My Name:
SOLUTIONS

Midterm 1: CS186, Spring 2015

Prof. J. Hellerstein

My Course Account: cs186-_____

*You should receive a double-sided answer sheet and a 7-page exam. Mark your name and login on **both** sides of the answer sheet, and in the blanks above. For each question, place **only your final answer** on the answer sheet—do not show work or formulas there. You may use the backs of the questions for scratch paper, but **do not tear off any pages**. We will ask you to turn in your question sheets as well as your answers.*

I. Query Processing [24 points + 5 extra credit]

Frank Underwood wants to collect opinion survey data to reassure citizens that he's listening. The database system used by Underwood is configured so that **100 pages** are available for each operator (sort, hash, join, etc.) that is used in a query plan. For the `SurveyResponse` table described below, there are **2000 records** per page. Each page access is an I/O.

```
SurveyResponse(
response_id int,
district text,
state text,
comments text,
...
)
```

1. [4 points] After collecting survey results for the first month, the survey data file has grown to **9,500 pages**. Assuming we are using external merge sort to sort the survey results by district and that you must **include the cost of writing the output to disk,** are the following statements **True** or **False**? (Remember to put your answers on the **answer sheet!)**
   a. The data can be sorted in 2 passes. **True** - 100 * 99 = 9900 pages in 2 passes
   b. The I/O cost (including output) will be 19,000 I/O's. **False** - need 4N I/O's
   c. Adding 3 additional buffer pages will reduce the number of I/O's. **False** - Not enough to do anything
   d. If we additionally allocated enough buffers only for double buffering of input and output, the external sort will have speed improvements. **True**
2. [5 points] After another month, the `SurveyResponse` table has grown to **11,000 pages**. However, the budget for the servers takes a hit and the system has been downgraded

such that there are **only 11 pages** available for each query operator. How many I/O's will the external merge sort (including the cost of writing the output) require with these new memory requirements?

(1 + log_{11 - 1} (11000 / 11)) * 2 (11000) = (1 + log_{10} (1000)) * 2 * 11000 = **88,000 I/O's**

(Query Processing, continued)

Underwood wants to better utilize the survey responses so that he can reach out to local representatives to discuss important and relevant issues. In addition to the previous **11,000-page** `SurveyResponse` table, there exists another table called `Representative` that contains each local official's full name; it consists of **500 pages** with **5000 records per page**.

```
Representative(
rep_id int,
full_name text,
district text,
state text,
...
)
```

3. [5 points] Assume that we once again make **100 pages** available to each query operator. How many I/O's are required to perform a simple, "unoptimized" sort-merge join between the `Representative` and `SurveyResponse` relations on the `district` and `state` attributes? ~~Again,~~ ~~**include the cost of writing the output.**~~

   (To clarify what we mean by "unoptimized" sort merge join, the algorithm is as follows. First, completely sort each relation on its join attribute, writing the results to disk. Then, merge the two sorted relations together, and include the cost of writing the resulting output to disk. You should assume that the number of records with duplicate district and state attributes is negligibly small, so no pathological "nested loops" I/O behavior occurs during merge.)

2 passes to sort 500 pages
3 passes to sort 11,000 pages
cost to sort SurveyResponse + cost to sort Representative + cost to join
= 2*500 * 2 + 2 * 11000 * 3 + 11000 + 500 = **79500 I/O's**

4. [5 points] Assume we want to optimize our join from the previous question as we discussed in class—combining the last phase of sorting with the merge join. If the `Representative` relation is of size **L pages** and the `SurveyResponse` relation is of size **S pages**, **how**

many pages (B) do you need for your buffer to perform a sort-merge join between `Representative` and `SurveyResponse` in just **2 passes**? Please **write the expression** that describes the number of buffer pages B in terms of L and S. ***You do not need to solve for B***; please just write the expression.

During the merge & join phase, we have B-1 input buffers to work with. If we wanted to have all the runs that were generated fit in one go, we can only have at most B - 1 sorted runs after pass 0. B - 1 must be equivalent to the number of sorted runs, which is equivalent to L / B + S / B.

