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 180 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: _______________________________________________________________

<table>
<thead>
<tr>
<th>Problem</th>
<th>Points</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>100</strong></td>
</tr>
</tbody>
</table>
1. State whether the following statements are True or False.

- As the number of copies increases, the write availability of the Read-One/Write-All (ROWA) algorithm increases.
  
  **FALSE**

- In a Chord ring with no failures, searches using successor links only are usually slower than searches using the finger table.
  
  **TRUE**

- It is more straightforward to execute complex queries in neighborhood search (i.e. Gnutella) compared to Distributed Hash Tables (i.e. Chord).
  
  **TRUE**

- Data processed by MapReduce must be relational (rows and columns).
  
  **FALSE**

- In Spark, defining a Resilient Distributed Dataset (RDD) causes it to be eagerly materialized.
  
  **FALSE**

2. Answer the following questions:

- What is one advantage of a small TTL in Gnutella searches? What is one disadvantage?

  **Advantage:** avoids the cost of flooding the network  
  **Advantage:** ensures your search completes quickly  
  **Disadvantage:** limits the portion of the network you search, so you may not find results
• Consider a network consisting of nodes, where some nodes have high bandwidth connections, and some nodes have low bandwidth connections. Which P2P topology is likely best for this network and why: neighborhood search, supernodes, or Chord?

   **Answer:** Supernodes is best, because the nodes with high bandwidth connections can do all of the expensive neighborhood search, while low bandwidth nodes can just connect to a supernode.

• What is the difference between a minor compaction and a merge compaction in BigTable? Why do we need merge compactions?

   **Answer:** A minor compaction writes a memlayer to disk, while a merge compaction combines multiple layers into one layer. A merge compaction is needed to keep the number of layers small, which is important for performance and storage usage.

• Why can’t we easily use a traditional database join in stream processing systems? What do we use instead?

   **Answer:** A regular join requires us to store all tuples that might join, which might be unbounded and thus impossible to store. Instead we usually use a “window join.”
3. Consider the MapReduce paradigm.

   a) It is usually easier to achieve load balanced mappers compared to reducers. Why?

      Answer: We can arbitrarily shard the input data among mappers to achieve load balancing. We have less control over the sharding of the intermediate data, which must have all data for a given reduce key (and these shards may have skewed sizes.) Also, the number of reducers we can use is bounded by the number of reduce keys, which may make it hard to load balance by assigning multiple reduce shards to a given reduce worker.

   b) Imagine we want to join two tables, R and S, where R is 100 TB in size and S is 2 GB in size. Would I prefer a map-side join or a reduce-side join? Why?

      Answer: Map side join. Since S is small, we can replicate it easily to all the mappers, and avoid shuffling the large R table. A reduce-side join would require shuffling R.
4. Consider pub/sub systems.

a) Imagine we have the following topic hierarchy:

1. NorthAmerica
   1.1. USA
      1.1.1. California
      1.1.2. NewYork
      1.1.3. Colorado
      1.1.4. Arizona
   1.2. Canada
   1.3. Mexico
2. Europe
   2.1. France
   2.2. England
   2.3. Germany
      2.3.1. Bavaria
      2.3.2. Saxony
      2.3.3. Saarland
3. Asia
   3.1. China
      3.1.1. Shandong
      3.1.2. Henan
   3.2. Japan
   3.3. Thailand

And the following subscriptions:
   S1: \{NorthAmerica.USA.California, Europe.Germany.Saarland\}
   S2: \{Asia.China, Asia.Japan\}
   S3: \{Europe\}

State, for each publication, which subscriptions it matches:

- P1: \{NorthAmerica.USA\}
  None

  S2

  S2, S3

  S1, S3
b) Consider the strawman architecture for Twitter we discussed in class. Define the user’s “feed” as the most recent tweets from the users they follow. Imagine we primarily want to minimize the response time of constructing a user’s feed. Would we prefer “pubs by user” or “notify by user” storage? Why?

Answer: We prefer notify by user, because the set of tweets in the feed are already chosen and collected in one place. With pubs by users, we have to visit all of the tweet lists of the users that are followed and collect the most recent tweets.

5. Recall the timestamp ordering concurrency control mechanism.

a) Imagine we are using the Thomas Write Rule, and we have:

\[ \text{MAX}_R[\text{‘foo’}] = 3\text{pm} \]
\[ \text{MAX}_W[\text{‘foo’}] = 5\text{pm} \]

Can we allow a transaction with timestamp 2pm to write ‘foo’? Why or why not?

Answer: No, 2pm is less than MAX_R

b) Imagine we are still using the Thomas Write Rule, and we have:

\[ \text{MAX}_R[\text{‘foo’}] = 5:30\text{pm} \]
\[ \text{MAX}_W[\text{‘foo’}] = 5:15\text{pm} \]

Can we allow a transaction with timestamp 5:20pm to write ‘foo’? Why or why not?

