Question 1. [11 MARKS]

Recall this schema, which we have used many times in class. Here we are adding one more relation called *Program*. It records the subject POSts that students are enrolled in. ("POSt" is short for "program of study", by the way.)

Relations

Student(sID, surName, firstName, campus, email, cgpa)

Course(dept, cNum, name, breadth)

Offering(oID, dept, cNum, term, instructor)

Took(sID, oID, grade)

Program(sID, POSt)

Integrity constraints

Offering[dept, cNum] ⊆ Course[dept, cNum]

Took[sID] ⊆ Student[sID]

Took[oID] ⊆ Offering[oID]

Program[sID] ⊆ Student[sID]

Part (a) [7 MARKS]

Write a query to find the sIDs of students on campus 'StG' who have exactly one subject POSt. Use only the basic operators $\Pi, \sigma, \bowtie, \times, \cap, \cup, -, \rho$, and assignment.

Solution:

$StG(sID) := (\Pi_{sID}\sigma_{campus='StG'}Student)$

$OnePOSt(sID) := \Pi_{P1.SID}\sigma_{P1.SID=P2.SID \wedge P1.POSt \neq P2.POSt}(\rho_{P1}Program \times \rho_{P2}Program)$

$Answer(sID) := StG \cap OnePOSt$

Part (b) [4 MARKS]

Consider the following query:

$$Croom(instructor, cNum) := (\Pi_{instructor} Offering) \times (\Pi_{cNum}\sigma_{dept='CSC' \wedge cNum \geq 200 \wedge cNum < 300} Course)$$

$$Flep(instructor, cNum) := Croom - (\Pi_{instructor,cNum} Offering)$$

$$Answer(instructor) := [(\Pi_{instructor} Offering) - (\Pi_{instructor} Flep)] - (\Pi_{instructor}\sigma_{dept='MAT'} Offering)$$

1. Below are instances of the relations that are relevant to this query. Add exactly 5 rows to *Offering* so that professors Able and Bland will not appear in the result of the query, but professors Cranky and Devlish will. Use only these instructor names, and be sure the instance you create is valid.

Course:

dept	cNum	name	breadth
CSC	108	Intro Prog	false
CSC	207	Intro Design	true
CSC	209	Sys Prog	false
MAT	137	Calculus	false
CSC	369	Op Sys	false
ENG	244	Shakespeare	true

Offering:

oID	dept	cNum	term	instructor
4	CSC	108	t1	Cranky
1	CSC	108	t2	Able
7	CSC	209	t1	Devlish
2	CSC	209	t1	Bland
9	CSC	207	t3	Bland

Solution:

- Professor Able is already not in the result, so there is nothing to do.
- Professor Bland is currently in the result, but we can get rid of him by having him teach some math course.
- Professor Cranky is not in the result, and we need to have him teach csc207 and csc209 (all the 200-level CSC courses), and no math courses, in order to get him in the result.
- Professor Devlish needs CSC207, and no math courses, in order to be in the result.
- To add a fifth new tuple and not mess up the results, we can add some course that is not a math course or a 2nd-year CSC course.

Here is a complete solution:

Offering:

oID	dept	cNum	term	instructor
4	CSC	108	t1	Cranky
1	CSC	108	t2	Able
7	CSC	209	t1	Devlish
2	CSC	209	t1	Bland
9	CSC	207	t3	Bland
10	MAT	137	t4	Bland
11	CSC	207	t4	Cranky
12	CSC	209	t4	Cranky
13	CSC	108	t4	Able

2. What does this query compute? 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:
All instructors who have taugh every 2nd-year CSC course, but no MAT course.

Question 2. [6 MARKS]

Part (a) [2 MARKS]

At UofT, a student may have no POSt, one POSt, or several POSts. In the previous question, we introduced a new relation called *Program* to record information about students' POSts. Instead of making a separate *Program* relation, we could add a column for *POSt* in the *Student* relation. Would that be a good design? Circle one answer:

Solution:

YES [NO]

Explain:

With this design, we wouldn't be able to record two POSts for one student. We also could not handle a student who has no POSt, since we don't have null values in relational algebra.

