I SQL [10 points]

Consider the following pairs of SQL queries. Your job is to figure out which of them are semantically equivalent (i.e., compute the same answer for all possible inputs).

1. For each pair, indicate if they are equivalent. If not, briefly describe a situation or example in which they would produce different answers.

(Select "Equivalent" or "Not Equivalent" for each pair.)

Query 1

```
SELECT x, m
FROM A,
  (SELECT MAX(B.b) AS m
  FROM B)
as C
WHERE A.a = C.m
GROUP BY x, m
```

Query 2

```
SELECT x, MAX(B.b)
FROM A, B
WHERE A.a = B.b
GROUP BY x
```

A. [**5 points**] Equivalent Not Equivalent

If "Not Equivalent", briefly describe why they are not equivalent, or give an example input where they would differ.

Not Equivalent. Query 2 selects the MAX from each group whereas Query 1 selects the MAX from the entire table of B

Query 1

```
– A.b is a unique non-null
– primary key for A
SELECT T.a
FROM (
SELECT A.a, B.c
FROM A LEFT OUTER JOIN B
ON A.b = B.c
) as T
WHERE T.c IS NOT NULL
```

Query 2

```
SELECT A.a
FROM A JOIN B
ON A.b = B.c
```

B. [**5 points**] Equivalent Not Equivalent

If "Not Equivalent", briefly describe why they are not equivalent, or give an example input where they would differ.

Equivalent.

Your answer:

II Disk, Buffers and Files [10 points]

1. [**4 points**] Consider the following relation:

```
CREATE TABLE flights (
flno: INTEGER, -- cannot be NULL
from: VARCHAR(8), -- NOT NULL
to: VARCHAR(8), -- NOT NULL
gate_number: CHAR(20), -- may be NULL
distance: INTEGER, -- NOT NULL
price: INTEGER, -- NOT NULL
pilot_name: CHAR(15), -- NOT NULL
PRIMARY KEY (flno)
);
```

A field of type CHAR(n) consists of exactly n characters, while a field of type VARCHAR(n) consists of at most n characters (assume that a field of VARCHAR(n) takes up at most n bytes). Assume that record headers take up 8 bytes, and integers are 4 bytes long. Note that columns in a primary key cannot be NULL. Which of the following, if any, are a possible size, in bytes, for some record of the flights relation, assuming a variable-length representation with a record header? There may be one, or more than one correct answer. **Your answer:**

A. 72

B. 69

C. 53

D. 51

E. 28

35-51, 55-71

2. [**6 points**] What is the maximum number of records that can be stored on a 1 KB (1024 bytes) page given the schema from the previous question? Assume that a [**record pointer, record length**] pair take up 8 bytes, the **slot count** is 4 bytes, and the **free space pointer** is 4 bytes.

In order to maximize the number of records that can be stored, we will consider the case when all records are of the minimum size. In the previous question, the minimum size of the record was calculated to be 14 bytes. Additionally, we need to account for the slot directory entires and the rest of the page footer, containing a **free space pointer** and a **slot count**. The **slot count** is 4B, the **free space pointer** is 4B, and the slot directory entries ([**record pointer, record length**]) are 8B per record.
The max number of records that can be stored is $\lfloor \frac{1024-8}{35+8} \rfloor = 23$

Your answer:

III File Organization [10 points]

In I/O (Input/Output) cost model for analysis, we define:
B: the number of data blocks in the file;
R: the number of records per block;
D: average time to read/write one disk block.
Please answer the following questions:

1. **[3 points]** Please compare the average I/O cost of range search in heap file and sorted file? Correct answer without proper explanation will earn no scores.
2. **[4 points]** Please compare the average I/O cost of deletion of a single record in heap file and sorted file? Assume that the sorted file keeps compacted after deletion. Correct answer without proper explanation will earn no scores.
3. **[4 points]** Is there any type of file organization to reduce the I/O cost of deletion? If yes, show the type and its I/O cost.

Your answer:

1. Heap file: B × D. It needs to scan the whole file to find all the matching records.
   Sorted file: $(log2B + \#pages)D$. The cost of $(log2B)D$ is used to locate the first matching record, and #pages × D is used to find all the matching records, with #pages being the number of pages containing the matching records.

2. Heap file: $(0.5B + 1)D$. The first part $(0.5B)$ × D denotes the average cost of locating the record, and the second part $1D$ is the cost of writing the revised page back into the disk. Sortedfile: $(log2B + B)D$. The first part $(log2B)D$ denotes the average cost of locating the record, and the second part BD is the average cost of shifting the rest pages by 1 record.

3. The indexed file with B+ tree index constructed on the sorted key. The I/O cost is $(log2B)D$.

IV Indexes and B+ Trees [10 points]

Given the following (degree d = 1) B+ tree:

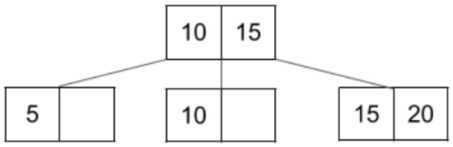

Figure 1: B+ Tree

1. **[4 points]** Draw what the tree looks like after adding 2, 6, and 12 in that order.
   For the following questions, consider the tree directly after 2, 6, and 12 have been added.
2. **[3 points]** What is the maximum number of inserts we can do without changing the height of the tree?
3. **[3 points]** What is the minimum number of inserts we can do that will change the height of the tree?

