CS 186 Spring 2021

Introduction to Database Systems

FINAL EXAM

INSTRUCTIONS

This is your exam. Complete it either at exam.cs61a.org or, if that doesn't work, by emailing course staff with your solutions before the exam deadline.

This exam is intended for the student with email address `<EMAILADDRESS>`. If this is not your email address, notify course staff immediately, as each exam is different. Do not distribute this exam PDF even after the exam ends, as some students may be taking the exam in a different time zone.

For questions with **circular bubbles**, you should select exactly *one* choice.

◯ You must choose either this option

◯ Or this one, but not both!

For questions with **square checkboxes**, you may select *multiple* choices.

☐ You could select this choice.

☐ You could select this one too!

You may start your exam now. Your exam is due at <DEADLINE> Pacific Time. Go to the next page to begin.

Preliminaries

1 CS W186 Spring 2021 Final

`Do not share this exam until solutions are released.`

1.0.1 Contents:

- The final has 10 questions, each with multiple parts, and worth a total of 164 points.

1.0.2 Taking the exam:

- You have 170 minutes to complete the midterm.
- You may print this exam to work on it.
- For each question, submit only your final answer on Examtool.
- For numerical answers, do not input any suffixes (i.e. if your answer is 5 I/Os, only input 5 and not 5 I/Os or 5 IOs).
- Make sure to save your answers in Examtool at the end of the exam, although the website should autosave your answers as well.

1.0.3 Aids:

- You may use three pages (double sided) of handwritten notes as well as a calculator.
- You must work individually on this exam.

1.0.4 Grading Notes:

- All I/Os must be written as integers. There is no such thing as 1.02 I/Os – that is actually 2 I/Os.
- 1 KB = 1024 bytes. We will be using powers of 2, not powers of 10
- Unsimplified answers, like those left in log format, will receive a point penalty.

(a) What is your full name?

(b) What is your student ID number?

1. **(16 points) Boba Trees**

   (a) **(3 points) True or False**

   Mark all statements that are true.

   i. **(3 pt)**

   ☐ All paths from the root of a B+ tree to a leaf node will always pass through the same number of inner nodes.

   ☐ Clustered Alternate 3 B+ trees are always at least as efficient as clustered Alternate 2 B+ trees in terms of I/O cost.

   ☐ A B+ tree with height h and fanout d will always have more than $(2d)(2d+1)^{(h-1)}$ keys.

   ☐ Having an index on a table makes inserts more efficient than inserts into a standard heap file.

   ☐ Alternate 2 B+ trees are useful for frequent full scans of a relation.

   ☐ Deleting a value from a B+ tree deletes it from any inner nodes it may be in as well.

   ☐ None of the above

(b) (9 points)

Up-and-coming boba store Boba Trees has a lot of information about their sales this year, and they want to use indices to speed up their queries.

Boba Trees has the following table `orders` containing information about their recent sales.

```
CREATE TABLE orders (
    sid INTEGER PRIMARY KEY,  // the sales id
    cid INTEGER,              // the customer id
    order TEXT,
    store_id INTEGER          // which of the store locations the order was at
);
```

We have 3 different indices on this table:

Index A:

- Alternative 2, clustered B+ Tree on `sid` with h = 2, d = 4
- There are 80 leaf nodes, all of which are full.
- The first record with `sid > 600` appears on the 70th leaf node (starting from the leftmost leaf node which is the 1st leaf node).

Index B:

- Alternative 3, unclustered B+ Tree on `store_id` with h = 2, d = 2
- There are 10 leaf nodes in total, and the first key with `store_id > 2` is on the second leaf node.

Index C:

- Alternative 3, unclustered B+ Tree on `cid` with h = 3, d = 3

Additional assumptions:

- The orders table has 100 data pages.
- There is exactly 1 record with cid = 50.
- There are 80 records with cid = 100.
- Assume that store_id has 20 unique values in the range [1,20], uniformly distributed.
- Each data page stores 20 records.

i. (1 pt) What is the maximum number of keys that Index A can hold?

ii. (2 pt) Calculate the minimum possible I/O cost for the following query: `SELECT * FROM orders WHERE order = 'Jasmine Milk Tea';`

iii. (2 pt) Calculate the minimum possible I/O cost for the following query: `SELECT * FROM orders WHERE cid = 50;`

iv. (2 pt) Calculate the minimum possible I/O cost for the following query: `SELECT * FROM orders WHERE cid = 100 AND sid > 600;`

v. (2 pt) If we wanted to do a range scan on `store_id` to find out how many orders we had at locations with `store_id` > 2, which index should we use, if any? Explain your answer.

(c) (4 points) Bulkloading

Uh oh! Boba Trees had an electrical fire in their store and lost Index A. They want to use bulkloading to quickly reconstruct index A; that is, a B+ tree on `sid` with d = 4. Assuming for this problem only that `orders` now contains 900 records and they use a fill factor of 0.75, answer the following questions.

i. (2 pt) How many leaf nodes would the reconstructed tree have?

ii. (2 pt) What is the minimum height of the reconstructed tree?

2. (10 points) Microtransactions and In-Game Concurrency

(a) (2 points) True or False

Mark all statements that are true.

i. (2 pt)

- ☐ In comparison to conflict serializability, view serializability is more efficient to check for complexity-wise.
- ☐ Locks help enforce the atomicity part of the ACID properties.
- ☐ The same graph can always be used for the conflict dependency graph and the waits-for graph.
- ☐ An advantage of Strict 2PL over regular 2PL is the prevention of cascading aborts.
- ☐ None of the above

(b) (3 points) Two-Phase Locking

Analyze the following schedules, and identify whether they follow two-phase locking. You may assume that all actions taken by the transactions are valid, and that the last action in the schedule for a transaction is the last action that transaction will take before it commits.

i. (1 pt)

Transaction 1	Transaction 2
Obtain IX(database)	
	Obtain IX(database)
Obtain S(table 1)	
Release S(table 1)	
	Obtain X(table 1)
Obtain X(table 2)	
	Release X(table 1)
	Release IX(database)
Release X(table 2)	
Release IX(database)	

◯ Follows 2PL

◯ Does not follow 2PL

ii. (1 pt)

Transaction 1	Transaction 2
Obtain IX(database)	
Obtain X(table 1)	
	Obtain IX(database)
Release X(table 1)	
	Obtain X(table 2)
Release IX(database)	
	Release X(table 2)
	Release IX(database)

◯ Follows 2PL

◯ Does not follow 2PL

iii. (1 pt)

Transaction 1	Transaction 2
Obtain IS(database)	
Obtain S(table 1)	
Release S(table 1)	
Release IS(database)	
	Obtain IX(database)
	Obtain X(table 1)
	Release X(table 1)
	Release IX(database)

◯ Follows 2PL

◯ Does not follow 2PL

(c) (5 points) Deadlocks

Use the schedule of lock requests below to answer the following questions. Assume transactions hold onto any locks they obtain for the entire length of the schedule. If a transaction aborts, assume that it remains inactive for the remainder of the schedule. Also assume that priority is assigned by the time of first operation (earlier operation = higher priority).