Part (b) [4 MARKS]

Consider this schema:

R(<u>one</u>, two, three) S[five, six] ⊆ R[one, two]

S(<u>four</u>, five, six) S[four] ⊆ T[eight]

T(<u>seven</u>, eight)

Suppose relation S has 100 tuples. How many tuples could R have? Circle all answers that do not violate the schema.

Solution:

0 [1] [82] [100] [101]

Suppose relation S has 100 tuples. How many tuples could T have? Circle all answers that do not violate the schema.

Solution:

0 1 82 [100] [101]

Question 3. [5 MARKS]

The question refers to the schema from Question 1. Write a query in SQL to find the total number of courses in the 'CSC' department each student has taked. Report the student id and the total number of *distinct* 'CSC' courses taken.

]

Solution:

```
SELECT sid, count(distinct cNum)
FROM Took, Offering
WHERE dept = 'CSC' AND Took.oID = Offering.oID
GROUP BY sid
```

Question 4. [8 MARKS]

Part (a) [3 MARKS]

Consider the same schema from the Question 1. Suppose we wrote the query

```
SELECT ________________________
FROM Offering, Took
WHERE Offering.oID = Took.oID
GROUP BY Offering.oID;
```

Which of the following could go in the `SELECT` clause? Circle all that apply.

`sID` `count(sID)` `Offering.oID` `grade` `avg(grade)` `count(instructor)` `oID`

Solution:

`sID` `count(sID)` `Offering.oID` `grade` `avg(grade)` `count(instructor)` `oID`

Here are the error messages for the problematic ones:

```
csc343h-dianeh=> select sID
csc343h-dianeh-> from offering, took
csc343h-dianeh-> where Offering.oID = Took.oID
csc343h-dianeh-> GROUP BY Offering.oID;
ERROR:  column "took.sid" must appear in the GROUP BY clause or be used in an aggregate function
LINE 1: select sID
               ^
csc343h-dianeh=> select grade
csc343h-dianeh-> from offering, took
csc343h-dianeh-> where Offering.oID = Took.oID
csc343h-dianeh-> GROUP BY Offering.oID;
ERROR:  column "took.grade" must appear in the GROUP BY clause or be used in an aggregate functio
LINE 1: select grade
               ^
csc343h-dianeh=> select oid
csc343h-dianeh-> from offering, took
csc343h-dianeh-> where Offering.oID = Took.oID
csc343h-dianeh-> GROUP BY Offering.oID;
ERROR:  column reference "oid" is ambiguous
LINE 1: select oid
```

Part (b) [3 MARKS]

We discussed in lecture how SQL subquery operators could possibly be implemented using other SQL operations. Suppose we have two tables `R(a,b)` and `S(b,c)`.

Consider the following two queries:

```
-- Query 1
SELECT a AS answer
FROM R
WHERE EXISTS
(SELECT * FROM S
 WHERE c > a AND R.b = S.b);

-- Query 2
SELECT R.a AS answer
FROM R, S
WHERE c > a AND R.b = S.b;
```

On the next page, give a **database instance** where these two queries produce *different* results, and the **results of the two queries**.

Solution:

```
insert into R values
(1, 2),
(1, 2),
(3, 4),
(5, 6);

insert into S values
(2, 101),
(4, 103),
(4, 203);

-- Query 1 gives:
 answer
--------
      1
      1
      3
(3 rows)

-- Query 2 gives:
 answer
--------
      1
      1
      3
      3
(4 rows)
```

Part (c) [2 MARKS]

Here is an attempt to fix Query 2, to make it identical to Query 1. Explain, in English, why this new query is *not* the same as Query 1. Don't just give a database instance – write a sentence or two describing the problem precisely.

```
-- Query 3
SELECT DISTINCT R.a AS answer
FROM R, S
WHERE c > a AND R.b = S.b;
```

Solution:

If R has duplicate `a` values they won't appear in this query, even though they both would in Query 1, if the conditions are satisfied for both tuples.

```
-- Query 3 gives:
 answer
--------
      1
      3
(2 rows)
```