# CSI 2532 - Homework 1 [120 points]

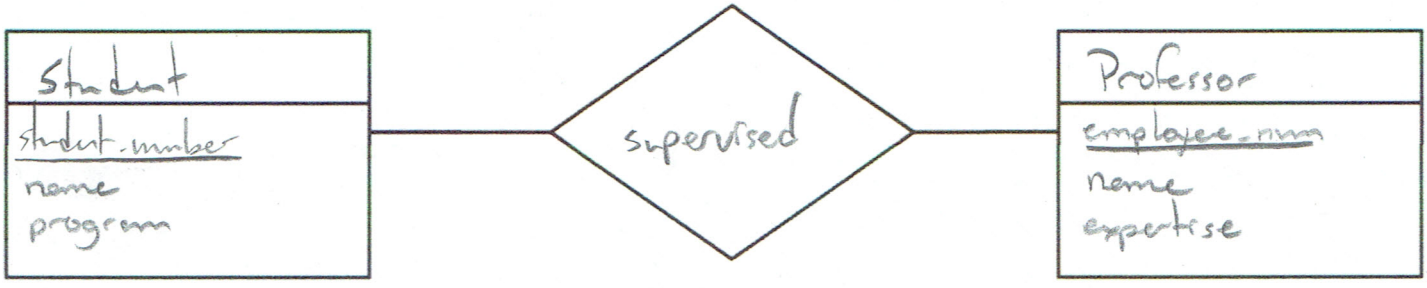
All work should be submitted via GitHub Classroom using the invite URL https://classroom.github.com/a/Nw040UjC.

Use GitHub Markdown and provide your solution within a `README.md` file. Use backticks (```) for code blocks, and use `![title](reference)` to embed images.

## Part A [60 points]: E-R Models

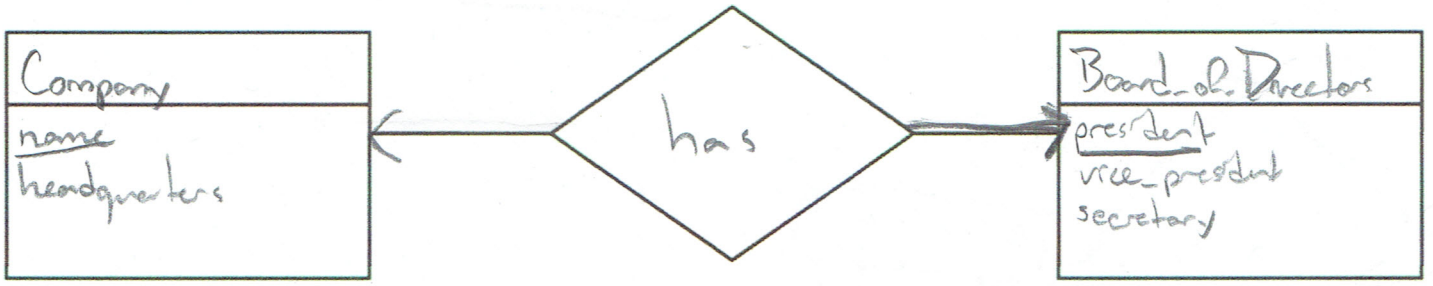
### A1 [15 points]: Relations, Cardinality and Participation

a) **[5 points]** A student can be supervised by any number of professors, and a professor can supervise any number of students. A student has a student number, name and enrolled in a specific program. A professor has an employee number, name and area of expertise.




b) **[5 points]** All course sections must be assigned to a course. A course has a department code (e.g. CSI or SEG) and a course number (e.g. 2532 or 4105). A course section is uniquely defined by section name (e.g. A or B or C), semester (e.g. Winter), a year and the course itself. A course section depends on its course.

c) **[5 points]** A company has a name and headquarters (country). A board of directors has a president, a vice-president, and a secretary. A company can have at most one board of directors (but it does not require one). All board of directions must have one and only one associated company to manage.




## A2 [30 points]: System Design

You have been asked to implement a database (using a Relational Diagram) for CorporateBnB, a company that allows anyone with a home office to provide meeting spaces to others. Any user can sign up for this service by providing their name, email and phone number. A user can manage office space, or a user can rent office space. An office property has a name, an address, and square footage. Each property has availability for when it can be rented. Availability tracks the date, and the daily rate for use, the check-in and check-out times and if that date is still available. When a user rents a property the database will track the arrival date and departure date.

Model the system above using an Entity-Relation diagram. You must include entities, relations, cardinality, participation and attributes. You do not need to specify attribute types (i.e. the domain).

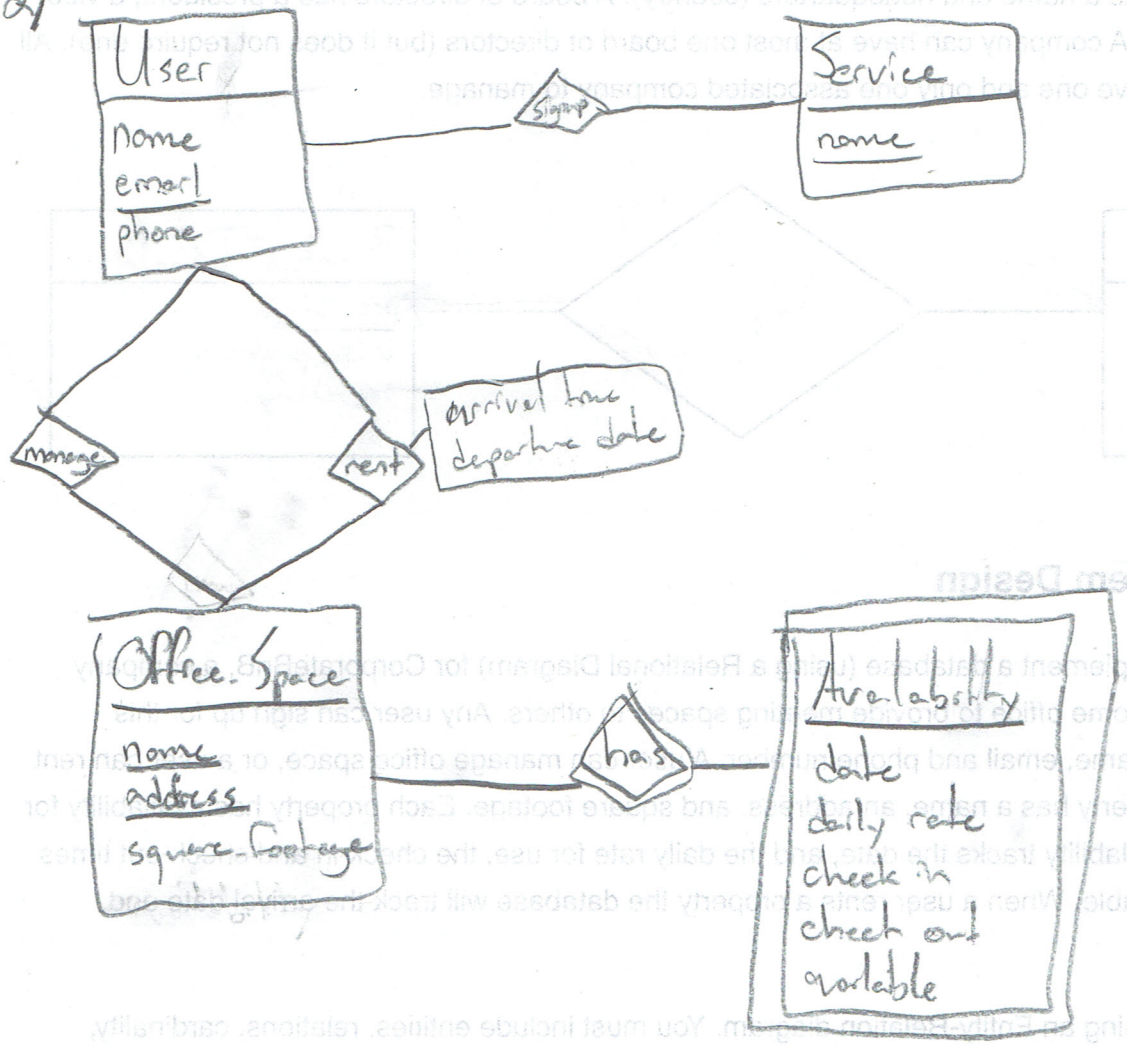
## A3 [15 points]: Relational Algebra

Write relational queries for the following situations.

a) **[7 points]** Find all office spaces in Ottawa that are available March 2, 2020.

b) **[8 points]** Find all users (name and email) and the property details (name, and city) and the rental information (date, and daily rate) of all rentals during the month of January 2020.