Transaction	1	2	3	4	5	6	7	8	9	10
T1		S(A)				S(B)			X(D)	
T2	X(B)						S(C)			
T3			X(C)	X(A)				X(D)		
T4					S(D)					S(C)

i. (1 pt) What transactions are deadlocked in this schedule? If there are multiple deadlocks, mark the transactions in the deadlock that will happen first.

☐ T1

☐ T2

☐ T3

☐ T4

☐ None of the above

ii. (1 pt) Assuming no use of deadlock avoidance algorithms, do any of the transactions in this schedule complete?

○ Yes

○ No

iii. (1 pt) Now assume that we use deadlock detection throughout the execution of the schedule. Which transaction should we abort to prevent further deadlocks for the rest of the schedule?

○ T1

○ T2

○ T3

○ T4

○ None of the above

iv. (1 pt) Now assume that we use wait-die deadlock avoidance. Which transactions are aborted?

☐ T1

☐ T2

☐ T3

☐ T4

☐ None of the above

v. (1 pt) Now assume that we use wound-wait deadlock avoidance. Which transactions are aborted?

☐ T1

☐ T2

☐ T3

☐ T4

☐ None

3. (15 points) EECS 16A anyone?

Alvin wants to use relations to store matrices. For instance, we will store the 3x2 matrix

$$M = \begin{bmatrix} 0 & 10 \\ 0 & 20 \\ 30 & 0 \end{bmatrix}$$

as the relation `M(INT row, INT col, INT val)`:

row	col	val
0	0	0
0	1	10
1	0	0
1	1	20
2	0	30
2	1	0

And the 3x1 vector

$$\vec{v} = \begin{bmatrix} 42 \\ 1 \\ 3 \end{bmatrix}$$

as the relation `v(INT row, INT val)`:

row	val
0	42
1	1
2	3

Recall a vector is a matrix with many rows but only one column. Also note the following matrix transformations:

- Transpose of a matrix: M^T is the row and col indices of M switched. For example,

$$M = \begin{bmatrix} 0 & 10 \\ 0 & 20 \\ 30 & 0 \end{bmatrix} \qquad M^T = \begin{bmatrix} 0 & 0 & 30 \\ 10 & 20 & 0 \end{bmatrix}$$

- Transpose of a vector: $\vec{v}^T$ is $\vec{v}$ but with columns instead of rows. For example,

$$\vec{v} = \begin{bmatrix} 42 \\ 1 \\ 3 \end{bmatrix} \qquad \vec{v}^T = \begin{bmatrix} 42 & 1 & 3 \end{bmatrix}$$

The following are the linear algebra operations that Alvin wants to run. Note that the first subscript letter represents the row number and the second subscript letter represents the column number (A_{ij} represents the value at ith row and jth column of A).

- Scalar-matrix sum: $C = s + A$, where $C_{ij} = s + A_{ij}$. s is an integer, and + is just addition of two integers.
- Scalar-vector sum: $\vec{c} = s + \vec{a}$, where $\vec{c_i} = s + \vec{a}_i$. s is an integer, and + is just addition of two integers.
- Matrix-vector product: $\vec{y} = A\vec{x}$, where $\vec{y_i} = \Sigma_j A_{ij}\vec{x}_j$. Notice that the output is a vector here.
- Vector-vector product: $\vec{x}^T\vec{y} = \Sigma_i \vec{x}_i\vec{y}_i$. Notice that the output is a scalar here.
- Matrix-matrix product: $C = AB$, where $C_{ik} = \Sigma_j A_{ij}B_{jk}$. Notice that the output is a matrix here.

- Trace of a matrix: $\text{tr}(A) = \Sigma_i A_{ii}$. Notice that the output is a scalar here.

In this problem relation `x` is the vector $\vec{x}$ and relations `A`, `B`, `C` are matrices A, B, and C, respectively. Assume all matrix multiplications are compatible.

(a) (9 points)

For each of the SQL expressions below, write the corresponding linear algebra operations for it only using the definitions above. For instance, if the query computes the trace of a matrix-matrix product between matrices A and B, write `tr(AB)` as the answer.

i. (2 pt)

```
SELECT SUM(A.val)
FROM A
WHERE A.col = A.row
```

Equivalent linear algebra operation:

◯ AA

◯ $\text{tr}(A)$

◯ $\text{tr}(AA)$

◯ None of the above

ii. (2 pt)

```
SELECT A.row, A.col, A.val + SUM(B.val * C.val)
FROM A, B, C
WHERE A.row = B.row AND B.col = C.row AND C.col = A.col
GROUP BY A.row, A.col, A.val
```

Equivalent linear algebra expression:

◯ $C + AB$

◯ $BC + A$

◯ ABC

◯ None of the above

iii. (2 pt)

```
SELECT SUM(A.val * x.val)
FROM A, x
WHERE A.col = x.row
GROUP BY A.row
```

Equivalent linear algebra expression:

◯ $\text{tr}(A) + \vec{x}$

◯ $\vec{x}^T A$

◯ $A\vec{x}$

◯ None of the above

iv. (3 pt)

```
SELECT SUM(A.val * x1.val * x2.val)
FROM A, x AS x1, x AS x2
WHERE x1.row = A.col AND A.row = x2.row
```

Equivalent linear algebra expression:

- ◯ $\vec{x}^T A \vec{x}$
- ◯ $A^T A \vec{x}$
- ◯ $\vec{x}^T \vec{x} A$
- ◯ None of the above

(b) (6 points)

