Please read all instructions (including these) carefully.

This is a closed-book exam. You are allowed one page of note sheets that you can write on both sides.

Write your name and UW student number below.

No electronic devices are allowed, including cell phones used merely as watches. Silence your cell phones and place them in your bag.

Solutions will be graded on correctness and clarity. Each problem has a relatively simple and straightforward solution. Partial solutions will be graded for partial credit.

There are 10 pages in this exam, not including this one.

There are 5 questions, each with multiple parts. If you get stuck on a question move on and come back to it later. Do not feel that you need to complete every question on the exam!

You have 50 minutes to work on the exam.

Please write your answers in the space provided on the exam, and clearly mark your solutions. You may use the blank pages as scratch paper. Do not use any additional scratch paper.

Relax. You are here to learn. Good luck!

By writing your name below, you certify that you have not received any unpermitted aid for this exam, and that you will not disclose the contents of the exam to anyone in the class who has not taken it.

NAME: _______________________

STUDENT NUMBER: _______________________

<table>
<thead>
<tr>
<th>Problem</th>
<th>Points</th>
<th>Problem</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>32</td>
<td>Total</td>
<td>100</td>
</tr>
</tbody>
</table>
Problem 1: Warm up (10 points total)

Select either True or False for each of the following questions. For each question you get 1 points for answering it correctly, -0.5 point for an incorrect answer, and 0 point for no answer. The minimum you will get for this entire problem is 0.

a) 1NF allows storing lists as an attribute.

b) Block-partitioning never results in data skew.

c) \( R \bowtie S \) is an instance of a theta-join.

d) All Datalog programs can be stratified.

e) In \( R(x,z) :- S(x,y), T(y,z), R(x,z) \) is an extensional predicate.

f) In relational query processing, physical query plans are translated into logical ones by the optimizer.

g) Subqueries can be used in the SELECT clause in SQL++, as long as the subquery returns a single value.

h) Using partitioning to scale up a database means cloning the entire dataset multiple times across different machines.

i) Relational data instances cannot be represented as trees.

j) In SQL, aggregates are always processed after grouping operations.
Problem 2: SQL (40 points total)

We will work with the following schema for sport teams in this exam.

Person(pid, name, age, salary, rating) -- pid = person ID
Team(tid, name) -- tid = team ID
MemberOf(pid, tid) -- pid is foreign key to Person, and tid is FK to Team
Games(winnerTid, loserTid) -- games won by winnerTid against loserTid,
    -- both are foreign keys to Team

You can assume none of the tables contains NULL values.

a) (10 points) Write a SQL query that returns the names of those who are currently members of multiple teams. Call the resulting column name.

b) (10 points) Write a SQL query that returns the team name(s) that has the highest total salary across all team members. Call the resulting column name.
c) (5 points) Given the following contents of Person:

<table>
<thead>
<tr>
<th>pid</th>
<th>name</th>
<th>age</th>
<th>salary</th>
<th>rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;A&quot;</td>
<td>30</td>
<td>50,000</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>&quot;B&quot;</td>
<td>24</td>
<td>10,000</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>&quot;C&quot;</td>
<td>21</td>
<td>15,000</td>
<td>2</td>
</tr>
</tbody>
</table>

What does the following query return? Write out the tuples in the resulting relation.

```sql
SELECT P1.rating as rating, AVG(P1.age) as avgAge
FROM Person P1
WHERE P1.age > 21
GROUP BY P1.rating
HAVING 1 < (SELECT COUNT(*)
            FROM Person P2
            WHERE P1.rating = P2.rating AND P2.age >= 21)
```
d) (10 points) Rewrite the Answer relation following relational algebra query into SQL without using GROUP BY.

\[
\begin{align*}
T1(\text{rating}, \text{sal}) &= \text{GroupBy}[\text{rating}, \text{avg(salary)->sal]}(\text{Person}) \\
\text{Answer}(r, \text{sal}) &= \text{Rename}[r, \text{sal}](T1)
\end{align*}
\]
e) (5 points) Are the following two queries equivalent? If so, write “Yes” below. Otherwise, write “No,” and make up the contents of Person such that, when run on the two queries, they will return different results. Write the contents in a table form, for instance, (not an answer). Make sure your data satisfies the key constraints!

<table>
<thead>
<tr>
<th>pid</th>
<th>name</th>
<th>age</th>
<th>salary</th>
<th>rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>“A”</td>
<td>30</td>
<td>50,000</td>
<td>3</td>
</tr>
</tbody>
</table>

SELECT P1.name
FROM Person P1
WHERE NOT EXISTS (SELECT *
                    FROM Person P2
                    WHERE P2.age < 21 AND P1.rating <= P2.rating)

SELECT P1.name
FROM Person P1
WHERE P1.rating > ANY (SELECT P2.rating
                         FROM Person P2
                         WHERE P2.age < 21)
Problem 3: Datalog (32 points total)

Same schema as before, repeated here for your reference.

