1. (3 pts.) Let us design a database for a university. This database will include information about departments, professors, and courses (and their offerings):

- Information about **professors** includes their SSN and name. The SSN of a professor is assumed to be unique, not shared by any other professor. Each professor is either a **junior professor** or a **tenured professor**. (Each professor must be in one category or the other, and cannot be in both categories simultaneously.) For junior professors, we record the number of years since they were hired. For tenured professors, we record the year in which they earned tenure.

- Information about **departments** includes their name. The name of a department is assumed to be unique, not shared by any other department.

- We need to be able to associate professors with the departments with which they are affiliated. Each professor has to be affiliated with at least one department.

- Each department has exactly one chairperson, which has to be a **tenured** professor. You do **not** need to model the fact that the chairperson of a department has to be affiliated with the department.

- Information about a **course** includes its number (e.g., “4111”) and name (e.g., “Introduction to Databases”). We also need to be able to know the unique department that owns each course: no crosslisting of courses across departments is allowed and every course is owned by exactly one department. **Note/hint:** You cannot assume that course number uniquely identifies a course; in fact, you cannot assume even that course number together with course name uniquely identify a course. However, course number uniquely identifies courses **within a department.**

- Finally, we need to record all **terms**—identified as semester (e.g., “spring”) and year (e.g., “2017”—in which each course has been offered in the history of the university, and what professor(s) taught each course offering. Assume that a course offering might be associated with zero or more professors: a course might be taught in some term by a grad student, which we will **not** model in our design; it's also OK for a course offering to be cotaught by multiple professors. Assume that a course is offered at most once during each term. In other words, a course cannot have multiple sections during one term. Finally, assume that a professor can teach courses “owned” by departments with which the professor is not affiliated.

Here is what we ask you to do. Render the university database in the version of the E/R model that we studied in class, with exactly the constraints and requirements specified above. Please state any assumptions that you make, but make sure that you don’t introduce new constraints that are not listed in the problem definition. **Note/hint:** Note that a weak entity set is a “regular” entity set in that it can participate in relationship sets other than the identifying relationship set, just as any other entity set.
2. (4 pts.) Translate the following ER diagram into the relational model by writing the SQL table creation commands necessary to define the relations. Be sure to include primary keys and foreign keys where applicable.
3. (3 pts.) In this question we ask you to apply constraint checking to a database. The database manages customer orders, a common commercial application of database systems, and consists of three relations, defined as follows:

CREATE TABLE Customers(id INTEGER, name CHAR(40), email CHAR(64), PRIMARY KEY (id), UNIQUE (email))

CREATE TABLE Orders(id INTEGER, custID INTEGER, saledate DATE, PRIMARY KEY (id), FOREIGN KEY (custID) REFERENCES Customers ON DELETE NO ACTION ON UPDATE CASCADE)

CREATE TABLE LineItems(id INTEGER, orderID INTEGER, item CHAR(64), quantity INTEGER, cost REAL, PRIMARY KEY (id, orderID), FOREIGN KEY (orderID) REFERENCES Orders ON DELETE CASCADE ON UPDATE CASCADE)

Customers

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Thomas Jefferson</td>
<td><a href="mailto:tj@monticello.com">tj@monticello.com</a></td>
</tr>
<tr>
<td>2</td>
<td>George Washington</td>
<td><a href="mailto:gw@vernon.net">gw@vernon.net</a></td>
</tr>
<tr>
<td>3</td>
<td>John Kennedy</td>
<td><a href="mailto:jfk@hyannis.org">jfk@hyannis.org</a></td>
</tr>
<tr>
<td>4</td>
<td>George H. W. Bush</td>
<td><a href="mailto:bush41@kennebunkport.us">bush41@kennebunkport.us</a></td>
</tr>
</tbody>
</table>

Orders

<table>
<thead>
<tr>
<th>id</th>
<th>custID</th>
<th>saledate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>03/05/1801</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>07/04/1961</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>04/06/2005</td>
</tr>
</tbody>
</table>

LineItems

<table>
<thead>
<tr>
<th>id</th>
<th>orderID</th>
<th>item</th>
<th>quantity</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Quills</td>
<td>12</td>
<td>3.50</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Parchment</td>
<td>5</td>
<td>3.88</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>Pens</td>
<td>7</td>
<td>23.00</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>Broccoli</td>
<td>3</td>
<td>4.20</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Sushi</td>
<td>4</td>
<td>3.90</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Cuban Cigars</td>
<td>12</td>
<td>65.00</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Tapered Candles</td>
<td>12</td>
<td>6.99</td>
</tr>
</tbody>
</table>
What you need to do: Apply each of the following commands to the database in the order they are listed. If a command fails because of a constraint violation, identify which constraint is violated and explain how. Show the state of the three relations after each legal command (i.e., each command that does not violate any constraints) has been applied.

**INSERT**
INTO Customers (id, name, email)
VALUES (6, 'Bill Clinton', 'wjc@whitewater.net')

**UPDATE**
Customers C
SET C.id = 6
WHERE C.name = 'George Washington'

**UPDATE**
Customers C
SET C.email = 'gw@vernon.net'
WHERE C.id = 6

**DELETE**
FROM Orders O
WHERE O.id = 3

**DELETE**
FROM Customers C
WHERE C.id = 3

**UPDATE**
Orders O
SET O.id = 10
WHERE O.id = 1

**INSERT**
INTO LineItems (id, orderID, item, quantity, cost)
VALUES (1, 5, 'Socks', 5, 7.99)

**DELETE**
FROM Orders O
WHERE O.saledate > 01/01/1900

**DELETE**
FROM Customers C
WHERE C.id = 3