CS 245
Midterm Exam – Winter 2014

This exam is open book and notes. You can use a calculator and your laptop to access course notes and videos (but not to communicate with other people). You have 70 minutes to complete the exam.

Print your name:____

The Honor Code is an undertaking of the students, individually and collectively:

1. that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;
2. that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.

While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

I acknowledge and accept the Honor Code.

Signed:____

Problem	Points	Maximum
1		10
2		10
3		10
4		10
5		10
6		10
Total		60

Problem 1 (10 points)

Suppose blocks of 1024 bytes are used to store variable-length records. The fixed part of the block header is 24 bytes, and includes a count of the number of records. In addition, the header contains a 4-byte pointer (offset) to each record in the block. The first record is placed at the end of the block, the second record starts where the first one ends, and so on. The size of each record is not stored explicitly, since the size can be computed from the pointers.

(a) How many records of size 50 bytes could we fully fit (no spanning) in such a block? (The block is designed to hold variable size records, but these records by coincidence happen to be all the same size.)

Number of records: $\lfloor \frac{1024-24}{50+4} \rfloor = 18$

(b) Suppose we want to fully store 20 records in the block, again all of the same size. What is the maximum size of such records?

Maximum record size: $\lfloor \frac{1024-24-4*20}{20} \rfloor = 46$

(c) Now assume the block layout is changed to accommodate only fixed size records. In this case, the block header is still 24 bytes and contains the common length of the records in the block.

How many records of size 50 bytes could we fully fit in such a block?

Number of records: $\lfloor \frac{1024-24}{50} \rfloor = 20$

(d) For this part, assume we allow record spanning. Records are fixed size, but since we can have record fragments we need to add a 5-byte header to every fragment to store its length (and other information). (The block header is still 24 bytes.)

Suppose we have 2 blocks of 1024 bytes each, we would like to store 3 records of 600 bytes each. We first allocate as much as we can of the records into the first block, and then use space from the second block. After the 3 records are stored, how many bytes are still available in the second block?

Free bytes available: 180

Record 2 is split into fragments 2-a and 2-b. Block 1 has room for 390 bytes of record 2-a (1024-24-5-600-5). Block 2 needs to store 210 bytes of record 2-b. Block 2 also needs to store block header, record 3, and two record headers (for record 2-b and 3). Therefore, 180 bytes (1024-24-5-210-5-600) are still available in block 2.

Problem 2 (10 points)

Consider a relational DBMS with the following relation `E`:

- `E` has attributes `id, name, age, salary`.
- `id` is the primary key attribute (no duplicate ids).
- `E` holds 1000 tuples with `id` values 1,2,3,...,1000.
- All records (tuples) in `E` are stored sequentially (in increasing order) based on their `id`.
- Each data block that holds `E` records can hold up to 5 records.

We are building a traditional index (not a B+-tree) for this sequential file. Each index block can hold up to 10 (value, pointer) entries (where the pointer can identify either a block or a record). Index entries are sorted by value. To search for a key in an index, assume the system traverses the blocks that make up the index is sequence, starting from the first block.

(a) Suppose we construct a dense index `INDEX1` on `E.id`. How many blocks must the system read to retrieve the contents of the record with `id` = 567?

Number of blocks:$\lceil \frac{567}{10} \rceil + 1 = 58$

57 `INDEX1` reads + 1 record read

(b) To improve access speed, we add an index `INDEX2` on `INDEX1`. What type of index makes sense, sparse or dense?

Type of index: Sparse

(c) After building `INDEX2`, how many blocks must the system read to retrieve the contents of the record with `id` = 567?

Number of blocks:$\lceil \lceil \frac{567}{10} \rceil / 10 \rceil + 1 + 1 = 8$

6 `INDEX2` reads + 1 `INDEX1` read + 1 record read

(d) Suppose now build a secondary index on `E.salary`. What type of index makes sense, sparse or dense?

Type of index: Dense

(e) Continuing with the secondary `E.salary` index, suppose that there are 50 distinct `salary` values, with an equal number of records per salary value. First let us say that we implement the `E.salary` index in a naive way, where duplicate values are stored in the index. That is, if there are x records `salary` = 1000, the index contains x entries of the form (1000, pointer). (Salaries are sorted in the index.)

Say we want to read the content of *all* records with the largest salary. In the worst case scenario, how many blocks need to be fetched?

Number of blocks: $\lceil \frac{1000}{10} \rceil + \lceil \frac{1000}{50} \rceil = 120$

1000 pointers at 10 per block + 20 reads for 20 records

Note: Assuming that one can jump to the final block and read from it will receive full credit as well.

(f) Next, suppose we build `E.salary` without duplicate values. For instance, if there are x records with `salary` = 1000, there will be a single entry in the index, pointing to a set of contiguous blocks that store the x record pointers. Salaries are still sorted in the index. Ignore space requirements for pointer counts and next block pointers.

In the worst case scenario, how many blocks need to be fetched to read the content of all records with the largest salary?

Number of blocks: $\lceil \frac{50}{10} \rceil + \lceil \frac{1000/50}{10} \rceil + \lceil \frac{1000}{50} \rceil = 27$

50 salary values at 10 per block + 20 pointers to records at 10 per block + 20 reads for 20 records

Note: The answer above assumes that one can fit 10 pointers per block, assuming that one can fit 20 pointers per block will also receive full credit. Assuming the one can jump to the final block will also receive full credit.

Problem 3 (10 points)

(a) Consider an extensible hash structure with the following characteristics:

- Buckets can hold up to two records.
- No overflow blocks are allowed.
- The hash function we use generates $b = 4$ bits total.
- Initially the extensible hash table is empty.