Person(pid, name, age, salary, rating) -- pid = person ID
Team(tid, name) -- tid = team ID
MemberOf(pid, tid) -- pid is foreign key to Person, and tid is FK to Team
Games(winnerTid, loserTid) -- games won by winnerTid against loserTid,  
-- both are foreign keys to Team

You can assume none of the tables contains NULL values.

a) (10 points) Write a safe Datalog query that finds all teams that have never lost any games. Return the team’s ID and name in the relation Q3a(tid, name).

b) (10 points) Write a safe Datalog query that find the teams that each of those teams in a) have directly or indirectly beaten. Return the results in the relation Q3b(winnerTid, loserTid). For instance, if A is in Q3a and A beats B, and B beats C, then Q3b should contain (A, B) and (A, C). You can use the Q3a relation without recomputing it.
c) (4 points each, 12 points total) Are the following relational algebra or SQL queries equivalent? If so, write “Yes” below. Otherwise, write “No,” and make up the contents of Person and MemberOf such that, when run on the two queries, they will return different results.

Write the contents in a table form, for instance, Person:

<table>
<thead>
<tr>
<th>pid</th>
<th>name</th>
<th>age</th>
<th>salary</th>
<th>rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>“J”</td>
<td>30</td>
<td>50,000</td>
<td>3</td>
</tr>
</tbody>
</table>

(not an answer). Make sure your data satisfies the key constraints!

Q1: SELECT *
    FROM Person as P, MemberOf as M
    WHERE P.age < 21 AND M.tid > 40 AND P.pid = M.pid

Q2: \( \sigma_{age<21}(\sigma_{tid>40}(MemberOf) \bowtie_{pid=pid} Person) \)

Q1: \( \sigma_{salary<10k}(\sigma_{age<21}(Person) \bowtie_{pid=pid} (\sigma_{tid>40}(MemberOf))) \)

Q2: \( \sigma_{salary<10k}(Person \bowtie_{pid=pid} (\sigma_{tid>40}(MemberOf))) \)
Schema repeated here for your reference:
Person(pid, name, age, salary, rating) -- pid = person ID
Team(tid, name) -- tid = team ID
memberOf(pid, tid) -- pid is foreign key to Person, and tid is FK to Team
Games(winnerTid, loserTid) -- games won by winnerTid against loserTid, both are FKs to Team

Q1: $\gamma_{tid, \text{max(salary)}\rightarrow ms}(\text{Person} \bowtie_{\text{pid}=\text{pid}} \text{MemberOf}) \bowtie_{\text{tid}=\text{tid}} \text{Team}$

Q2: $\gamma_{tid, \text{max(salary)}\rightarrow ms}(\text{Person} \bowtie_{\text{pid}=\text{pid}} \text{MemberOf})$
Problem 4: SQL++ (15 points total)

Same schema as before, repeated here for your reference.

Person(pid, name, age, salary, rating) -- pid = person ID
Team(tid, name) -- tid = team ID
MemberOf(pid, tid) -- pid is foreign key to Person, and tid is FK to Team
Games(winnerTid, loserTid) -- games won by winnerTid against loserTid,
    -- both are FKs to Team

You can assume none of the tables contains NULL values.

a) (10 points) Assume we now have the information in four JSON documents with the following
schema and sample contents (quotation marks omitted for clarity):

Person = [ {pid:1, name:P1, age:30, salary:10k, rating:3.5},
    {pid:2, name:P2, age:24, salary:5k, rating:4.0}, ... ]
Team = [ {tid:1, name:T1}, {tid:2, name:T2}, ... ]
MemberOf = [ {pid:1, tid:1}, {pid:1, tid:2}, ... ]
Games = [ {winnerTid:1, loserTid:2}, {winnerTid:3, loserTid:4}, ... ]

Write a SQL++ query that combines data from MemberOf and Person as part of the new Team_new
document with the following schema and sample contents:

Team_new = [ {tid:1, name:T1,
        members:[ {pid:1, name:P1, age:30, salary:10k, rating:3.5},
            {pid:2, name:P2, age:24, salary:5k, rating:4.0}, ... }],
    {tid:2, name:T2, Members:[ ... ]}, ... ]
b) (5 points) Given the following contents of $Team_{new}$:

$Team_{new} = \{ \{tid: 1, name: T1, members: [\{pid: 1, name: P1\},\{pid: 2, name: P2\},\{pid: 3, name: P3\}\}],
\{tid: 2, name: T2, members: [\{pid: 3, name: P3\},\{pid: 2, name: P2\},\{pid: 4, name: P4\}\}],
\{tid: 3, name: T3, members: [\{pid: 3, name: P3\},\{pid: 2, name: P2\},\{pid: 4, name: P4\}\]\} \}$

What does the following query return? Write the answer as a well-formed JSON document.

```
SELECT DISTINCT m1.pid
FROM Team_{new} AS t1, t1.members AS m1, Team_{new} AS t2, t2.members AS m2
WHERE m1.pid = m2.pid AND t1.tid <> t2.tid
ORDER BY m1.pid
```

Problem 5: Trivia! (3 points)

Name 3 of the course staff below their pictures (first name is fine).

![Course staff pictures]