Here is the matrix-relation schema again for your convenience.

$$M = \begin{bmatrix} 0 & 10 \\ 0 & 20 \\ 30 & 0 \end{bmatrix}$$

is the relation `M(INT row, INT col, INT val)`:

row	col	val
0	0	0
0	1	10
1	0	0
1	1	20
2	0	30
2	1	0

And the 3x1 vector

$$\vec{v} = \begin{bmatrix} 42 \\ 1 \\ 3 \end{bmatrix}$$

is the relation `v(INT row, INT val)`:

row	val
0	42
1	1
2	3

Alvin now wants to store the matrices and vectors to the disk. Assume the following:

- Integers and pointers are 4 bytes.
- Each page is 1024 bytes.
- Data pages follow the unpacked bitmap layout with fixed record size as presented in lecture.

i. (3 pt) What is the minimum number of pages it takes to store a 100x100 matrix? Write down the number of pages below. No need to show any calculation or explanation.

ii. (3 pt) Now suppose we change the table schema for a matrix A to be `A(BYTE row, BYTE col, INT val)` instead, where each `BYTE` integer is exactly 1 byte. What is the minimum number of pages it takes to store a 100x100 matrix? Write down the number of pages below. No need to show any calculation or explanation.

4. (21 points) Dogejoin and Parallel Equereum

(a) (12 points)

i. (1 pt) Suppose we have table D and table E with [D] = 50, [E] = 40, and number of buffer pages B = 12. What is the minimum number of I/Os it would take to perform a Block Nested Loop Join of D and E?

ii. (2 pt) Consider the same [D] = 50 and [E] = 40 as the previous question but suppose we can now add buffer pages. What is the minimum number of total buffer pages we can have so that a Block Nested Loop Join of D and E takes strictly fewer than 200 I/Os?

iii. (2 pt) Now assume [D] = 50, [E] = 40, and again B = 12. How many I/Os would an **unoptimized** Sort Merge Join take?

iv. (2 pt) With the same assumptions as the previous question, how many I/Os would a **fully optimized** Sort Merge Join take?

v. (3 pt) Now assume [D] = 50, B = 12, but we do not know [E]. We want to perform an optimized Sort Merge Join but also want to know when we can't apply the optimization. What is the minimum [E] that will prevent us from using the optimization from the previous question?

vi. (2 pt) Explain your answer to the previous question.

(b) (9 points)

i. (1 pt) Suppose again we have table D and table E but now both much larger with [D] = 500 and [E] = 750. If all pages start off on one machine and we want to perform a parallel Grace Hash Join, how much network cost in KB will we incur? Assume we have 10 machines, 1 page = 4 KB, all pages start off full, and hash partitioning is uniform.

ii. (3 pt) Building off the previous question, how many I/Os across all machines will parallel Grace Hash Join take after hash partitioning? Assume tuples are written directly to disk when coming off the network (no I/Os are incurred) and each machine has B = 11 buffer pages.

iii. (3 pt) Building off the previous two questions, we want to figure out how much time it will take to run the entire parallel Grace Hash Join. Assume it takes 1 ms to send 1 KB of data from one machine to another, a machine can send data to other machines in parallel, but each individual table must be transmitted serially (i.e. machine 1 can send both table D **and** E to all other machines at the same time but it can only send 1 KB of table D and 1 KB of table E to each machine every 1 ms), 1 I/O takes 5 ms, hash partitioning is uniform, all data starts on 1 machine, and the answer to the previous question is 5000 I/Os. How much time in ms will the entire parallel GHJ take?

iv. (2 pt) Now assume we again have 10 machines in total, table D starts off one 1 machine, [D] = 10, table E is round-robin partitioned across the 10 machines, and [E] is unknown. What is the minimum [E] such that a broadcast join with table D being sent to every machine will incur a lower network cost than a parallel Grace Hash Join with uniform partitioning?

5. (16 points) Disk/Files: I'm not ready to say good-byte T_T

(a) (1 pt) Write operations are typically more time consuming, so we typically only count the number of page writes to disk when we count I/Os.

○ True

○ False

(b) (1 pt) Fill in the blank. In variable length record formats where we use a record header, we store the fixed length fields __________ the variable length fields.

○ before

○ after

(c) (1 pt) Fill in the blank. In variable length record formats where we use a record header, we store the pointers to the __________ of the variable length fields.

○ beginning

○ end

(d) (3 pt) Suppose I have a packed, sorted page of 13 fixed length records. Each record is 10 bytes long. If I wish to delete the record with the second smallest key on the page, how many bytes will I need to change on the page? Assume all integers and pointers are 4 bytes, and that records are sorted in ascending order. Hint: consider what changes the header needs to make as well.

(e) (2 pt) An unpacked page with variable length records is 4 KiB (or 4096 bytes). Assume the page has 6 bytes of metadata, saving the remainder of the page for the records and page footer. Suppose the page is initially empty and then Jerry inserts two records. The first record is 20 bytes long, and the second is 40 bytes long. How many bytes of free space are remaining? Assume all integers and pointers are 4 bytes. Hint: Remember that the page footer contains pointer/length pairs, a pointer to free space, and a slot count.

(f) (2 pt) Suppose I have an unpacked page of variable length records that currently holds 20 different records and has 1000 bytes of free space on the page. If I delete one record with a size of 50 bytes, how many bytes of free space do I have immediately after? What will my slot count be immediately after? Separate your answers to these two questions with a comma and no spaces (ex. 1000,20).

(g) (1 pt) Which of the following pages may encounter fragmentation? Mark all that apply.

- ☐ Packed pages with fixed length records.
- ☐ Packed pages with variable length records.
- ☐ Unpacked pages with fixed length records.
- ☐ Unpacked pages with variable length records.
- ☐ None of the above.

(h) (2 pt) In a page directory implementation of heap files, we have 2021 data pages. Assume each header page holds information for up to 200 data pages. If we have the least number of header pages possible, what is the maximum number of I/Os it could take to insert a record assuming that at least one data page has room for said record? Assume you do not need to check for duplicate primary keys.

(i) (1 pt) Jerry is writing goodbye notes. As he writes each note, he enters it into a database. The schema for the notes table includes (1) the recipient name as the primary key for each note's database entry, (2) the note's message body and (3) the address of the recipient. After writing all notes, he plans to scan through them all and send each one out to the listed address. Which page and file format combination is best for his use case?