Your answer:

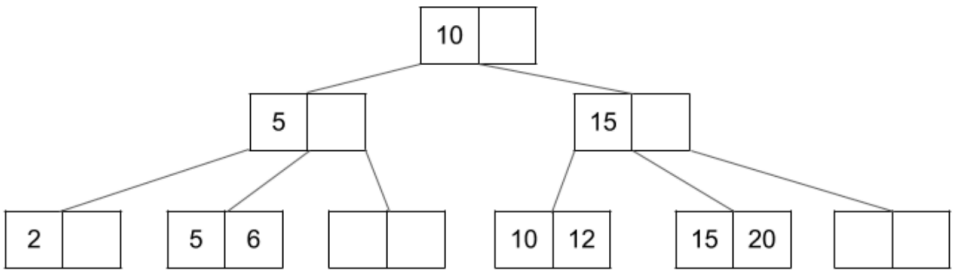

Figure 2: B+ Tree

2. Answer: 11 inserts. The maximum number of keys for a fixed height h is given by $2d \cdot (2d+1)^h$. We have $d = 1$ from the question, and $h = 2$ (we must remember h is the number of non-root layers). Plugging these numbers in gives us $2 * 3^2 = 18$. Now, we already have 7 keys in the tree, so we can insert 18-7=11 more.
3. Answer: 4 inserts (e.g. 16, 17, 18, 19). The idea is to repeatedly insert into the fullest nodes; this is hopefully apparent from understanding how and when B-trees split.

V Buffer Management [10 points]

We're given a buffer pool with 4 pages, which is empty to begin with. Answer the following questions given this access pattern:

ABCDEBADCAEC

1. **[2 points]** What is the hit rate for if we use the MRU policy?
2. **[2 points]** What is the hit rate for if we use the Clock policy?
3. **[2 points]** Which pages are in the buffer pool after this sequences of accesses if we use the Clock policy?
4. **[2 points]** Which pages have their reference bits set if we use the Clock policy?
5. **[2 points]** Which page is the arm of the clock pointing to if we use the Clock policy?

Your answer:

1. 4/12
2. B, D, A, C
3. 5/12
4. A, C, D, E.
5. A, C, E.
6. A

VI Relational Algebra [10 points]

Given the three following relations:
teams(teamid, tname)
players(playerid, pname, teamid, entrytime)
schools(schoolid, sname, playerid)
Answer the following questions:

1. **[3 points]** Write an expression that finds the name of every player in school named "STU".

   Solution
   $\pi_{\text{players.pname}}(\sigma_{\text{schools.sname='STU'}}(\text{schools} \bowtie_{\text{schools.playerid=players.playerid}} \text{players}))$

2. **[3 points]** Write an expression that is equivalent to the following SQL query:

```
SELECT teamid AS tid
FROM players
WHERE players.teamid NOT IN
(SELECT teamid FROM teams )
AND entrytime=2024;
```

   Solution
   $\rho_{\text{teamid}\to\text{tid}}(\pi_{\text{players.teamid}}(\sigma_{\text{entrytime=2024}}(\textbf{players})) - \pi_{\text{players.teamid}}(\textbf{players} \bowtie_{\textbf{players.teamid=teams.teamid}} \textbf{teams}))$

3. **[4 points]** For every record in players, output the name of the player, the name of his team, and the name of his school. Mark all that apply.

   A. $(\pi_{\text{sname}}(\text{schools})) \bowtie (\pi_{\text{tname}}(\text{teams})) \bowtie (\pi_{\text{pname}}(\text{players}))$

   B. $\pi_{\text{sname,tname,pname}}(\text{schools} \bowtie \text{teams} \bowtie \text{players})$

   C. $\pi_{\text{sname,tname,pname}}(\text{schools} \bowtie \text{teams} \bowtie (\pi_{\text{schoolid,teamid,pname}}(\text{players})))$

   D. $\pi_{\text{sname,tname,pname}}((\pi_{\text{schoolid,sname}}(\text{schools})) \bowtie (\pi_{\text{teamid,tname}}(\text{teams})) \bowtie (\pi_{\text{pname}}(\text{players})))$

   Solution
   B.
   In C, the player table has no attribute called schoolid

VII External Sorting [10 points]

Your "dream machine" allocates 64MB for the buffer (memory) for its external sorting algorithms. All input is read from disk and all output is written to disk. The I/O cost is the number of page reads/writes, where each page is 256KB large. Note that 1024KB = 1MB.

1. **[3 points]** What's the I/O cost of fully sorting a 4096 MB file with external merge sort? Explain (Just write down the equation form is also OK).

