Assignment 1: Recap of Standard Database Issues  
(1 P.)

(a) Transactions
  - What are the ACID properties? Define them and illustrate through examples how each property contributes to the reliability of database transactions.
  - Specify an example of a lost update, a phantom read, and a dirty read, by defining two transactions \( T_1 \) and \( T_2 \) and, for each problem, create an example schedule.
  - Describe the Two Phase Locking (2PL) protocol. How can it be modified to prevent cascading aborts?
  - Is the following schedule of read and write operations conflict serializable? Explain what conflict serializability means and how one can decide if a schedule is conflict serializable or not.

\[
r_1(a)r_2(b)r_3(a)w_1(b)w_3(c)r_5(b)w_2(d)w_4(e)r_5(d)r_4(b)w_3(a)r_3(e)
\]

(b) Relational Algebra and SQL

Consider the following relational schema, where \( \rightarrow \) expresses a foreign key relationship:

- Person(PID, firstName, lastName, age)
- Car(CID, name, year, model, price)
- Mechanic(PID \( \rightarrow \) Person.PID, city)
- Repairs(PID \( \rightarrow \) Person.PID, car \( \rightarrow \) Car.CID, date, reason, amount)
- Owns(owner \( \rightarrow \) Person.PID, car \( \rightarrow \) Car.CID, purchaseDate)

Given the following expressions in relational algebra, check their correctness. If they are incorrect, briefly state the reason.

1. \( \pi_{firstName,lastName}(\sigma_{year<2010}(\pi_{firstName}(Person) \cup \pi_{lastName}(Person))) \)
2. \( \text{Person \bowtie \text{Person.lastName}=Mechanic.city \pi_{city='K-Town'}(Mechanic)} \)

- Translate the following queries into SQL and relational algebra expressions:
  1. What are the PIDs, last names and ages of the persons that are younger than 25 years?
  2. What are the cities in which at least one car has been repaired because of broken windshield?
  3. What are the first names and ages of mechanics who repaired a Mercedes S550 on April 15, 2015?
  4. List all cars whose price is lower than 5000 and whose sum of repairs cost is larger than their price.

(c) Indices

Given the index definition (of type B+ Tree) in respect to the subsequent queries, elaborate whether the index is suitable? In case the given index is not suitable or not “optimal”, are there better suited index definitions?

\$1, \$2 \text{ and PATTERN are each the parameters (variables) of the queries.}

1. CREATE INDEX index1 ON table1 (a, datum);

\[
\begin{align*}
\text{SELECT id, a, datum} \\
\text{FROM table1} \\
\text{WHERE a = $1} \\
\text{ORDER BY datum DESC} \\
\text{LIMIT 1;}
\end{align*}
\]
2. CREATE INDEX index2 ON table2 (a, b);

```sql
SELECT id, a, b 
FROM table2 
WHERE a = $1 
AND b = $2;
```

```sql
SELECT id, a, b 
FROM table2 
WHERE b = $1;
```

3. CREATE INDEX index3 ON table3 (name);

```sql
SELECT id, name 
FROM table3 
WHERE name LIKE '%PATTERN%';
```

(d) Result Sizes

Given two relations R and S with |R| = m and |S| = n, specify the min and max number of results for the following operations:

- \( R \bowtie S \)
- \( R \times S \)
- \( \sigma(S) \)

Assignment 2: MapReduce Fundamentals (1 P.)

1. Describe what each of the following MapReduce jobs does.

(a) map(nr, txt)

```java
words = split (txt, ', ')
for (i=0; i < |words| - 1; i++)
    emit(words[i] + ' ' + words[i+1], 1)
```

reduce(key, vals)

```java
s=0
for v : vals
    s += v
if(s > 100)
    emit(key, s)
```

(b) map(nr, txt)

```java
words = split (txt, ', ')
for (i=0; i < |words|; i++)
    emit(txt, length(words[i]))
```

reduce(key, vals)

```java
s=0
```
2. Consider the following combiner functions. Each one corresponds to the MapReduce jobs given in the first part of the assignment. Would these functions influence the final result of the MapReduce jobs and would they improve the performance of the tasks? Explain your answer.

(a) \text{combine(key, vals)}
\begin{align*}
& s = 0 \\
& \text{for } v : \text{vals} \\
& \quad s += v \\
& \text{emit(key, s)}
\end{align*}

(b) \text{combine(key, vals)}
\begin{align*}
& s = 0 \\
& c = 0 \\
& \text{for } v : \text{vals} \\
& \quad s += v \\
& \quad c += 1 \\
& \text{if } (s >= 2) \\
& \quad \text{emit(key, s/c)}
\end{align*}

(c) \text{combine(key, vals)}
\begin{align*}
& s = 0 \\
& \text{for } v : \text{vals} \\
& \quad s += 1 \\
& \quad \text{if } (s >= 2) \\
& \quad \text{emit(key, vals)}
\end{align*}

Assignment 3: Access Log Analysis with MapReduce (1 P.)

The MapReduce framework is particularly suitable for processing log files. Given an access log file, structured like the one in the table below:

- write a MapReduce job that will output all the ip addresses that have been accessed more than 1000 times within 60 minutes starting on the hour (e.g. from 15:00-15:59).
- write a MapReduce job that will output the ip addresses that have been accessed more than 1000 times within any 60 minutes time frame (e.g. 14:25-15:25).

<table>
<thead>
<tr>
<th>Logid</th>
<th>IPAddress</th>
<th>Date</th>
<th>Time</th>
<th>Site</th>
<th>Search Engine</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>192.188.20.99</td>
<td>11/2/2013</td>
<td>19:22:00</td>
<td>/news/</td>
<td>Google</td>
</tr>
<tr>
<td>101</td>
<td>168.222.22.22</td>
<td>11/2/2013</td>
<td>13:45:00</td>
<td>/wetter/index.html</td>
<td>Yahoo</td>
</tr>
<tr>
<td>102</td>
<td>189.999.99.99</td>
<td>11/2/2013</td>
<td>13:23:00</td>
<td>/home/</td>
<td>Google</td>
</tr>
<tr>
<td>103</td>
<td>187.788.222.20</td>
<td>11/9/2013</td>
<td>16:11:00</td>
<td>/home/</td>
<td>Bing</td>
</tr>
<tr>
<td>104</td>
<td>199.299.29.29</td>
<td>11/2/2013</td>
<td>19:09:00</td>
<td>/news/</td>
<td>Google</td>
</tr>
<tr>
<td>105</td>
<td>290.202.20.90</td>
<td>11/9/2013</td>
<td>22:09:00</td>
<td>/lottery/</td>
<td>Bing</td>
</tr>
</tbody>
</table>