- ○ packed, heap file
- ○ unpacked, heap file
- ○ packed, sorted file
- ○ unpacked, sorted file

(j) (1 pt) Jerry realizes that he might like to update the notes after writing them but before sending them out. Thus, he would like to be able to quickly retrieve their corresponding note and make adjustments. Recall that the primary key of his database is the recipient name. Which page and file format combination is best for his use case?

- ○ packed, heap file
- ○ unpacked, heap file
- ○ packed, sorted file
- ○ unpacked, sorted file

(k) (1 pt) Jerry has received many notes as well! He decided to store the notes he received also in a database with the sender name as the primary key. In the next year, he wishes to read the note from a friend when that person's name pops into his head. Which page and file format combination is best for his use case?

- ○ packed, heap file
- ○ unpacked, heap file
- ○ packed, sorted file
- ○ unpacked, sorted file

6. (21 points) Query Optimus Prime

For this question, assume that each column's values are uniformly distributed, and that all of the column distributions are independent of each other.

Schema	Pages	Table Stats
CREATE TABLE **R**(a FLOAT, b INTEGER)	50 pages with 100 records each	N/A
CREATE TABLE **S**(a FLOAT, b INTEGER)	40 pages with 100 records each	**S.a:** min = 0, max = 40; **S.b:** min = 1, max = 20, 20 unique values
CREATE TABLE **T**(a FLOAT, b INTEGER)	30 pages with 100 records each	**T.a:** min = 0, max = 40

(a) (6 points) System R Query Optimizer - Pass 1

In the remaining problems, we will optimize the following query using the System R (aka Selinger) query optimizer. Additionally, assume that the **selectivity of `T.a <= 20` is 1/2**.

```
SELECT *
FROM R INNER JOIN S ON R.a = S.a
    INNER JOIN T ON S.a = T.a
WHERE R.a <= 50 AND S.a <= 20 AND T.a <= 20
ORDER BY R.b
```

i. (1 pt) Estimate the cost in I/Os of using a full scan as our access plan for T in Pass 1.

ii. (3 pt) For this problem, assume we have an unclustered, height 2, Alt. 3 B+ tree built on T.a with 20 leaf nodes. Estimate the cost in I/Os of using an index scan on T.a as our access plan for T in Pass 1.

iii. (2 pt) For this problem only, assume that we have the following I/O costs for single-table access plans:

Option	Access Plan	I/O Cost
a	R: Full scan	60 I/Os
b	R: Index scan on R.a	50 I/Os
c	R: Index scan on R.b	70 I/Os
d	S: Full scan	45 I/Os
e	S: Index scan on S.a	55 I/Os
f	S: Index scan on S.b	50 I/Os
g	T: Full scan	25 I/Os
h	T: Index scan on T.a	80 I/Os

Which of the following plans will be kept at the end of Pass 1?

☐ Option a

☐ Option b

☐ Option c

- ☐ Option d
- ☐ Option e
- ☐ Option f
- ☐ Option g
- ☐ Option h
- ☐ None of the above

(b) (13 points) System R Query Optimizer - Pass 2/3

Let's now look at passes 2 and 3 of the System R Query Optimizer. Assume that:

- We have **B = 7** buffer pages.
- R originally has 50 pages. There are **25 pages** for which `R.a <= 50`.
- S originally has 40 pages. There are **20 pages** for which `S.a <= 20`.
- T originally has 30 pages. There are **15 pages** for which `T.a <= 20`.

For your convenience, here is a copy of the query we are optimizing:

```
SELECT *
    FROM R INNER JOIN S ON R.a = S.a
           INNER JOIN T ON S.a = T.a
    WHERE R.a <= 50 AND S.a <= 20 AND T.a <= 20
    ORDER BY R.b
```

i. (3 pt) Using the System R query optimizer, suppose we choose the full scan on R and the full scan on S as our single table access plans. What is the estimated I/O cost of R BNLJ S (with R as the outer relation and S as the inner relation) including the cost of the full scans?

ii. (1 pt) Does the previous join (R BNLJ S, using full scans as the access plans for R and S) produce an interesting order?

◯ Yes

◯ No

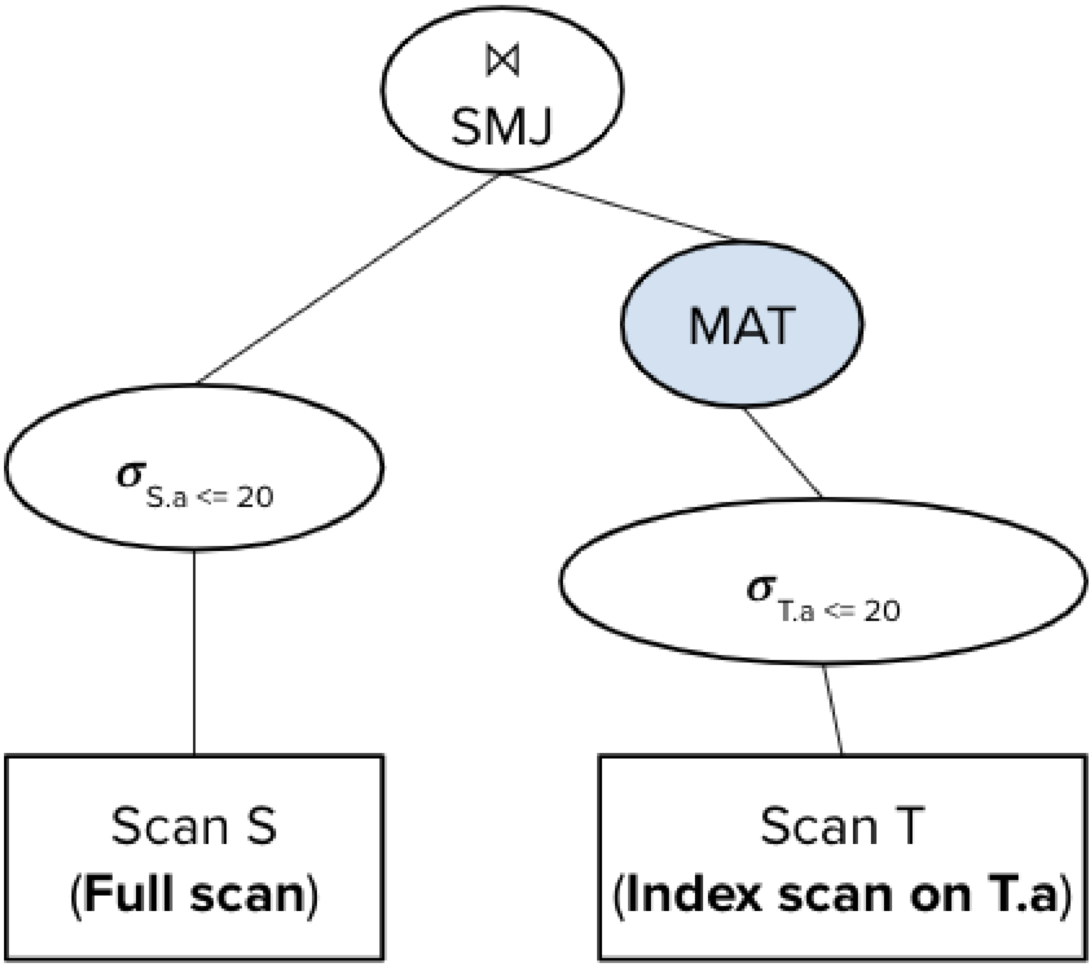
iii. (1 pt) For this problem only: suppose we use an index scan on R.b as our access plan for R, and a full scan as our access plan for S. Using these access plans, will R BNLJ S produce an interesting order?