A2)


User
name
email
phone
Signup
Service
name
manage
rent
arrival time
departure date
Office_Space
name
address
square_footage
has
Availability
date
daily rate
check in
check out
available

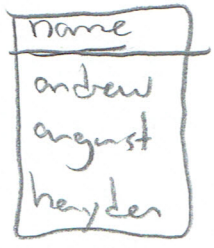
```
INSERT INTO licenses (user_id, software_name, access_code)
VALUES
  (48, 'MS Word', 'abc123'),
  (49, 'MS Word', 'def456'),
  (50, 'MS Word', 'hij789'),
  (48, 'Sketch', 'x1y2z3'),
  (51, 'Sketch', 'x2y3z4');
```

## B1. [15 points] Reading SQL Queries

For each of the following, show the output of the query OR (briefly) explain why the query does run and re-write it so that it can be run.

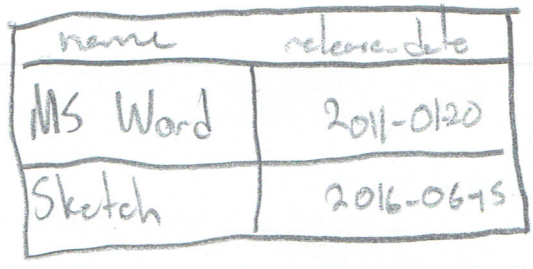
a) **[4 points]**

```
SELECT
  name,
  EXTRACT(year from age(users.join_date)) AS experience
FROM users
JOIN licenses ON licenses.user_id = users.id
WHERE licenses.software_name = 'MS Word'
ORDER BY users.name;
```



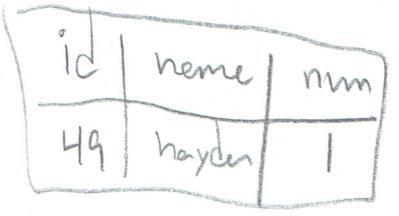

b) **[4 points]**

```
SELECT name,
       released_date
FROM softwares
WHERE released_date < '2018-01-01'
ORDER BY released_date ASC;
```



c) **[7 points]**

```
WITH users_2019 (id, name) AS
  (SELECT *
   FROM users
   WHERE join_date BETWEEN '2019-01-01' AND '2019-12-31')
SELECT id,
       name,
       count(licenses.access_code) AS num
FROM users_2019
LEFT JOIN licenses ON licenses.user_id = id
GROUP BY name
ORDER BY num DESC;
```




## B2. [15 points] Writing SQL Queries

a) **[3 points]** Find the names of all users that joined before Jan 1, 2020.

Select name From users Where join_date = "2020-01-01"

b) **[5 points]** Show the number of software licenses of each user (even those without any licenses). Sort the results by the number of licenses (most to least) and then by name (A - Z).

c) **[5 points]** Insert additional data into the database to demonstrate the proper sorting of (b).

d) **[2 points]** Update the sketch version to be version "51" (released Jan 1, 2020).

Update version Where name = "Sketch"

## B3. [30 marks] Updating SQL Schemas

After you updated Sketch to version 51, you noticed a design flaw in your database. Not all users upgrade to the latest version of the software just because it is available.

If you try to execute the following it will fail.

```
INSERT INTO softwares (name, version, released_date)
VALUES
  ('MS Word', '2020', '2019-06-15'),
  ('Chrome', 'v92', '2020-01-01'),
  ('Sketch', '52', '2020-02-11');
```

Using SQL write SQL migrations to improve the design of the database system.

Each migration should be atomic. Hint: if you require multiple SQL queries then wrap the calls in a transaction (BEGIN; ... COMMIT;)

Use `\d+ <table name>` (e.g. `\d+ licenses;` ) to see details about a Postgres table.

a) **[5 points]** Add the software version to the licenses table.

Insert Into licenses (user_id, software_name, access_code) Values (51, Sketch, x2y3z4)

b) **[5 points]** Update the softwares table to include the name AND version as the primary key.

Alter Table softwares add constraint name Primary key, version Primary key

c) **[10 points]** Update the licenses table to allow users to have multiple versions of the same software. To demonstrate this works, add Sketch 52 to "andrew" user with access code "xxxyyy111". Do not hard code the user id, it should work for any database with an "andrew" user.

Insert Into licenses (user_id, software_name, access_code) Values (48, Sketch, xxxyyy111)

d) **[10 points]** Sketch is providing a promotion (access code "1monthfree") for version 52. Give that license to anyone that does not yet it, allowing them to keep any older version they might have. Do not hard code the user list, it should work for any database.