in a company:

Employees (empId: Int, name: VARCHAR(40) nullable, salary: Float, age: Int nullable)

Consider the following sample tuples:

(100, Carey, 100000.00, 55)
(200, Venkatasubramanian, 115000.00, )
(300, Li, 87000.00, 35)

(a) (5 points) Draw a picture of what your CS122C/CS222 instructor’s record would look like if it was represented using the API data format from Part 2 of this term’s course project – i.e., when being passed into or out of the routines that make up the RM layer. Note: Do not use any arrows in your pictures – instead, show what the actual bits in memory would be to represent them! (Assume that integers are 32 bits long, floating point numbers are 32 bits long, and individual characters are each 8 bits long.) To get full credit your picture must be very clear about what each of the components of your formatted record are and also what their respective sizes are.

- 21 bytes
- 100
- 5
- Carey
- 100000.00
- 55

(b) (4 points) Identify two significant disadvantages to the format in (a) that would lead you to not want to use it as the actual on-page data format in a heap file; explain each one very briefly.

i. Can't randomly access fields

ii. Doesn't handle nulls well

(c) (6 points) Draw a picture of a representation of this same record using an improved format that overcomes the disadvantages you identified in (b) and would therefore be a good candidate for the on-page data format for storing records on disk in a DBMS.

SCORE: 15
Question 2: Tree Structured Indexes (25 points)

Consider the following B+ tree index:

(a) (2 point) What is the order \( d \) of this B+ tree?

(b) (6 points) Show the B+ tree index that would result from deleting the entry whose key value is 303.

(c) (6 points) Show the B+ tree index that would result from deleting the entry with key value 54 from the original index.

SCORE: 14
Question 2: Tree Structured Indexes (cont.)

Consider the following B+ tree index:

(d) (6 points) Show the B+ tree index that would result from inserting an entry with key value 279 into the original index.

(e) (5 points) For any two-level B+ tree index of the same order \( d \) as this example, what are the odds of an insert operation causing it to become a three-level index? (Consider the various possible states of such an index; you can figure it out!) Circle your answer below and then very briefly explain your choice.

1/100 1/2 1/3 1/4 1/9 1/16

Reason:

\[
\text{Rough answer: } \quad \text{Prob (full leaf)} \times \text{Prob (full root)} \\
= \frac{1}{3} \times \frac{1}{3} = \frac{1}{9}
\]

\[
\text{Right answer: } \quad \frac{1}{4} \times \frac{1}{3} = \frac{1}{12}
\]
Question 3: Dynamic Hashed Files (25 points)

Consider storing a set of Professor records in a key-value store implemented as a dynamic hashed index. Here are the keys for an initially small data set along with the base 2 and base 10 results from applying a handy open-source hash function called $h$ that you have been given to hash those keys.

<table>
<thead>
<tr>
<th>Key (k)</th>
<th>$h(k)$ base 2</th>
<th>$h(k)$ base 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alex</td>
<td>11110000</td>
<td>240</td>
</tr>
<tr>
<td>Chen</td>
<td>00011111</td>
<td>31</td>
</tr>
<tr>
<td>Mike</td>
<td>11111110</td>
<td>254</td>
</tr>
<tr>
<td>Ramesh</td>
<td>01010101</td>
<td>85</td>
</tr>
<tr>
<td>Rina</td>
<td>00000011</td>
<td>3</td>
</tr>
<tr>
<td>Natini</td>
<td>11100111</td>
<td>231</td>
</tr>
<tr>
<td>Pierre</td>
<td>00100100</td>
<td>36</td>
</tr>
</tbody>
</table>

First consider using extendible hashing as the approach for organizing the resulting file. Assume that the bits of $h(K)$ are used in most-to-least order of significance, like in lecture, and that only two records will fit on a page. The figure below indicates the presence of records on a leaf page of the index by simply listing the keys of those records.

(a) (6 points) Show the resulting file structure if Pierre’s record is inserted into this extendible hashed file.

SCORE: 6
Question 3: Dynamic Hashed Files (cont.)

<table>
<thead>
<tr>
<th>Key (k)</th>
<th>h(k) base 2</th>
<th>h(k) base 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alex</td>
<td>11110000</td>
<td>240</td>
</tr>
<tr>
<td>Chen</td>
<td>00011111</td>
<td>31</td>
</tr>
<tr>
<td>Mike</td>
<td>11111110</td>
<td>254</td>
</tr>
<tr>
<td>Ramesh</td>
<td>01010101</td>
<td>85</td>
</tr>
<tr>
<td>Rina</td>
<td>00000011</td>
<td>3</td>
</tr>
<tr>
<td>Nalini</td>
<td>11100111</td>
<td>231</td>
</tr>
<tr>
<td>Pierre</td>
<td>00100100</td>
<td>36</td>
</tr>
</tbody>
</table>

(b) (6 points) Show what the resulting file structure would be if Nalini’s record is inserted into the original extendible hashed file.
Question 3: Dynamic Hashed Files (cont.)