Say we insert X records, where the search key of each record generates a *distinct* 4-bit hash value (no collisions). No records are deleted during this process. We are told that after the X insertions, 4 buckets have been allocated. (Note that the previous sentence does not refer to the size of the directory.)

What is the minimum possible value of X?

Minimum value of X: 3

(b) In the same scenario as part (a), what is the maximum possible value of X?

Maximum value of X: 8

(c) Next consider a linear hash table (not the extensible hash table of parts (a) and (b)). This table has the following characteristics:

- Buckets can hold up to two records.
- No overflow blocks are allowed, so when necessary the table is expanded to avoid the use of overflow chains.
- The hash function we use generates $b = 4$ bits total.
- Initially the linear hash table is empty.

We first insert three records with hash keys 0000, 0111, and 1111. After that, two more records are inserted. The keys of all five inserted records are distinct, and there are no deletions at any point.

We are told that after the insertions, a total of Y blocks have been allocated. What is the minimum possible value of Y?

Minimum value of Y: 3

(d) In the same scenario of part (c), what is the maximum possible value of Y?

Maximum value of Y: 8

Problem 4 (10 points)

Consider a B+ Tree where each node can have at most 3 keys. Suppose the tree initially has a root node with two children leaf nodes, which contain keys $\{1, 3, 5\}$ and $\{7, 9, 11\}$ respectively. The root node has a single key 7.

(a) Four distinct keys from the set $\{0, 2, 4, 6, 8, 10, 12\}$ are added to the tree in some order, which causes the tree to grow by one level. Which four keys were added to the tree? Write the four keys separated by commas. If there are multiple solutions, write any one solution.

Sequence of 4 keys: Any set which has one or three of $\{0, 2, 4, 6\}$ and three or one of $\{8, 10, 12\}$ (as an example, numbers 0,8,10,12 is a valid solution)

(b) Suppose we are back to the original tree with two leaf nodes having keys $\{1, 3, 5\}$ and $\{7, 9, 11\}$. Now X distinct keys from the set $\{0, 2, 4, 6, 8, 10, 12\}$ are added to the tree in some order, and the tree does *not* grow a level. What is the minimum possible value of X?

Minimum value of X: 0

(c) In the same scenario as part (b), what is the maximum possible value of X?

Maximum value of X: 6

Problem 5 (10 points)

Consider two relations $R(A, B, C)$ and $S(B, C, D)$. We want to estimate the number of tuples and the size of the following expression: $U = \pi_{ACD}[(\sigma_{A=3 \wedge B=5} R) \bowtie S)]$.

We are given the following information:

- $T(R) = 100000$; $V(R, A) = 20$; $V(R, B) = 50$; $V(R, C) = 150$.
- $T(S) = 5000$; $V(S, B) = 100$; $V(S, C) = 200$; $V(S, D) = 30$.
- All attributes are 10 bytes in size.
- We use the "containment of value sets" assumption.
- We assume query values are selected from values in the relations.

(a) First consider the innermost select $W = \sigma_{A=3 \wedge B=5} R$. Compute the following values.

$T(W) = 100$ $\quad S(W) = 30$

$V(W, A) = 1$ $\quad V(W, B) = 1$

$V(W, C) = 100$

Explanation: $T(W) = \frac{T(R)}{V(R,A)V(R,B)} = \frac{100000}{20*50} = 100$

Note: $V(W, C)$ is a trick question. The maximum possible value for $V(W, C)$ is $T(W)$.

(b) Next consider the join $Y = W \bowtie S$. Compute the following values.

$T(Y) = 25$ $\quad S(Y) = 40$

$V(Y, A) = 1$ $\quad V(Y, B) = 1$

$V(Y, C) = 25$ $\quad V(Y, D) = 25$

Explanation: $T(Y) = \frac{T(W)T(S)}{max(V(W,B),V(S,B))max(V(W,C),V(S,C))} = \frac{100*5000}{100*200} = 25$.

(c) Finally consider the full expression $U = \pi_{ACD}[(\sigma_{A=3 \wedge B=5} R) \bowtie S)]$. Compute the following values.

$T(U) = 25$ $\quad$ $S(U) = 30$

$V(U, A) = 1$ $\quad$ $V(U, C) = 25$

$V(U, D) = 25$

Problem 6 (10 points)

State if the following statements are true or false. Please write TRUE or FALSE in the space provided. (You will be penalized for incorrect answers, so leave the box blank if you don't know.)

(a) The hash join algorithm assumes that the input relations are sorted.

ANSWER: False

(b) The transformation $\sigma_p(R - S) = \sigma_p(R) - S$ is valid for both sets and bags.

ANSWER: True

(c) Keeping full histograms can result in more accurate estimates than just keeping a value count, but histograms requires more processing to maintain, so they could result in worse overall performance.

ANSWER: True

(d) Consider relation $R(A, B, C, D)$. The following transformation is valid for sets, where P is a predicate involving attributes B and C: $\pi_A[\sigma_P(R)] = \sigma_P[\pi_A(R)]$

ANSWER: False

(e) Consider a disk that partitions tracks into three groups: the outermost tracks store more data, the middle tracks store less data, and the innermost the least. In such a disk, the expected rotational delay will depend on what track is being accessed.

ANSWER: False

(f) Extensible hashing makes searching over an encrypted database very efficient.

ANSWER: False

(g) The five minute rule can be useful for determining what data resides in memory and what data resides on disk.

ANSWER: True

(h) Data access on an SSD is random access (as in main memory), not block oriented (as on a disk).

ANSWER: False

(i) A traditional "row store" organization is better than a "column store" for transactional applications that access data for a single customer or entity at a time.

ANSWER: True

(j) The elevator disk-scheduling algorithm is named after the way real elevators make their stops.

ANSWER: True