◯ Yes

◯ No

iv. (3 pt) For this problem only: suppose we are using a different query optimizer, which uses the following query plan. This query optimizer differs from the System R query optimizer in that it chooses to materialize the output of the single-table access plan for T.

Additionally, assume performing an index scan on T using the index on T.a (which simultaneously applies the selection on T.a) costs **80 I/Os**, and applying the selection on S.a takes place during the first pass of sorting S in the SMJ algorithm.



Using this query plan, what is the average case I/O cost of S SMJ T using **unoptimized** SMJ?

v. (1 pt) Does the previous join (S SMJ T, using a full scan as the access plan for S and an index scan on T.a as the access plan for T) produce an interesting order?

- ◯ Yes
- ◯ No

vi. (2 pt) Assume that we have the following I/O costs for joining 2 tables together:

Option	Access Plan	Sorted on	I/O Cost
a	R $\bowtie$ S	N/A	300 I/Os
b	R $\bowtie$ S	R.b	400 I/Os
c	S $\bowtie$ R	S.a	500 I/Os
d	S $\bowtie$ T	N/A	350 I/Os
e	T $\bowtie$ S	T.a	400 I/Os
f	R $\bowtie$ T	R.b	500 I/Os
g	T $\bowtie$ R	N/A	450 I/Os

Using just this information, along with the query, which of the following plans will be kept at the end of Pass 2?

For your convenience, here is a copy of the query we are optimizing:

```
SELECT *
    FROM R INNER JOIN S ON R.a = S.a
           INNER JOIN T ON S.a = T.a
    WHERE R.a <= 50 AND S.a <= 20 AND T.a <= 20
    ORDER BY R.b
```

☐ Option a

☐ Option b

☐ Option c

☐ Option d

☐ Option e

☐ Option f

☐ Option g

☐ None of the above

vii. (2 pt) Assume that the access plans retained at the end of Pass 2 were R $\bowtie$ S and S $\bowtie$ T. Which of the following access plans will the System R query optimizer consider in Pass 3?

☐ (R $\bowtie$ S) $\bowtie$ T

☐ (S $\bowtie$ R) $\bowtie$ T

☐ (R $\bowtie$ T) $\bowtie$ S

☐ (S $\bowtie$ T) $\bowtie$ R

☐ T $\bowtie$ (R $\bowtie$ S)

☐ R $\bowtie$ (S $\bowtie$ T)

☐ S $\bowtie$ (R $\bowtie$ T)

☐ None of the above

(c) (2 points) Query Optimization in MongoDB (Brian's Guest Lecture)

i. (2 pt) Which of the following query optimization techniques are used to optimize the MongoDB aggregation pipeline?

- ☐ Moving $match stages to the top of the aggregation pipeline
- ☐ `$match` coalescing
- ☐ Avoiding cartesian products
- ☐ Dead code elimination
- ☐ None of the above

7. (15 points) A Well-Deserved Recovery

Oh no! Our database just crashed! Let's do some recovery. Assume for all parts that we're running ARIES as described and implemented in Project 5.

(a) (6 points) Analysis

Below is the **log recovered from disk when recovery starts**.

LSN	Record	PrevLSN
0	Master	-
10	T2 Updates P2	null
20	T1 Updates P1	null
30	T3 Updates P3	null
40	Commit T2	10
50	T1 Updates P1	20
60	End T2	40
70	T3 Updates P2	30
80	Abort T3	70
90	T1 Updates P4	50

The first step to performing recovery is to run the **Analysis Phase**. Answer the following questions about analysis .

i. (1 pt) Which of the following pages are in the dirty page table at the end of analysis?

☐ P1

☐ P2

☐ P3

☐ P4

☐ None of the above

ii. (1 pt) What is T1's status at the end of analysis?

◯ COMMITTING

◯ RECOVERY_ABORTING

◯ RUNNING

◯ T1 is not in the transaction table

iii. (1 pt) What is T2's status at the end of analysis?

◯ COMMITTING

◯ RECOVERY_ABORTING

◯ RUNNING

◯ T2 is not in the transaction table

iv. (1 pt) What is T3's status at the end of analysis?

◯ COMMITTING

◯ RECOVERY_ABORTING

◯ RUNNING

◯ T3 is not in the transaction table

v. (2 pt) The log that we recover from the disk contains records that have been flushed prior to the crash; however, there may have been other records that were not flushed. Which of the following are a possible set of log records for T1 before the crash? (Select all that apply)

LSN	Record	PrevLSN
20	T1 Updates P1	null
50	T1 Updates P1	20
90	T1 Updates P4	50
100	T1 Updates P1	90
110	T1 Updates P4	100

☐

LSN	Record	PrevLSN
20	T1 Updates P1	null
50	T1 Updates P1	20
90	T1 Updates P4	50
100	T1 Updates P1	90
110	Commit T1	100
120	End T1	110

☐

LSN	Record	PrevLSN
20	T1 Updates P1	null
50	T1 Updates P1	20
90	T1 Updates P4	50
100	T1 Updates P1	90
110	Abort T1	100

☐

LSN	Record	PrevLSN
20	T1 Updates P1	null
50	T1 Updates P1	20
90	T1 Updates P4	50
100	T1 Updates P1	90
110	Abort T1	100
120	End T1	110

☐

- ☐ None of the above

(b) (6 points) Redo

This next section is independent from the previous part. Suppose we have a **different** log shown below.