   Solution
   B = 256 pages
   N = 16384 pages
   passes = 2
   I/O = 2*2*4096 MB = 16384 MB = 65536 pages

2. **[4 points]** Suppose we reduce the size of our buffer to 32 MB. What is the largest file size (in MB) that we can externally sort in 2 passes? Explain (Just write down the equation form is also OK).

   Solution
   B = 128 pages
   N = B(B-1) = 16256 pages = 4064 MB

3. **[3 points]** If we double the buffer size, how much larger of a file can we externally sort in k passes?
   A. 2 times larger
   B. 2k times larger
   C. 2^k times larger
   D. k^2 times larger

   Solution
   C

Your answer:

VIII External Hashing [10 points]

Your "dream machine" allocates 64 MB for the buffer (memory) for its external sorting algorithms. All input is read from disk and all output is written to disk. The I/O cost is the number of page reads/writes, where each page is 256KB large. Note that 1024KB = 1MB.

1. **[3 points]** How does the I/O cost of hashing a 4096 MB file compared with sorting that file? Assume that the data is uniformly distributed on the hash key and that your hashing function distributes the records evenly.
   A. External hashing costs fewer I/Os.
   B. External merge sort costs fewer I/Os.
   C. They cost the same I/Os.

   Solution
   C.
   Both need 2 passes.

2. **[3 points]** Suppose you are hashing a 4096 MB file and one of the partitions is 128 MB and all other partitions fit in the 64 MB buffer and are distributed evenly. How many I/Os does it cost? Explain (Just write down the equation form is also OK).

   Solution
   4*4096(Divide1 + Conquer) + 2*128(Divide2) = 16640 MB
   So the IO is 16640*4 = 66560

3. **[4 points]** In this machine, what is the minimum number of pages we could externally hash to guarantee that we would have to use recursive partitioning? Explain (Just write down the equation form is also OK).

   Solution
   B(B-1)+1 = (64*4-1)*64*4 + 1 = 65281

Your answer:

IX Iterations and Joins [10 points]

1. **[4 points]** Use "T" or "F" to determine whether each of the following statements is true or false.

   (a) Block(chunk)-nested loop join is always better than page-oriented nested loop join.

   Solution
   F. (Consider when B=3)

   (b) Hybrid Hash-Join is always better than block(chunk)-nested loops join.

   Solution
   F.

2. **[6 points]** There are two tables R and S. R has attributes (x,y) and S has attributes (y,z). The set of values of y in R are the same as the set of values of y in S. Assume that there are no indexes available and that there are 10 pages in the buffer available. Table R has 200 pages with 5 tuples per page. Table S has 50 pages with 10 tuples per page. Compute the I/O costs for the following joins. The I/O cost is the number of page reads/writes.

   (a) **[3 points]** Hash join with S as the outer relation and R as the inner relation.

   Solution
   3*(200+50) = 750
   Scan R and Probe S. $|s| < (B-1) * (B-2)$, so partitioning phase only needs 1 pass. The cost is equal to $3[R] + 3[S]$

   (b) **[3 points]** Sort merge join with S as the outer relation and R as the inner relation in the worst case.

   Solution
   6*200 + 4*50 + 50 + 50*10*200 = 101450
   The cost is equal to sort$R$ + sort$S$ + $[S] + |S| * [R]$

X Query Optimization [10 points]

Consider the following query with a histogram on Sales.quantity:

```
SELECT name, price
FROM Item, Store, Sales
WHERE Item.iid = Sales.iid AND Store.sid = Sales.sid AND Sales.quantity >= 200
ORDER BY price
```

Table 1: Histogram on Sales.quantity

0-100	100-200	200-300	300-400	400-500
15%	30%	25%	20%	10%

1. [**2 points**] True or False: Assume a sale always has a corresponding item and store. Applying a filter using the histogram on Sales.quantity before joining always yields a cheaper plan for this query.

   A. True

   B. False

   If Sales is the inner table when it is joined, then pushing the select down would not reduce the I/O cost unless we materialized the results.

2. [**2 points**] Based on the histogram on Sales.quantity, what is the selectivity for Sales.quantity >= 200? Assume that the histogram boundaries are left inclusive and right exclusive.

   1 - 0.15 - 0.3 = 0.55.

3. [**3 points**] Suppose we have the following list of 2 table subplans and their costs. Mark the letters of the plans that will be chosen by the optimizer. Ignore interesting orders.

   A. Item ⋈ Sales (2,000 IO)

   B. Sales ⋈ Store (4,000 IO)

   C. Sales ⋈ Item (1,500 IO)

   D. Store ⋈ Sales (3,000 IO)

   Plans C and D will be chosen. Since Plans A and C both join the same sets of tables, we retain C over A because it has a cheaper IO cost. The same reasoning applies when we decide to retain Plan D over Plan B.

4. [**3 points**] True or False: If we have an unclustered B+-tree index on Sales.quantity, the System R optimizer may consider the index scan due to interesting order even if a sequential scan is cheaper.

   A. True

   B. False

   False. since quantity is not used in the query downstream, it is not an interesting order.

Your answer: