PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

Midterm Test

CSC 343H1F, 2016
Section L0101/L2001 (Horton)

Duration — 50 minutes
No aids allowed

Student Number: |__|__|__|__|__|__|__|__|__|__|

Last Name: ____________________________________

First Name: ____________________________________

Section normally attended (Circle one): L0101/L2001(1pm) L0201/L2003 (3pm) L5101/L2501(6pm)

Do **not** turn this page until you have received the signal to start.
In the meantime, please fill out the identification section above,
and read the instructions below.

This test consists of 4 questions on 10 pages (including this one). When you receive the signal to start, please make sure that your copy of the test is complete.

If you use any space for rough work or need to scratch out an answer, circle the part that you want us to mark.

You may write in pencil, however, work written in pencil will not be considered for remarking.

Good Luck!

Question 1. [12 marks]

Consider this schema for Twitter, a social media platform where users post messages called “tweets”.

Relations

User(userID, name, email)
A Twitter user.

Tweet(tweetID, userID, content, day)
The user with *userID* made a tweet containing *content* on *day*.

Follows(a, b)
User *a* follows user *b* on Twitter, which means that *a* has subscribed to *b*’s tweets.

Likes(who, what, d)
User *who* liked tweet *what* on day *d*.

Integrity Constraints

Tweet[userID] $\subseteq$ User[userID]

Follows[a] $\subseteq$ User[userID]

Follows[b] $\subseteq$ User[userID]

Likes[who] $\subseteq$ User[userID]

Likes[what] $\subseteq$ Tweet[tweetID]

Part (a) [2 marks]

Does the schema enforce this constraint: A user cannot like the same tweet twice? Circle one:

[✓ Yes] No.

If yes, explain; if not write a new constraint to enforce it:

since Likes have key who,what
same user-tweet combination never duplicates

Part (b) [2 marks]

Does the schema enforce this constraint: You can’t follow yourself? Circle one:

Yes [✓ No.]

If yes, explain; if not write a new constraint to enforce it:

select a=b on Follows is empty set

Part (c) [1 mark]

Suppose relation *Likes* has 300 tuples. How many tuples could *Users* have? Circle all that apply:

0 [✓ 1] [✓ 256] [✓ 300] [✓ 912]

Part (d) [2 marks]

Suppose relation *User* has m tuples and relation *Tweet* has n tuples. What is the maximum number of tuples that relation *Likes* can have?

m x n

Explain how the schema imposes this limit:

by key of Likes
there are m x n possible combinations

Part (e) [3 MARKS]

Suppose we add the following constraint: Likes[who] ⊆ Follows[b]. Make the smallest possible non-empty instance of relations *Likes* and *Follows* that violates this constraint:

Express this constraint in English:

liker must have follower
so cant like tweets if there is follower

What kind of constraint is it? Circle all that apply:

referential integrity constraint   foreign-key constraint   integrity constraint

Part (f) [2 MARKS]

Which of the following queries can be expressed using the same form of relational algebra that we used in class and on Assignment 1, that is: the operators $\Pi, \sigma, \bowtie, \bowtie_{condition}, \times, \cap, \cup, -, \rho$ and assignment? Circle all that apply.

1. Find everyone who follows 6 or more people who have never liked a tweet.
2. Let's say user X is "upstream" of Y if either X follows Y, or X follows someone else who is upstream of Y. Find every user who is upstream of the person with userID 'Oprah'.
3. Find the second last tweet from the person with userID 'Oprah'.
4. Find the user who follows the most people.
5. Find the user who made the first tweet.

Question 2. [8 MARKS]

Here is the schema from Assignment 1. A few attributes and relations have been omitted for simplicity.

Relations

Product(<u>DIN</u>, manufacturer, name, form, schedule)
A tuple in this relation represents a drug product.

Price(<u>DIN</u>, price)
The price of a drug product.

Prescription(<u>RxID</u>, date, patient, drug, doctor)
A prescription for *drug* was written on *date* for *patient* by *doctor*. Attribute *patient* is the patient's OHIP number.

Filled(<u>RxID</u>, date, pharmacist)
Prescription *RxID* was filled by *pharmacist* on *date*. Attribute *pharmacist* is the pharmacist's OCP number.

Integrity constraints

$\text{Price}[\text{DIN}] \subseteq \text{Product}[\text{DIN}]$

$\text{Prescription}[\text{drug}] \subseteq \text{Product}[\text{DIN}]$

$\text{Filled}[\text{RxID}] \subseteq \text{Prescription}[\text{RxID}]$

$\Pi_{\text{schedule}}\text{Product} \subseteq \{\text{“prescription”}, \text{“narcotic”}, \text{“OTC”}\}$

Write a query in relational algebra to report the OHIP number of every patient who has had a prescription that (a) was for the most expensive drug product (or a product tied for most expensive) and (b) they never filled.

Use only the basic operators $\Pi, \sigma, \bowtie, \times, \cap, \cup, -, \rho$, and assignment.

Continue your answer here if more space is needed.

Question 3. [6 marks]

Suppose we have implemented the Twitter schema from Question 1 in SQL, and the tables currently contain the following:

```
Profile:
 userid |     name     |    email
--------+--------------+--------------
 adele  | Adele Adkins |
 drizzy | Drake        |
 potus  | Barack Obama | potus@gov.us
 rjm    | Renee Miller | rjm@cs
```

```
Follows:
   a    |   b
--------+--------
 potus  | drizzy
 drizzy | rjm
```

```
Tweet:
 tweetid | userid | content  |    day
---------+--------+----------+------------
      15 | adele  | Hello    | 2016-10-16
      61 | adele  | It's me  | 2016-10-16
      33 | potus  | 6 weeks  | 2016-10-11
      28 | rjm    | in the 6 | 2016-10-10
```

```
Likes:
  who   | what |     d
--------+------+------------
 drizzy |   61 | 2016-10-18
 rjm    |   33 | 2016-10-17
 drizzy |   15 | 2016-10-16
 potus  |   15 | 2016-10-16
```

Show the output of each of the queries below. If a query will not run successfully, write “Illegal”.

```
SELECT who
FROM Likes JOIN Tweet ON what = tweetID
WHERE userID = 'adele';
```

```
SELECT userID, count(tweetID), count(day)
FROM Tweet
GROUP BY userID;
```

```
SELECT count(*) AS num1, count(email) AS num2
FROM Profile;
```

```
SELECT name, content
FROM Profile NATURAL RIGHT JOIN Tweet;
```

```
SELECT tweetID, count(who)
FROM Tweet, Likes
WHERE tweetID = what;
```

```
(SELECT a AS userID
FROM Follows
WHERE b = 'drizzy')
   UNION ALL
(SELECT userID
FROM Tweet
WHERE content like '%6%');
```

Question 4. [4 marks]

Write a query to find the userID of everyone who has made more than one Tweet. Ensure that it would work on any instance of the database, not simply the one above.

# 1: ______/12

2: ______/ 8

3: ______/ 6

4: ______/ 4

TOTAL: ______/30