LSN	Record	PrevLSN
0	Master	-
10	T1 Updates P4	null
20	T3 Updates P2	null
30	T3 Updates P3	20
40	Begin Checkpoint	-
50	T1 Updates P4	10
60	T3 Updates P2	30
70	T2 Updates P1	null
80	End Checkpoint	-
90	Commit T1	50
100	Abort T3	60
110	T2 Updates P1	70
120	CLR T3 LSN 60, undoNextLSN=30	100

Assume after running analysis on this log, we get the following transaction table and dirty page table.

Transaction Table

Transaction	Status	LastLSN
T2	RECOVERY_ABORTING	110
T3	RECOVERY_ABORTING	120

Dirty Page Table

Page Number	RecLSN
P1	70
P2	60
P4	50

The next step is the **Redo Phase**. For the following parts, assume pageLSN(disk) < LSN always.

i. (1 pt) What LSN does Redo begin at?

ii. (2 pt) Select the LSNs of all of the operations that are redone during the Redo phase.

☐ 10

☐ 20

☐ 30

☐ 40

☐ 50

☐ 60

☐ 70

☐ 80

☐ 90

☐ 100

☐ 110

☐ 120

☐ None of the above

iii. (2 pt) At LSNs 40-80, we perform a fuzzy checkpoint. Which of the following are possible transaction tables stored in the checkpoint record? (Select all that apply)

Transaction	Status	LastLSN
T1	RUNNING	10
T3	RUNNING	20

☐

Transaction	Status	LastLSN
T1	RUNNING	50
T3	RUNNING	60

☐

Transaction	Status	LastLSN
T1	RUNNING	50
T2	RUNNING	70
T3	RUNNING	60

☐

Transaction	Status	LastLSN
T1	RUNNING	90
T2	RUNNING	70
T3	RUNNING	60

☐

☐ None of the above

iv. (1 pt) What is a possible reason why P3 is not in the dirty page table when we begin analysis even though it was modified at LSN 30? Briefly explain in 1 sentence. (Hint: When do we remove pages from the dirty page table?)

(c) (3 points) Undo

The log, transaction table, and dirty page table from the previous part are copied below for convenience.

LSN	Record	PrevLSN
0	Master	-
10	T1 Updates P4	null
20	T3 Updates P2	null
30	T3 Updates P3	20
40	Begin Checkpoint	-
50	T1 Updates P4	10
60	T3 Updates P2	30
70	T2 Updates P1	null
80	End Checkpoint	-
90	Commit T1	50
100	Abort T3	60
110	T2 Updates P1	70
120	CLR T3 LSN 60, undoNextLSN=30	100

Transaction Table

Transaction	Status	LastLSN
T2	RECOVERY_ABORTING	110
T3	RECOVERY_ABORTING	120

Dirty Page Table

Page Number	RecLSN
P1	70
P2	60
P4	50

The final step is the **Undo Phase**.

i. (1 pt) Which of the following LSNs are in the initial priority queue?

- ☐ 10
- ☐ 20
- ☐ 30
- ☐ 40
- ☐ 50
- ☐ 60
- ☐ 70
- ☐ 80
- ☐ 90
- ☐ 100
- ☐ 110
- ☐ 120
- ☐ None of the above

ii. (2 pt) Which of the following records are undone?

- ☐ 10
- ☐ 20
- ☐ 30
- ☐ 40
- ☐ 50
- ☐ 60
- ☐ 70
- ☐ 80
- ☐ 90
- ☐ 100
- ☐ 110
- ☐ 120
- ☐ None of the above

8. (16 points) No Shirt, No Shoes, NoSQL

(a) (7 points) True/False

For the following 7 questions, indicate whether the statement is true or false.

i. (1 pt) Retrieval queries in MQL can be used to perform operations similar to INNER JOINs and GROUP BYs in SQL.

◯ True

◯ False

ii. (1 pt) In MapReduce, all mappers must complete their work before the reduce phase can start.

◯ True

◯ False

iii. (1 pt) In MapReduce, small intermediate results do not have to be written to disk.

◯ True

◯ False

iv. (1 pt) In Spark, transformations are executed immediately while actions are executed lazily.

◯ True

◯ False

v. (1 pt) The principle of atomicity is part of both ACID and BASE.

◯ True

◯ False

vi. (1 pt) In addition to primitives, JSON also allows for collection types like ordered lists, unordered sets, and maps.

◯ True

◯ False

vii. (1 pt) Unlike Spark DataFrames, Spark Datasets are unordered and don't allow duplicate objects.

◯ True

◯ False

(b) (7 points) Multiple Guess

i. (1 pt) MapReduce utilizes which of the following types of parallelism?

◯ Inter-Query Parallelism

◯ Inter-Operator Parallelism

◯ Intra-Operator Parallelism

ii. (2 pt) For the following question, consider the map and reduce functions and select which SQL query would be equivalent. Assume map is applied to tuples from relation R with the integer valued columns A, B, and C.

```
map(Tuple t):
    EmitIntermediate(t.B, t.A)
```

```
reduce(Integer B, Iterator values):
    r = Integer.MIN_VALUE
    for each v in values:
       if r < v:
             r = v
    Emit(B, r)
```

◯ `SELECT MAX(A) FROM R GROUP BY B;`

◯ `SELECT * FROM R WHERE B=186;`

◯ `SELECT B FROM R;`

◯ `SELECT COUNT(*) FROM R;`

◯ None of the above

iii. (2 pt) For the following question, consider the map and reduce functions and select which SQL query would be equivalent. Assume map is applied to tuples from relation R with the integer valued columns A, B, and C.

```
map(Tuple t):
    EmitIntermediate(186, t.B)
```

```
reduce(Integer N, Iterator values):
    for each v in values:
        Emit(v)
```

