COSC-4411(M) Midterm #1

Sur / Last Name:
Given / First Name:
Student ID:

- **Instructor:** Parke Godfrey
- **Exam Duration:** 75 minutes
- **Term:** Winter 2004

Answer the following questions to the best of your knowledge. Be precise and be careful. The exam is open-book and open-notes. Write any assumptions you need to make along with your answers, whenever necessary.

There are five major questions. Points for each question and sub-question are as indicated. In total, the exam is out of 50 points.

If you need additional space for an answer, just indicate clearly where you are continuing.

Regrade Policy

- Regrading should only be requested in writing. Write what you would like to be reconsidered. Note, however, that an exam accepted for regrading will be reviewed and regraded in entirety (all questions).

Grading Box	
1.	
2.	
3.	
4.	
5.	
Total	

1. (10 points) **Buffer Pool.** *Okay, I've replaced the replacement strategy. What next?* [short answer / analysis]

   Dr. Mark Dogfurry of *Very Small Databases, Inc.*, has devised the following replacement strategy.

   Within the database system, every transaction has a unique timestamp value, start, which is the time the transaction commenced. (A transaction is, for example, a query executing. It will pin, and then unpin, a number of pages.) It is always a transaction that *pins* a page.

   Associated with each buffer pool frame is an xtime and a ctime. When a page's pin_count $= 0$ and the page is then pinned, or the page is initially fetched into the pool, its frame's xtime is set equal to the start value of the transaction that requested (pinned) the page. If the page is already pinned (pin_count > 0), its frame's xtime is set to the transaction's start *if* start is newer than the frame's current xtime; otherwise, its xtime value is left as is.

   The frame's ctime is set to the current clock time whenever the page's pin_count becomes 0.

   For replacement, the page with the *oldest* xtime over all pages with pin_count $= 0$ is chosen. In the case of ties for oldest xtime, the one with the *newest* ctime of those is chosen.

   a. (3 points) What type of replacement strategy is this? (LRU, MRU, Clock, hybrid, etc.?) Briefly describe.

b. (4 points) Identify an advantage Dr. Dogfurry's replacement strategy might have (compared with the basic replacement strategies of LRU, MRU, and Clock).

c. (3 points) Identify a disadvantage Dr. Dogfurry's replacement strategy might have (compared with the basic replacement strategies of LRU, MRU, and Clock).

2. (10 points) **Index Logic.** *Take the next index to the left.* [short answer / exercise]

a. (5 points) You are told that the following indexes are available on the table **Employee**:

	key	**type**	**clustered?**
A.	name, address	tree	yes
B.	age, salary	hash	no
C.	name	tree	no
D.	salary, age	hash	no
E.	name, age	tree	yes

You are suspicious that this information is not correct. Why? Identify three problems with what is reported.

b. (5 points) Consider the query

```
SELECT order#, amount, when
    FROM Purchases
    WHERE amount BETWEEN 25 AND 30
      AND when > '1999-11-14';
```

- There are 10,000,000 purchase records.
- There are 25 records are on each data-record page, on average.
- 4,000,000 purchase records have when $>$ '1999-11-14'.
- 50,000 purchase records have $25 \leq$ amount ≤ 30.

Two indexes are available:

A. A clustered B+ tree index on when of type alternative 2. The index pages are three deep, with the leaf pages at depth four.

B. An unclustered B+ tree index on amount of type alternative 2. The index pages are three deep, with the leaf pages at depth four.

For each index, 50 data entries fit per data-entry (leaf) page.

What is the I/O cost using each index to evaluate the query? So which index is best for this?

3. (10 points) **General.** *Grab bag.* [multiple choice]

a. (2 points) Consider

I. clustered tree indexes
II. unclustered tree indexes
III. clustered hash indexes
IV. unclustered hash indexes

Range queries can benefit from

A. Just **I**.
B. Just **I** & **II**.
C. Just **I**, **II**, & **III**.
D. Just **I** & **III**.
E. Potentially any of **I**, **II**, **III**, & **IV**.

b. (2 points) The buffer manager manages

A. lock management for transaction processing
B. query processing
C. file allocation and deallocation
D. disk memory
E. main memory

for the database system.

c. (2 points) Which of the following is false?

A. Locating a record by key in a sorted file by binary search and locating it via a B+ tree make practically the same number of key comparisons.
B. Locating a record by key in a sorted file by binary search requires more I/O's than locating it via a B+ tree, in general.
C. A bulk build of a B+ tree is faster than building it by inserting a record at a time.
D. If the data records are kept in a sorted file, there is no need for a B+ tree index based on the same search / sort key.
E. If there is an unclustered B+ tree index over the data records, this does not mean that the records are necessarily sorted.

d. (2 points) Which of the following is false?

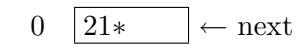
A. The trend is that disk I/O speeds are getting faster in ratio to CPU speeds.
B. Page size is dictated by the hardware.
C. Generally, many records fit on a page.
D. Sequential reads and writes are important to a database system's performance.
E. Generally, I/O time dominates CPU time in database operations.

e. (2 points) The external merge sort routine

A. for its merge passes requires that the input runs all be of equal length.
B. can accommodate variable length input runs in a merge pass, but may in that case need to allocate more output frames.
C. must use quick-sort in its pass 0.
D. may not be faster sorting a given input file given twice the buffer pool allocation.
E. cannot sort a file that is already sorted on a different key.

4. (10 points) **Index Mechanics.** *Always losing your keys?* [exercise]

   A linear hash has just been started. The linear hashed file currently just has one bucket (primary page). The current hash function pair is $\langle h_0, h_1 \rangle$. Here, h_0 masks for *zero* (!) right-hand bits from the hashed key, and so always returns bucket address 0. Hash function h_1 masks for 1 right-hand bit, h_2 for 2, and so forth. Assume that each page can hold two entries. The file currently has one entry of $21*$ (10101_2).

   A split should be triggered whenever an overflow page is created.

   Show the linear hashed file after each of the following inserts:

   - $14*$ (1110_2),
   - $7*$ (111_2),
   - $35*$ (100011_2), and
   - $28*$ (11100_2).

   The insertions are cumulative, so your final hashed file should contain $21*$, $14*$, $7*$, $35*$, and $28*$.

5. (10 points) **External Sorting.** *I'll pass.* [analysis]

Consider the following "optimization". If the last merge of the current merge pass would involve a k-way merge where $k < (B-1)$, then the routine "fills up" the last merge by adding $(B-1)-k$ current runs—runs just created in this same pass in previous merge steps—to make for a $(B-1)$-way merge, when possible.

---

a. (5 points) Say that *pass* $(i-1)$ produced 8 runs and that we are doing 3-way merges. In *pass* i, we first take 3 of the 8 runs and merge them into a new single run. We next take the next 3 of the 8 runs and merge them into a new run. Now we only have 2 of the original 8 runs left. In the basic external sort routine, we would finish *pass* i by merging these 2 runs in a 2-way merge.

Under the revised algorithm (with the "optimization"), we would perform a 3-way merge again instead, using the remaining 2 runs of the 8 (from *pass* $(i-1)$) *and* one of the 2 new runs just created in the first two merges of *pass* i.

Does this help in this example? Why or why not?

b. (5 points) Does this "optimization" actually always / ever make an external sort more efficient? That is, does it always / ever save I/O's? Argue briefly why or why not.

(Scratch space.)

Relax. Turn in your exam. Go home.