<table>
<thead>
<tr>
<th>Key (k)</th>
<th>h(k) base 2</th>
<th>h(k) base 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alex</td>
<td>11100000</td>
<td>240</td>
</tr>
<tr>
<td>Chen</td>
<td>00011111</td>
<td>31</td>
</tr>
<tr>
<td>Mike</td>
<td>11111110</td>
<td>254</td>
</tr>
<tr>
<td>Ramesh</td>
<td>01010101</td>
<td>85</td>
</tr>
<tr>
<td>Rina</td>
<td>00000011</td>
<td>3</td>
</tr>
<tr>
<td>Nalini</td>
<td>11100111</td>
<td>231</td>
</tr>
<tr>
<td>Pierre</td>
<td>00100100</td>
<td>36</td>
</tr>
</tbody>
</table>

Now consider the following linear hashed file with the indicated initial file parameters and current state. Records on pages of the file are represented in the picture by simply listing their key values. Use the same open-source hash function as before as the basis of the linear hashed file’s hash function sequence.

\[ h_0(k) = h(k) \mod N \]

\[ N = 2, \ b = 2 \]

\[ \text{level} = 1, \ \text{next} = 1 \]

(c) (6 points) Show the state of the file after inserting Pierre’s record. Assume an uncontrolled splitting policy is used, so whenever a new overflow page is added a split is to be triggered. (Be sure to include both level and next in your picture.)

(d) (6 points) Show what the state of the file would be after instead inserting Nalini’s record into the original linear hashed file structure.
Question 4: Short Answer Questions (20 points)

Very briefly answer each of the following questions.

(a) (2 points) What is the key characteristic of a dense (vs. a sparse) index?

1. It has one entry per indexed record.

(b) (4 points) For which of the following database data access patterns does the LRU buffer replacement policy used in many operating systems work well? (Circle all that apply.)

i. Sequential access to blocks that will not be rereferenced (e.g., a table scan);
ii. Sequential access to blocks that will be cyclically rereferenced (e.g., during a join);
iii. Random access to blocks that will not be referenced again;
iv. Random access to blocks for which there is a nonzero probability of rereference;

(c) (4 points) What is the file system abstraction provided by most operating systems and why is it often cited as being inappropriate for building the storage layer of a DBMS on top of?

File = sequence of bytes. DBMS want blocks of data to manage as files or records, indices, etc. OS files are too ordered and have too high an overhead as a result.

(d) (2 points) System R pioneered the idea and one of the earliest implementations of statistics-driven, cost-based relational query optimization. (Circle the correct answer.)

true

false

(e) (2 points) Linear hashing is superior to B+ trees in terms of the average-case exact-match search cost. (Circle the correct answer.)

true

false

(f) (2 points) Linear hashing is superior to B+ trees in terms of the worst-case exact-match search cost. (Circle the correct answer.)

true

false

(g) (2 points) The recovery subsystem of System R used a combination of the following techniques and/or technologies. (Circle all that apply.)

shadow paging

flash memory

logging

distributed shared memory

SCORE: 20
Question 5: File Organization Performance Analysis (15 points)

Consider a table that contains 1,000,000 user records for a new, up-and-coming social network startup company. Suppose that disk pages are 8KB long and that the average user record (including its internal headers and such) is 200 bytes long; this means that at most 40 records will fit on a page. Assume that the primary key for this table is its user id column and that up to 400 key-pointer pairs will fit on one page. Finally, suppose that a random read costs 10 msec and that a sequential read costs 2 msec.

(a) (3 points) What is the expected I/O cost for a successful user id search if this table is simply stored as an unindexed heap file?

\[
\frac{1,000,000 \text{ REC}s}{40 \text{ REC}s/\text{PAGe}} = 25,000 \text{ PAGes} \quad \text{and} \quad \frac{25,000}{400} = 63 \text{ PAGES AT THE 3-LEVEL TREE LEVEL (3-LEVEL_TREE, all)}
\]

12,500 PAGES + 0.102 SEC/PAGE = 25 SEC

(b) (4 points) What is the expected I/O cost for a successful user id search if this table is instead stored as a sorted file?

\[
\log_2 (25,000) + 10 \text{ MSEC} \approx 150 \text{ MSEC}
\]

(c) (4 points) What is the expected I/O cost for a successful user id search if this table is instead stored as a B+ tree containing the data records in its leaf pages? (Assume that the buffer pool contains no pages of the file at the start of the search.)

3-LEVEL TREE (SEE ABOVE) SO:

2 * 10 MSEC = 20 MSEC

(d) (4 points) What is the expected I/O cost for a successful user id search if this table is instead stored as an extendible hashed file with records in its leaf buckets? (Do not assume the directory is already in memory, and again assume a cold buffer pool at the outset.)

MUST READ 1 DIRECTORIE PAGE (WHOSE LOCATION IS COMPARABLE)

PLUS ONE LEAF BUCKET SO:

2 * 10 MSEC = 20 MSEC

SCORE: 15