Name_________________________________
Section You Usually Attend________________

UNIVERSITY OF CALIFORNIA
College of Engineering
Department of EECS, Computer Science Division

Midterm Exam: Introduction to Database Systems

This exam has four problems, each worth 25 points. There is also 10 points of extra credit available. 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 question, be sure to read it carefully and to *answer all parts of every question!*

You **must** write your answers on the **answer sheets** provided. You may use the sheets with questions as scratch paper. You also must **write your name** at the top of **every page**, and you must turn in all the pages of the exam. **Do not** remove pages from the stapled exam! Two pages of extra answer space have been provided at the back in case you run out of space while answering. If you run out of space, be sure to make a "forward reference" to the page number where your answer continues.

Good luck!

1. Sorting [25 points]

To receive full credit for this problem you ***must*** *write down relevant formulas and show your work.*

We are trying to sort a file containing 250,000 blocks with only 250 buffers available in the buffer pool for sorting. The file contains fixed length tuples.

a) [5 points] How many initial runs will be generated if quicksort is used to sort tuples in memory?

b) [5 points] How many total I/O will the sort perform, *including* the cost of writing out the output?

In the following questions, assume that the average time to locate a random block (seek and rotational delay) takes 10 ms, while the time to transfer a block takes 5 ms.

c) [5 points] During our sort, if we merge runs a block at a time (in memory we set aside one buffer frame per input run for merging) what percent of the total disk bandwidth are we utilizing?

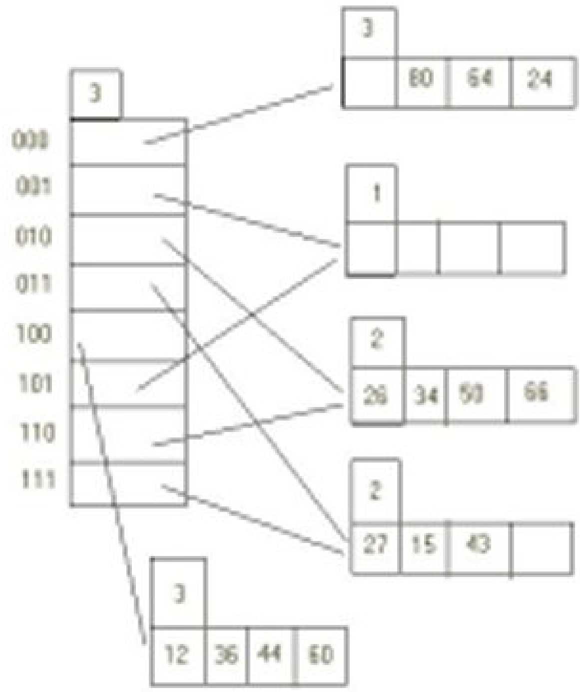
d) [5 points] Suppose we want to achieve a disk bandwidth utilization of 90%. How many buffer frames per run should we set aside in our buffer pool?

e) [5 points] What is the tradeoff in setting aside aside more buffer frames (i.e. higher disk bandwidth utilization) per run?

2. Indexes [25 points]

Hash Indexes

a) [5 points] Fill in the global depth and local depth for each bucket in the extendible hashing index shown to the right.



b) [5 points] What is the minimum number of entries that needed to be inserted to cause the creation of a new bucket?

c) [5 points] Give an example for such a sequence of entries.

B+ Tress

d) [5 points] What is the maximum height of a B+-tree with *N* data entries and fanout *k*?

e) [5 points] Consider the bulk-loading algorithm we studied in class. Assume there are *N* data entries, *k* entries fit on a leaf node, and the internal nodes have fanout *k*. Ignoring the effects of buffer management, explain briefly why bulk-loading is more or less efficient than building the tree via multiple insertions.

f) [Extra Credit: 10 points] In addition to range and equality selection queries, we would like to efficiently support queries of the form "how many tuples fall in the range of *x* to *y*". Describe how you would augment a B+-tree to support such queries. Outline how these "count" queries would be evaluated in your tree. Also outline the steps involved in inserting a new entry. Your augmented B+-tree should be able to support such count queries at much lower cost than normal B+-tree.

3. Disk, Indexes and Buffer Management [25 points]

a) [5 points] List the three major components of disk delay, organized from most to least time consuming.]

b) [5 points] We discussed two possible formats for variable-length records: one with delimiters, and another with pointers. Explain which one is preferable and why.

c) [5 points] Explain why you cannot have a sparse, unclustered index. You may want to illustrate with a example.

d) [5 points] Assume you have a buffer pool with 3 frames and a file with four pages (A, B, C, D). Assuming that the buffer pool starts out empty, give an access pattern of 8 page accesses that makes LRU do an I/O for each page access.

e) [5 points] Given the access pattern ABCDACACDC, and an empty buffer pool with 3 frames, count the number of disk reads necessary for each of LRU, MRU, and LRU-2. If a "tie" occurs in LRU-2, replace the lowest-numbered frame.

4. Formal Query Languages [25 points]

The World Organization of Furry Friends (WOOFF) has asked you to answer some questions for them. They have a database with the following schema. **Breeders** are people who raise and sell **Pets**. Pets belong to different **Breeds** (e.g. "Scmauzer", "Poodle", "Siamese", etc.) The **Pedigree** of the pet identifies the pet's breeder and breed. If a pet is of mixed breed, it's considered to belong to multiple breeds - that is, *that pet (and its breeder) will appear many times in the Pedigree table*. You may assume that pets have only one breeder. Primary keys are underlined.

Breeders(<u>brdr_id: integer</u>, brdr_name: text, age: integer)
Pets(<u>pid: integer</u>, pname: text, weight: float)
Breeds(<u>br_id: integer</u>, br_name: text, friendliness: int)
Pedigree(<u>brdr_id: integer, pid: integer, br_id: integer</u>)

a) [5 points] Give a relational calculus query to find the weights of all pets named 'Tiny'.

b) [5 points] Give a relational calculus query to find the br_id, breed name and friendliness of all breeds raised by Breeder Bob.

c) [5 points] Give a relational algebra query to find the breed name and friendliness of all breeds for which Breeder Bob raised a pet named 'Killer'. *Please draw a "tree" for your algebra, rather than a parenthesized expression.*

d) [5 points] Give a relational algebra tree to find the names of Breeders who have bred dogs in the breeds "Doberman" and "Chihuahua". (In your answer, you should count breeders who raised Doberman/Chihuahua mixes.)

e) [5 points] Give a relational calculus query ***or*** relational algebra tree (you can choose either!) to find the names of pets who belong to all the same breeds as a pet named 'Icky'.

Note: you may use the DRC or TRC, whichever you prefer.