Answer: No, 5:20pm is less than MAX_R
6. Imagine we have the following relations (primary keys underlined):

Students(StudentId, FirstName, LastName, Major, StartYear, HomeState)
Registration(StudentId, CourseId, Quarter)

Registration.StudentId is a foreign key referring to Students.StudentId. We have the following fragments of the Students relation (where < means “lexicographically smaller than” and >= means “lexicographically larger than”):

Site1: $\sigma$ Major < 'Computer Science' Students
Site2: $\sigma$ Major >= 'Computer Science' AND Major < 'English' Students
Site3: $\sigma$ Major >= 'English' Students

We fragment Registration to the same sites, using a derived horizontal fragmentation using the same predicates. State which fragments the following queries execute at:

a. SELECT * FROM Students WHERE StartYear = 2014 AND Major = 'Geology';

Answer: Site3

b. SELECT * FROM Registrations NATURAL JOIN Students WHERE Major <= 'Drama' AND Quarter = 'Spring2015';

Answer: Site1 and Site2

c. SELECT * FROM Registrations WHERE CourseId = 'CS101';

Answer: All sites
7. Consider a distributed information retrieval system consisting of an inverted index hosting 16 terms ($t_1$ to $t_{16}$). The queries that our system receives are pairs of terms. Specifically, we receive 10 queries per second for each pair ($t_i$, $t_{i+1}$) for all $i$ in $[1,15]$. (That is, 10 queries per second for pair ($t_1$, $t_2$), 10 queries per second for pair ($t_2$, $t_3$), and so on.) In addition, we receive 15 queries per second for each pair ($t_i$, $t_{i+4}$) for all $i$ in $[1,12]$.

We want to horizontally fragment the inverted index across 4 nodes. The cost we pay for a query ($t_a$, $t_b$) when both terms $t_a$, $t_b$, are hosted under the same fragment/node, is 0. Otherwise, if the two terms are hosted under different fragments, we pay a cost of 1 unit, for each query. For example, if terms $t_1$, $t_2$ are hosted under different fragments, the cost per second coming from ($t_1$, $t_2$) is 10 (since we receive 10 of those queries per second).

### a) What is the overall cost (per second) for the following fragmentation:

<table>
<thead>
<tr>
<th>Fragment 1</th>
<th>Fragment 2</th>
<th>Fragment 3</th>
<th>Fragment 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1,t_2,t_3,t_4$</td>
<td>$t_5,t_6,t_7,t_8$</td>
<td>$t_9,t_{10},t_{11},t_{12}$</td>
<td>$t_{13},t_{14},t_{15},t_{16}$</td>
</tr>
</tbody>
</table>

**Answer**

Consider Figure 1, that visualizes the above fragmentation: each node refers to a term (in order from $t_1$ to $t_{16}$), each edge to a query, nodes with the same color are under the same fragment, and we pay for the edges with red color (between nodes with different colors), while we do not pay for they edges with a light grey color. If we sum the cost (per second) over all red edges we get: $12 \times 15 + 3 \times 10 = 210$

**Cost:** 210

---

Figure 1. Fragmentation in (a)
b) Give a fragmentation with exactly 4 terms per fragment that has an overall cost (per second) lower or equal to 130:

Answer

<table>
<thead>
<tr>
<th>Fragment 1</th>
<th>Fragment 2</th>
<th>Fragment 3</th>
<th>Fragment 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1, t_2, t_5, t_6 )</td>
<td>( t_3, t_4, t_7, t_8 )</td>
<td>( t_9, t_{10}, t_{13}, t_{14} )</td>
<td>( t_{11}, t_{12}, t_{15}, t_{16} )</td>
</tr>
</tbody>
</table>

Consider Figure 2, that visualizes the fragmentation in our solution (same conventions as in Figure 1). If we sum the cost (per second) over all red edges we get: \( 4 \times 15 + 7 \times 10 = 130 \)

Figure 2. Our solution for (b)

Cost: 130

Consider a distributed database of four nodes A, B, C, D. Moreover, consider the commit quorum \( CQ_1 = \{ \{A,B\}, \{A,C\}, \{C,D\} \} \). (In this context, a commit quorum is a set of elements; each element is a set of nodes that can agree to commit. For example, if both A and B agree to commit, or A and C, or C and D, then we will commit.)

a. We say that an abort quorum \( AQ \) matches a commit quorum \( CQ \), when each of \( AQ \)'s elements (sets) intersects with all of the \( CQ \)'s elements (sets). Which of the following abort quorums match \( CQ_1 \) (indicate using M for matching and nonM for non-matching)?

\[
\begin{align*}
AQ_1 &= \{ \{B,C\} \} & \text{M} \\
AQ_2 &= \{ \{A,D\} \} & \text{M} \\
AQ_3 &= \{ \{A,B\}, \{A,C\} \} & \text{nonM} \\
AQ_4 &= \{ \{B,C\}, \{B,D\} \} & \text{nonM} \\
AQ_5 &= \{ \{A,D\}, \{A,C\} \} & \text{M} \\
AQ_6 &= \{ \{A,C\}, \{A,D\}, \{B,D\} \} & \text{nonM}
\end{align*}
\]
AQ7 = \{\{A, D\}, \{B, D\}, \{C, D\}\} \hspace{1cm} \text{nonM}
AQ8 = \{\{B, C\}, \{A, D\}, \{C, D\}\} \hspace{1cm} \text{nonM}

b. The most flexible (matching) abort quorum (MFMAQ) for a given commit quorum CQ, is the one that includes all possible subsets of nodes involved in CQ (the nodes are A, B, C, D in case of CQ1), such that each subset intersects with all sets of CQ. In addition, in the MFMAQ, there are no sets that are supersets of other sets (in the MFMAQ). Which is the MFMAQ for CQ1 (it doesn't have to be one of the quorums in (a))?

Answer: MFMAQ = \{\{B, C\}, \{A, D\}, \{A, C\}\}