**OBJECTIVE:** Get some experience with a slightly larger modeling assignment than we have been doing in the weekly homework assignments.

**INTRODUCTION:** This will be the first phase in a three-part term project that will give you some experience using a programming language (Python) to interface between the user and the database that we are using to serialize the data that the user is giving us.

I needed to get a key made so that I could get into ECS 308 to let in my students. Along the way, I learned a couple of interesting things about the lock and key system at CSULB:

* Keys are issued by Building and Maintenance to employees (staff or faculty) who make a request. Each request specifies which room they want access to, and the identity of the employee. University employees have an ID number, just like students.
  + An employee can make more than one request to access a room. They might get a key to that room, then return the key when they no longer need it, and then get assigned to teach in that room again later and they need to request access again.
  + When the request is made, we track who made the request and when it was made.
* A room is uniquely identified by room **number within** a building. Assume the room number is an integer. Do not worry about rooms that can be subdivided. That gets a bit more complex than we want to get into now.
* Each room has at least one door. Assume that the door has a name, like “front”, “back”, “south”, and so forth.
  + Assume that there are only a few possible names for a door.
  + No two doors to a room will have the same name.
* Every key issued is a copy of a master key. For some reason, these master keys are called “hooks”. Don’t ask me why, I just work here.
* Each hook will open several doors, and each door could be opened by several hooks. For instance, the back door to the CECS department office will open for two of the keys that I have.
* Not all the doors of a room will open for the same hook. For instance, the department office **front** door will not open for a key that I have, but the back door **will**.
* Each key is issued to an employee at a particular date and time. Eventually, that employee will return that key. At that point, that same key could be issued to some other employee. We keep track of when each key is loaned out, and when it was returned (once it has been returned).
* **Sometimes** an employee loses their key. When such a loss is reported, we record that and charge the employee $25 for the key.
* We want to make sure that we never issue a key unless it has been requested.
* We want to make sure that we never issue two keys to the same employee that are the same hook. That would be silly since those two keys on the same hook would be identical.
  + The tricky part here is that an employee could get access to one room, return the key for that room, then request access for another room that happens to be the same hook.
    - In that case, the request would be granted since they no longer have the first key.
  + On the other hand, if the employee still has the first key, the second request needs to be denied.

**PROCEDURE:** For this part of the project, you need to:

1. UML model all the business rules that you can from the above statements.
2. For each of the business rules that you cannot implement by the way that you structure the database, write out in English pseudo code how you will implement that business rule in your Python code.
3. Using the above UML model as your starting place, build an ERD diagram. **Note**: you are not actually going to implement the tables for this ERD diagram. Instead, SQL Alchemy will build your tables **for you**.

**WHAT TO TURN IN:**

* The UML diagram of the above business rules.
* The English pseudo code for those business rules that you will have to implement in Python.
* The ERD model of your UML.