◯ `SELECT B, MIN(A) FROM R GROUP BY B;`

◯ `SELECT * FROM R WHERE B=186;`

◯ `SELECT B FROM R;`

◯ `SELECT COUNT(*) FROM R;`

◯ None of the above

iv. (2 pt) For the following question, consider the map and reduce functions and select which SQL query would be equivalent. Assume map is applied to tuples from relation R with the integer valued columns A, B, and C.

```
map(Tuple t):
    if  t.B = 186:
        EmitIntermediate(t.B, t);
```

```
reduce(Integer A, Iterator values):
    for each v in values:
        Emit(v);
```

- ◯ `SELECT MIN(A) FROM R GROUP BY B;`
- ◯ `SELECT * FROM R WHERE B=186;`
- ◯ `SELECT B FROM R;`
- ◯ `SELECT COUNT(*) FROM R;`
- ◯ None of the above

(c) (2 points) Short Answer

i. (1 pt) Consider the MongoDB collection `stuff` with the following documents:

```
{a: 61,  b: [1,2]}
{a: 82,  b: [3]}
{a: 123, b: [4,5,6]}
{a: 220, b: [7,8,9,10]}
```

How many documents would the following query yield? (Your answer should be an integer)

```
db.stuff.aggregate([
    {$match: { a: {$gte: 70}}},
    {$unwind: "$b"}
]);
```

ii. (1 pt) Consider the MongoDB collection `stuff` with the following documents:

```
{a: 61,  b: [1,2]}
{a: 82,  b: [3]}
{a: 123, b: [4,5,6]}
{a: 220, b: [7,8,9,10]}
```

The following query yields a single document with the field `thing`. What value would the `thing` field of the resulting document be? (Your answer should be an integer)

```
db.stuff.aggregate([
    {$sort: {a: -1}},
    {$limit: 2},
    {$group: {_id: null, thing: {$sum: {$first: "$b"}}}}
]);
```

9. (16 points) Sorted Eggs and Hash Browns

(a) (3.5 points) I/Os and Dollars

Your company is switching compute plans, and they need your help to figure out the most cost-efficient way to process your data. Each buffer page costs $10, but the cost of each IO is variable. You're given a choice between three plans.

Plan 1: 100 buffer pages, $0.01 per IO
Plan 2: 50 buffer pages, $0.05 per I/O
Plan 3: 10 buffer pages, $0.25 per I/O

Your company decides to go with a trial plan first and gives a small table to sort to see how it goes. With a file size of N=100 pages...

i. (1 pt) In sorting the file, how many I/Os are incurred under plan 1?

ii. (1 pt) How many I/Os are incurred under plan 2?

iii. (1 pt) How many I/Os are incurred under plan 3?

iv. (0.5 pt) What plan is cheapest for the N=100 pages scenario?

◯ PLAN 1

◯ PLAN 2

◯ PLAN 3

(b) (3.5 points)

Your company is happy with the results and decides to now try sorting a file size of N=10,000 pages.

i. (1 pt) How many I/Os are incurred under plan 1?

ii. (1 pt) How many I/Os are incurred under plan 2?

iii. (1 pt) How many I/Os are incurred under plan 3?

iv. (0.5 pt) What plan is cheapest for the N=10,000 pages scenario?

- ◯ PLAN 1
- ◯ PLAN 2
- ◯ PLAN 3

(c) (7 points)

Your company also inquires about hashing a smaller table of theirs, where N=100. Assume uniform hashing with a perfect hash function.

i. (2 pt) How many I/Os will this take to hash this table under Plan 1?

ii. (2 pt) How many I/Os does it take under Plan 2?

iii. (2 pt) How many I/Os does it take under Plan 3?

iv. (1 pt) What is the cheapest plan?

- ◯ PLAN 1
- ◯ PLAN 2
- ◯ PLAN 3

(d) (2 points) True or False

Mark all statements that are true.

i. (2 pt)

- ☐ Hashing will always result in the same number of I/Os writing to disk as the I/Os reading from disk.
- ☐ Sorting will always result in the same number of I/Os writing to disk as the I/Os reading from disk.
- ☐ You can only build an in-memory hash table if you have B-1 pages as your partition file size, where B is the number of pages in the buffer.
- ☐ Assuming uniform data spread and no key skew, with B pages of memory and a file size of N pages, hashing a table will always be at least as costly as sorting it.
- ☐ None of the above

10. (18 points) 2 Faze (Commit) 2 Furious

(a) (2 points) True or False

For the following 4 questions, select whether the statement is True or False.

i. (0.5 pt) Presumed abort always results in fewer messages sent across the network than regular 2PC.

◯ True

◯ False

ii. (0.5 pt) Presumed abort always results in fewer flushed log writes than regular 2PC.

◯ True

◯ False

iii. (0.5 pt) Machine M1 can be the coordinator for transaction $T1$ at the same time that machine M2 is the coordinator for transaction $T2$.