Hence, L / B + S / B <= B - 1 or L + S <= (B - 1) * B

(Query Processing, continued)

For the remainder of this question, assume that:

- The size of the relation Messages is **20000 pages**.
- You have **102 pages** available in memory for your query processing algorithms
- I/O cost refers to the number of disk reads and writes required by your procedure.
- You **do** need to account for the cost to read the relations in from disk.
- You **do** need to account for the cost to write your final output to disk.

Please disregard any caching effects of the buffer pool—assume that any memory that is used is controlled explicitly by the join algorithms.

You work for ~~the NSA~~ a secure messaging app trying to gather some statistics to ~~collect intelligence~~ improve your service.

You have the following table:

```
Messages(message_id int,
         sender_id int,
         receiver_id int,
         encrypted_content text,
         timestamp date,
         ...)
```

You want to make a graph of people who have talked to each other. In other words, the pair (A, B) should show up in our output if person A has sent person B at least one message and person B has sent person A at least one message. You send your system this query:

```
SELECT M1.sender_id, M2.sender_id
  FROM Messages M1, Messages M2
 WHERE M1.sender_id = M2.receiver_id
   AND M1.receiver_id = M2.sender_id;
```

Assume that `sender_id` and `receiver_id` are foreign keys pointing to some other table that uniquely identifies people using our service.

5. [5 points] What is the cost of a hash join of Messages with itself? Use the `receiver_id` of the first instance of `Messages` and the `sender_id` of the second instance of Messages as your hash key. Assume we have hash functions that distribute these keys perfectly. Use the hash join algorithm from class; do not attempt to invent any special optimizations for self-joins.
   200,000 I/Os (we need two full passes of Messages)

6. **Extra Credit [5 points]:** Now you want to optimize hash join for the special case of this particular self-join. In particular, you want to hash `Messages` only once.
   a. What should be the input to the hash function? Accepted any answer that produced a symmetric hash (hash(sender, receiver) = hash(receiver, sender))
   b. What is the cost of this optimized hash join?
   100,000 I/Os (half of 5.)

II. File Organization + Indexes [9 points]

Consider the following table:

```
CREATE TABLE Students (
  sid INT PRIMARY KEY,
  name VARCHAR(20),
  email VARCHAR(20),
  gpa FLOAT,
  years_enrolled INT,
);
```

Consider the following three queries:

```
(A) SELECT * FROM Students WHERE years_enrolled >= 2;
(B) DELETE FROM Students WHERE id = 123456;
(C) INSERT INTO Students VALUES (9999, 'Joe', 'j@b.com', 4.0, 2);
```

1. [6 points] For this question, assume there are not any indexes, and we **leave each page about 2/3 full**. Order these queries in **ascending order by the total number of I/Os required** to execute them; ties can be listed in any order.

   a. If `Students` table is arranged in a heap file, what is the order of the queries?
   C, B, A

   b. If `Students` table is arranged in a sorted file, sorted on SID, what is the order of queries?
   B, C, A or C, B, A

   c. What would be the size of a single tuple using fixed length records? Assume that all numeric types are 4 bytes long.
   4 + 20 + 20 + 4 + 4 = 52 bytes

Now, consider the following query: `SELECT * FROM Students WHERE SID=2357777`. Assume that we could build:

i. An alternative 1 B+-tree on SID
ii. An alternative 2, clustered B+-tree on SID
iii. An alternative 3, clustered B+-tree on SID
iv. An alternative 2 unclustered B+-tree on SID

2. [2 points] Please mark each of the following statements True or False on the answer sheet
   a. Index (ii) can provide better performance for this query than Index (iv), due to the potential for improved spatial locality in a clustered index.
   False - Since SID is a primary key, only one tuple will match

   b. Index (i) may be taller than Index (ii).
   True - In Alternative 1, data lives in the index, so data pages are index pages

