# CSE-4411A

## Assignment #1

1. (5 points) **Buffer Pool.** *A popularity contest.* [Analysis]

   Dr. Mark Dogfurry, database researcher, has an idea for a better replacement strategy, a type of improvement for *least recently used*.

   He suggests that, each page of the database, to keep a counter *fetched*, and a time-stamp *created*. Counter *fetched* would be used to count the entired number of times the page has been fetched into the buffer pool (since the page was created). Time *created* simply records the time the page was created.

   The ratio of $\textit{fetched}/(\textit{now} - \textit{created})$ would measure a page's "popularity". A replacement policy could pick the frame with the *least popular* page to reuse.

   a. (2 points) Would *least popular* likely work well as a replacement policy? Why or why not?

   b. (3 points) Would there be any additional cost that Dr. Dogfurry's counter *fetched* and time-stamp *created* would add to the operation of the buffer pool?

2. (5 points) **Index Logic.** *That doesn't make sense!* [Short Answer]

Consider the following schema.

**Employee**(<u>e#</u>, name, salary, d#)
  FK (d#) refs **Department**
**Department**(<u>d#</u>, name, location, budget)
**Project**(<u>d#</u>, <u>title</u>, budget, leader)
  FK (d#) refs **Department**
  FK (leader) refs **Employee** (e#)

Dr. Dogfurry says he has made the following indexes on the above schema.

- **Employee**:
  - **A.** unclustered hash index on e#, name
  - **B.** clustered tree index on d#, salary
- **Department**:
  - **C.** clustered hash index on d#
  - **D.** unclustered tree index on name, location
- **Project**:
  - **E.** clustered tree index on d#, title
  - **F.** clustered hash index on leader

---

a. (3 points) You do not trust what Dr. Dogfurry has said about the indexes. There are logical impossibilities, and indexes that do not make sense. (That is, another index would be better in *every* situation than that one.)

Identify at least three of these problems.

b. (2 points) An index is usually created by the system when a primary key is declared for a table. The reason is that integrity checking would be too expensive without an index.

Consider a table like **Employee** with a one-field key e#. What type of index would be more appropriate for this? Why?

What if the primary key were *composite*—consisting of multiple fields—such as title, year, studio for a table, say, **Movie**?

3. (5 points) **Index Mechanics.** *Cannot see the forest for the trees.* [Exercise]

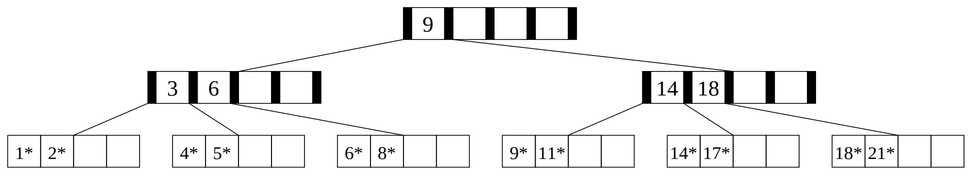

Figure 1: A B+ tree.

a. (1 point) B+ trees are a type of balanced search tree. State the structural invariant of the B+ tree that ensures it remains balanced (of log depth).

b. (2 points) Consider the B+ tree of order two in Fig 1.

Show a legally resulting B+ tree if record `9*` were eliminated from the tree.

c. (2 points) Consider the tree in 3b (before you modified it). There is a key *3* in an index page, but not in the data entry pages.

Explain how that could have happened, *or* why that must be a mistake.

4. (5 points) **Page Organization.** *Space must be free!* [Exercise]

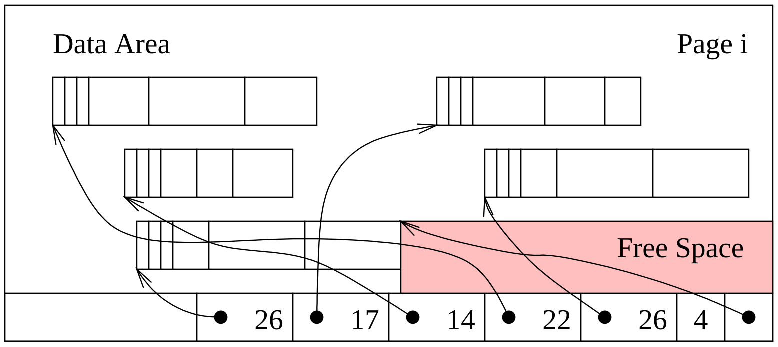

Figure 2: Page layout with variable length records.

Consider a page layout with variable length records and a slot directory as in Figure 2.

Say that we want to add a new record to page *i*, but there is not enough space for the new record in page *i*'s *free space*. However, there is enough space overall on the page for the new record. That is, if the records already on page *i* were rearranged ideally, then the new record would fit. Namely, we have to *pack* the records on page *i*: move them all so that they are sequentially adjacent starting from the beginning of the page. After this, all the empty space does belong to *free space*.

Write a pseudo-code algorithm for packing a page.