Question 1. [4 MARKS]

Suppose that

- $R1$ is a relation with $t1 \geq 1$ tuples.
- $R2$ is a relation with $t2 \geq 1$ tuples.
- $R3$ is a relation with $t3 \geq 1$ tuples.
- Both $R1$ and $R2$ have the same $a1 \geq 1$ attributes.
- $R3$ has $a3 \geq 1$ attributes.
- L is a list of attributes
- c is a boolean expression.

Assume that the expressions below are legal expressions of relational algebra. Fill in the table to indicate the number of tuples in the relation that is the result of each expression. Assume the set semantics.

Solution:

	Minimum number of tuples	Maximum number of tuples
$(R1 \cup \sigma_c R2) \bowtie R1$	$t1$	$t1$
$\Pi_L R3 \cup (\Pi_L R1 \cap \Pi_L R2)$	$t3$	$t3 + min(t1, t2)$

Question 2. [4 MARKS]

Suppose we have two relations: Patient(PID, height) and Caresfor(PID, doctor). Consider the following instance of that schema:

Patient

PID	height
12	11
25	12
96	11
20	12
33	33
44	8

Caresfor

PID	doctor
33	44
20	44
25	30
20	30
20	18
96	30
30	20

Part (a) [2 MARKS]

Give the result (schema and data) returned by the following query. Use the same tabular format as above.

$S := Patient \bowtie Caresfor$

$T(PID, height, doctor) := \Pi_{S2.PID, S2.height, S2.doctor}(\sigma_{S1.doctor=S2.doctor \wedge S1.height>S2.height}(\rho_{S1}(S) \times \rho_{S2}(S)))$

$Answer := \Pi_{height, doctor}(S - T)$

Solution:

height	doctor
33	44
12	30
12	18

Part (b) [2 MARKS]

Describe what this query computes. Do not describe the steps it takes, only what is in the result, and make your answer general to any instance of the schema.

Solution:

Every doctor (who has a patient in the Patients table) and the height of their tallest patient(s).

Question 3. [15 MARKS]

We used the following schema many times in lecture:

Relations

- Students(SID, surName, campus)
- Courses(CID, cName, WR)
- Offerings(OID, CID, term, instructor)
- Took(SID, OID, grade)

Integrity constraints

- Offerings[CID] $\subseteq$ Courses[CID]
- Took[SID] $\subseteq$ Students[SID]
- Took[OID] $\subseteq$ Offerings[OID]

Part (a) [6 MARKS]

Consider all the writing requirement courses that Atwood has taught at one point or another. Write a query in relational algebra to report the instructors who have taught (it doesn't matter what offering) every one of these. Use only the basic operators $\Pi, \sigma, \bowtie, \times, \cap, \cup, -, \rho$.

Solution:

$AWRCourses(CID) := \Pi_{CID}\sigma_{instructor=\text{“}Atwood\text{”} \wedge WR=True} Offering$
$ShouldHaveTaught(instructor, CID) := (\Pi_{instructor} Offerings) \times AWRCourses$
$DidTeach(instructor, CID) := \Pi_{instructor,CID} Offerings$
$MissedSome(instructor, CID) := ShouldHaveTaught - DidTeach$
$\Pi_{instructor} Offerings - \Pi_{instructor} MissedSome$

Part (b) [1 MARK]

Make the smallest possible instance of Took that violates its key constraint but is otherwise valid.

Solution:

SID	OID	grade
1	2	3
1	2	4

Any two tuples with the same SID and OID will violate the constraint.

Part (c) [8 MARKS]

Suppose we want to find all the instructors who have never taught a course that Clarke hasn't taught at some point. Which of the following syntactically legal queries will report that?
(2 marks for each correct answer, -1 for each incorrect answer.)

1. $Temp := (\Pi_{instructor,course} \mathit{Offering}) - (\Pi_{instructor,course} \sigma_{instructor=\text{“Clarke”}} \mathit{Offering})$
   $\Pi_{instructor} \mathit{Offering} - \Pi_{instructor} Temp$

   Correct [✓ Incorrect]

   There was a typo here (“course” should be “CID”), but even with that fixed the query is wrong.

2. $Temp := \Pi_{CID}[\mathit{Offering} - (\sigma_{instructor=\text{“Clarke”}} \mathit{Offering})]$
   $(\Pi_{instructor} \mathit{Offering}) - ((\Pi_{instructor} \mathit{Offering}) \bowtie Temp)$

   Correct [✓ Incorrect]

   This has a syntax error (the set difference has operands with different attributes). Even if you fix that by projecting the RHS operand onto instructor, the query is wrong.

3. $Temp := (\Pi_{CID} \mathit{Offering}) - (\Pi_{CID} \sigma_{instructor=\text{“Clarke”}} \mathit{Offering})$
   $(\Pi_{instructor} \mathit{Offering}) - \Pi_{instructor}(\mathit{Offering} \bowtie Temp)$

   [✓ Correct] Incorrect

4. $Temp := \Pi_{CID} \sigma_{instructor \neq \text{“Clarke”}} \mathit{Offering}$
   $(\Pi_{instructor} \mathit{Offering}) - \Pi_{instructor}(\mathit{Offering} \bowtie Temp)$

   Correct [✓ Incorrect]

Question 4. [7 marks]

Consider the following schema about athletes and their results in the long jump event:

```
-- Long jump results.  "who" is the athlete whose
-- result it is, "distance" is how far they jumped, and
-- "t" is when they did it.
create table longjump (
        who int,
        distance float not null,
        t timestamp,
        primary key (who, t),
        foreign key (who) references athlete(aID)
);
```

```
create table athlete (
        aID int primary key,
        name text not null
);
```

Part (a) [3 marks]

Complete the `where` condition in the following SQL query so that it reports every athlete who jumped a distance that was not the shortest distance anyone ever jumped. If there are duplicates, report them all.

```
SELECT aid, name
FROM longjump NATURAL JOIN athlete
WHERE
```

Solution:

```
aid = who and distance > some (select distance from longjump);
```

Part (b) [1 mark]

Assume that your query correctly answers the question above. Is it possible that every athlete could appear in the result?
(One mark for a correct answer and -0.5 for an incorrect answer.)

[✓ Yes] No

Part (c) [1 mark]

Can a tuple in table `longjump` have a value for `who` that is `null`?
(One mark for a correct answer and -0.5 for an incorrect answer.)

Yes [✓ No]

Part (d) [2 marks]

Suppose we have a relation Huh(a, b, c). Fill in the blanks below to make the queries legal:

```
SELECT MAX(a), c, MIN(b) from Huh  ___________GROUP BY c____________;


SELECT  ________a_________  ,  ________avg(b)_______  FROM Huh GROUP BY a HAVING a < 50;
```

(Any column can be aggregated in the second SELECT, and column `a` can appear without aggregation.)