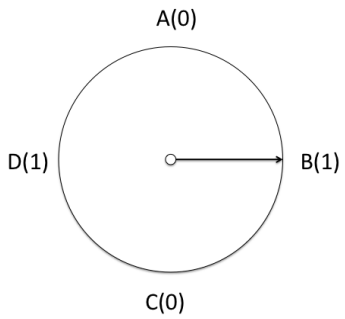
3. [1 point] For which of the five fields (`sid, name, email, gpa, years_enrolled`) would you most expect an alternative 3 index to be more space efficient than an alternative 2 index?
   years_enrolled - the field with the most duplicates (aka the least cardinality)

III. Buffer Replacement [12 points]

1. [2 points] Given 4 buffer pages and an access pattern of pages:
   I, L, O, V, E, D, B, Y, I, P, P, E
   Which pages are in the buffer pool at the end if we used an MRU cache policy?
   ELOY

2. [2 points] Given 4 buffer pages and an access pattern of pages:
   A, B, T, P, H, A, C, N, M, O, A, A, D, E, A, B, C, B, E, A, F, G, H, A, C, N, M, O, A, T, P, H,
   (Hint: you don't need to draw out a chart with every page access!)
   Which pages are in the buffer pool at the end if we used an LRU cache policy?
   ATPH - the last four page requests will necessarily be in the buffer pool with LRU

3. [2 points] Given that we are using a clock replacement policy, the state of which is shown below. On the next replacement request, the first frame to be considered is the one where the clock hand is currently pointing.



   a. A request comes in for page E. Which page does this replace? C
   b. What is the reference bit of the frame holding E after E is unpinned? 1

4. [6 points] Assume that we have a database workload that repeatedly scans a single relation R, which is much bigger than the buffer pool. We learned in class that LRU replacement performs poorly in this case. Consider the following replacement policies:
   i. Least-Recently Used (LRU): replace the page in the least-recently unpinned frame.
   ii. Most-Recently Used (MRU): replace the page in the most-recently unpinned frame.
   iii. Pin All But One (PABO): Given B buffers, pin the first B-1 pages that you access into B-1 frames, and always replace the page in the remaining frame.
   iv. Random: Choose the page to replace randomly among the unpinned pages.

   a. Which of the five protocols performs optimally for this workload? (Note: the correct answer may be zero, one, or more than one of the above!)
   (ii) MRU

   Note: this question is tricky! It's tempting to say PABO as well, but a small example shows MRU outperforms PABO for sequential flooding:

   Sequential flooding over ABCDEF (6 pages) with 4 page buffer

   MRU:
   Pass0: buffer is now ABCF
   Pass1: buffer is now ABEF (hits: ABCF, misses: DE), hit rate: 4/6
   Pass2: buffer is now ADEF (hits: ADEF, misses: BC), hit rate: 4/6
   …

   PABO:
   Pass0: buffer is now ABCF

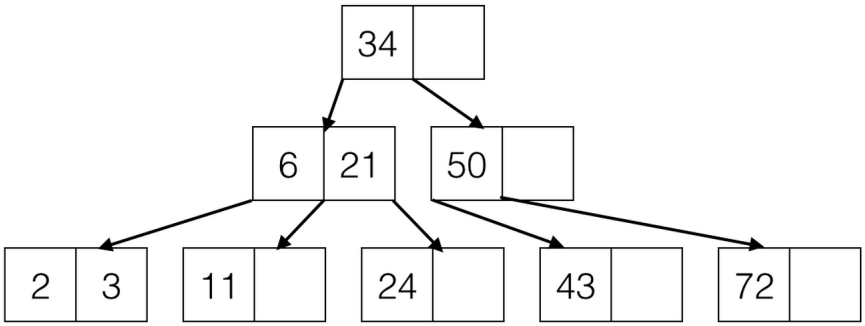
Pass1: buffer is now ABCF (hits: ABC, misses: DEF), hit rate: 3/6
Pass2: buffer is now ABCF (hits: ABC, misses: DEF), hit rate: 3/6

b. Which of the five protocols performs pessimally (as badly as possible) for this workload? (Again, could be zero, one, or more than one!)
(i) LRU

c. Which of the five protocols is worse than Random (again: could be 0, 1, or more!)
(i) LRU

IV B+-Trees [14 points]

Assume we have the following B+-Tree of order 1. Each index node must have either 1 or 2 keys (2 or 3 pointers), and the leaf nodes can hold up to 2 entries.