◯ True

◯ False

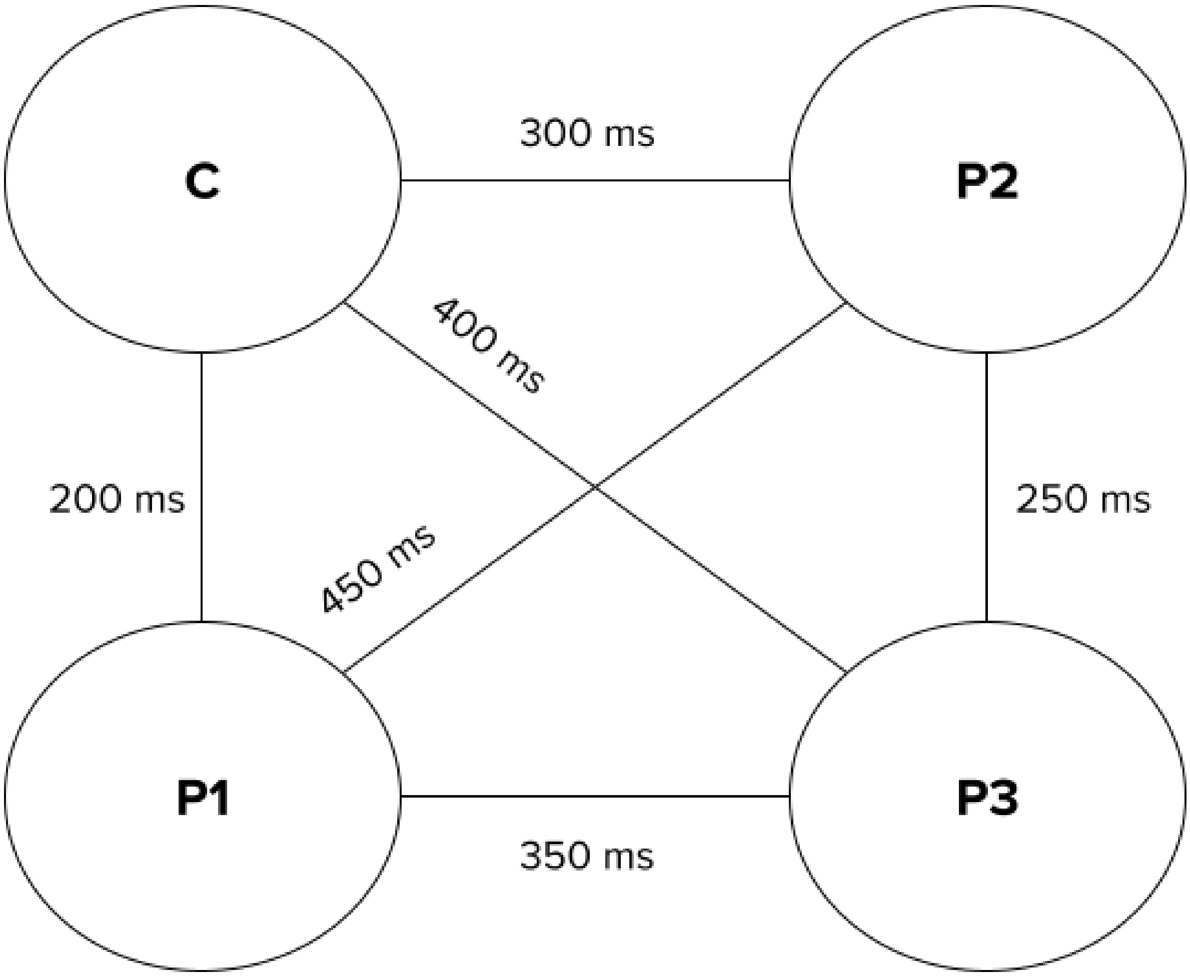
iv. (0.5 pt) If all participants crash at the same time, the transaction will always abort.

◯ True

◯ False

(b) (16 points)

Our database runs 2 Phase Commit **without any optimizations** with coordinator C and 3 participants: $P1$, $P2$, and $P3$. The communication latencies to send a message between each machine are shown in the diagram below. All latencies are in milliseconds. The latencies are symmetric (i.e. it takes 200 ms to send a message from C to $P1$ and also to send a message from $P1$ to C).



Assume the following information:

- Flushing a log record to disk takes 50 ms. All other operations (except sendings messages) occur instantaneously.
- Timeouts occur after 2500 ms.
- All participants vote YES.
- When a participant wants to ask the coordinator about the status of a transaction, it sends a STATUS INQUIRY message.

For the rest of this question, consider the following chronological sequence of events. All event times are shown in milliseconds since the 2PC round began.

- 0 ms: C begins the 2PC round.
- 500 ms: $P3$ crashes.
- 1000 ms: $P3$ recovers.
- 1500 ms: C crashes.
- 2000 ms: C recovers.

NOTE: ***A coordinator sending a message to all participants at the same time counts as sending ONE message.***

For questions that ask at what time an event occurs, provide the time in milliseconds since the 2PC round began and input ONLY the number.

i. (1 pt) What is the 1st message C receives?

○ PREPARE

○ COMMIT

○ ACK

○ STATUS INQUIRY

○ VOTE YES

○ VOTE NO

ii. (1 pt) What is the latest time at which C receives VOTE YES? Put NONE if C never receives VOTE YES.

iii. (1 pt) For this question only, assume the answer to the previous question is 750.

What is the earliest time at which C sends COMMIT? Put NONE if C never sends COMMIT.

iv. (1 pt) Select all of the following that belong in the blank.

At time _ _ _ _ , C receives ACK.

☐ 1350

☐ 1550

☐ 2000

☐ 2400

☐ None of the above

v. (1 pt) Select all of the following that belong in the blank.

At time _ _ _ _ , C sends COMMIT.

☐ 850

☐ 2000

☐ 2200

☐ 2400

☐ None of the above

vi. (1 pt) For this question only, assume C sends 4 COMMIT messages in total.

How many messages does C send in total?

vii. (1 pt) Select all of the following that belong in the blank.

At time ____ , $P1$ begins to flush a log record to disk.

- ☐ 100
- ☐ 200
- ☐ 1100
- ☐ 2200
- ☐ None of the above

viii. (1 pt) How many messages does $P1$ receive in total?

ix. (1 pt) For this question only, assume the answer to the previous question is 6.

How many messages does $P1$ send in total?

x. (1 pt) Select all of the following that belong in the blank.

At time ____ , $P2$ begins to flush COMMIT to disk.

- ☐ 300
- ☐ 500
- ☐ 1200
- ☐ 2300
- ☐ None of the above

xi. (1 pt) Select whether the following statement is True or False.

$P2$ sends more messages than $P1$.

- ○ True
- ○ False

xii. **(1 pt)** Select all of the following that belong in the blank.

After time 500, $P3$ sends ____.

☐ PREPARE

☐ COMMIT

☐ ACK

☐ STATUS INQUIRY

☐ VOTE YES

☐ VOTE NO

☐ $P3$ sends no messages after time 500.

xiii. **(1 pt)** What machine sends the final message?

◯ C

◯ $P1$

◯ $P2$

◯ $P3$

xiv. **(1 pt)** What is the earliest time at which an END log record can be emitted?

xv. **(2 pt)** Consider an optimization to 2PC, where the coordinator flushes an ACK log record everytime it receives an ACK from a participant. In case the coordinator crashes, it will know upon recovery which participants have already sent an ACK.

Using this optimization, how many milliseconds earlier can an END log record be emitted in comparison to the regular case without the optimization.

For example, if this optimization allows an END log record to be emitted at 4000 and the answer to the previous the question is 4500, your answer would be 500.

No more questions.