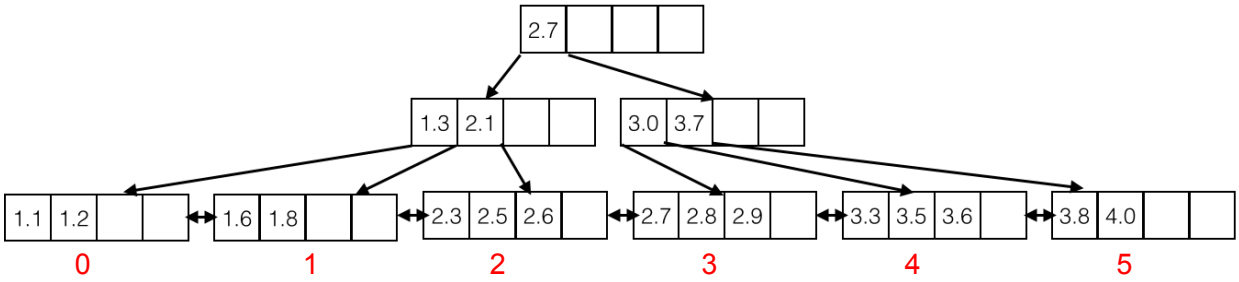
1. [5 points] What is the maximum number of keys you could insert that would NOT change the height of the above tree?
   12
   A possible insertion pattern is: 12, 25, 42, 73, 41, 40, 74, 71, 39, 38, 70, 69

2. [5 points] What is the minimum number of keys you could insert to change the height of this tree?
   3
   A possible insertion pattern is: 1, 4, 5

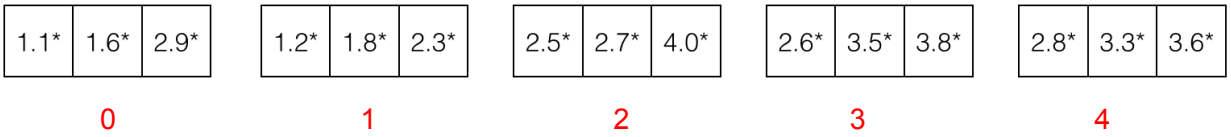
Assume that the following B+-Tree is an unclustered, alternative-2 index on gpa. The actual data is laid out in a separate heap file as shown below, where each record is depicted by its key, followed by an asterisk to represent the rest of the record. There are pointers (not shown) from the leaf pages of the B+-tree to the corresponding tuples in the heap file.

3. [2 point] How many **leaf index pages** would be read by the following query?
   ```
   SELECT * FROM Students WHERE gpa >= 1.4 AND  gpa <= 3.5
   ```
   4
   Labeling each leaf index page from 0 to 5 from left to right, the following leaf pages would be read: 1, 2, 3, 4

4. [2 point] How many **heap file page** reads would you read if you simply followed the record ids in the index data entries as soon as you see them (i.e., if you did not sort the record ids first)? Assume you can only hold one heap file page in memory at any time.
   9
   Labeling each heap file page from 0 to 4 from left to right, here is the access pattern for the heap file page reads: 0, 1, 2, 3, 2, 4, 0, 4, 3


2.7
1.3
2.1
3.0
3.7
1.1
1.2
1.6
1.8
2.3
2.5
2.6
2.7
2.8
2.9
3.3
3.5
3.6
3.8
4.0
0
1
2
3
4
5

Heap File:


1.1*
1.6*
2.9*
1.2*
1.8*
2.3*
2.5*
2.7*
4.0*
2.6*
3.5*
3.8*
2.8*
3.3*
3.6*
0
1
2
3
4

V Query Languages [15 points]

1. [3 points] Consider two relations with the same schema: R(A,B) and S(A,B). Which one of the following relational algebra expressions is not equivalent to the others?
    a. $\pi_{R.A}((R \cup S) - S)$
    b. $\pi_{R.A}R - \pi_{R.A}(R \cap S)$
    c. $\pi_{R.A}(R - S) \cap \pi_{R.A}R$
    d. They are all equivalent.

    Everyone was given credit on this question.
    The correct answer is b:
    A counter example is R(A, B) = {(1,2), (1,3)}, S(A, B) = {(1,2)}. a and c will have size 1, while b will have size 0.

The CS 186 TAs have decided to go into real estate! All of the information we need is described by the following schema, where the primary key of each table is underlined:

```
Homes(home_id int, city text, bedrooms int, bathrooms int,
      area int)
Transactions(home_id int, buyer_id int, seller_id int,
             transaction_date date, sale_price int)
Buyers(buyer_id int, name text)
Sellers(seller_id int, name text)
```

For the query language questions below, **fill in the blanks on the answer sheet** to complete the query. For each SQL query and nested subquery, please **start a new line when you reach a SQL keyword** (SELECT, WHERE, AND, etc.). However, do not start a new line for aggregate functions (COUNT, SUM, etc.), and comparisons (LIKE, AS, IN, NOT IN, EXISTS, NOT EXISTS, ANY, or ALL.)

2. [2 points] Fill in the blanks in the SQL query to find the duplicate-free set of id's of all homes in Berkeley with at least 6 bedrooms and at least 2 bathrooms that were bought by "Bobby Tables".

```
SELECT DISTINCT H.home_id
FROM Homes H, Transactions T, Buyers B
WHERE H.home_id=T.home_id
AND T.buyer_id=B.buyer_id
AND H.city="Berkeley"
AND H.bedrooms>=6
AND H.bathrooms>=2
AND B.name="Bobby Tables"
;
```

3. [2 points] Now for the same Query as question 2, fill in the blanks for the given relational algebra plan. Note: blanks for relations are **bold**; blanks for subscripts are subscripted and not bold.

$\pi_{\text{home\_id}}$ (
  $\sigma_{\text{name="Bobby Tables"}}$ (
    (($\sigma_{\text{city="Berkeley" AND bedrooms>=6 AND bathrooms>=2}}$ Homes)
    ⋈ Transactions) ⋈ Buyers))

An alternate solution was to perform the selection on the Buyers table first.

$\pi_{\text{home\_id}}$ (
  $\sigma_{\text{city="Berkeley" and bedrooms>=6 and bathrooms>=2}}$ (
    (($\sigma_{\text{name="Bobby Tables"}}$ Buyers)
    ⋈ Transactions) ⋈ Homes))

4. [3 points] Fill in the blanks in the SQL query that finds, for each home in Berkeley, the id of the home and the price for which it was sold. If the home has not been sold yet, **the price should be NULL**.

```
SELECT H.home_id, T.sale_price
FROM Homes H
LEFT OUTER JOIN Transactions T
ON H.home_id=T.home_id
WHERE H.city="Berkeley"
;
```

An alternate solution was to use Transactions in the FROM clause and perform a RIGHT OUTER JOIN with Homes.

```
SELECT H.home_id, T.sale_price
FROM Transactions T
RIGHT OUTER JOIN Homes H
ON H.home_id=T.home_id
WHERE H.city="Berkeley"
;
```

5. [5 points] Fill in the blanks in the SQL query to find the name of the city with the highest number of homes that have not yet been sold. **Include the number of unsold homes in your output.**

The most common solution was to check that home_id was NOT IN the result of the subquery:

```
SELECT H.city, COUNT(*) AS cnt
FROM Homes H
WHERE H.home_id NOT IN (
```

```
    SELECT DISTINCT T.home_id
    FROM Transactions T
)
GROUP BY H.city
ORDER BY cnt DESC
LIMIT 1
;
```

Another solution used NOT EXISTS with a correlated subquery:

```
SELECT H.city, COUNT(*) AS cnt
FROM Homes H
WHERE NOT EXISTS (
    SELECT DISTINCT T.home_id
    FROM Transactions T
    WHERE H.home_id=T.home_id
)
GROUP BY H.city
ORDER BY cnt DESC
LIMIT 1;
```