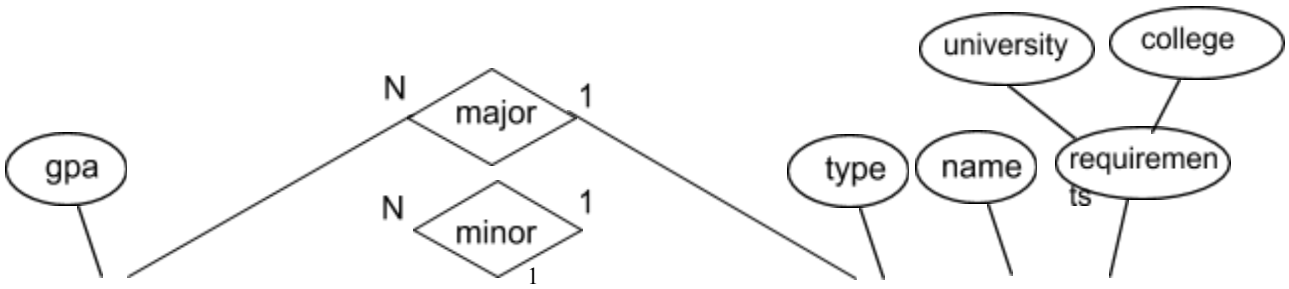
COSC 30603
Lab Assignment 3: EER Model

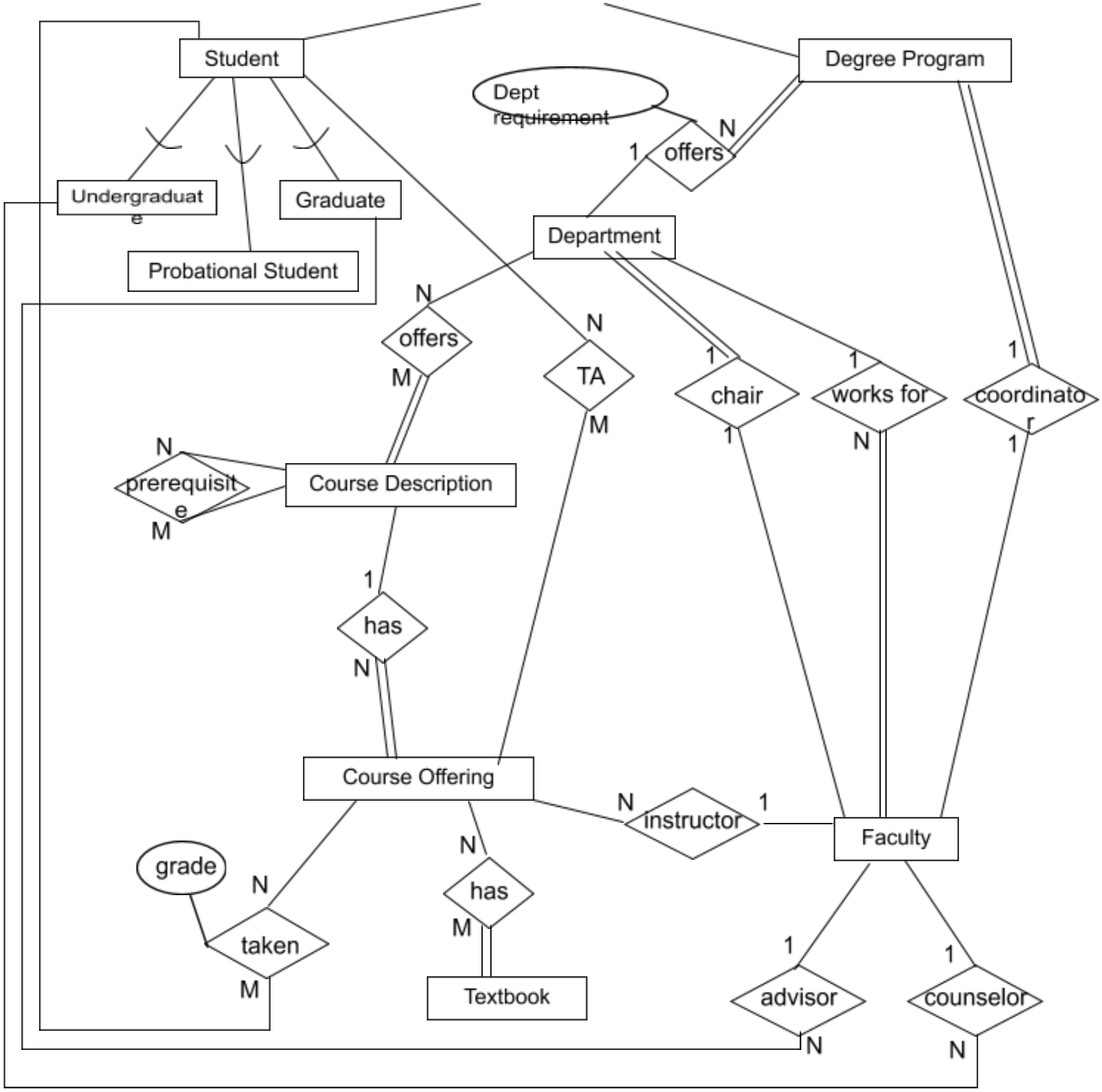
You need to work in a SQL client software, e.g. MySQL Workbench, Navicat (not free), SQLYog Community Version (Windows), Sequel Pro (Mac) etc.
Don't use terminal this time.

Name: Alfredo Perez

Due: Sep 11

1. Requirement Specification
   Consider the following set of requirements for a student database that is used to keep track of student information in a university.
   (a) The database should keep track of a student's name, id, social security number, current address and phone number, permanent address and phone number, birth date, sex, class, major department, minor department (if any), and degree program (B.A., B.S., etc.).
   (b) Course description has course title, course number, short description, number of credit hours, offering department, prerequisite courses, set of text books and semester offered.
   (c) Course offering contains sequence number, year, semester, time, place, instructor and TAs.
   (d) The department has the following information: department name, department location, phone, chairperson, faculty members and degree programs.
   (e) Other information such as course grades, grade point average, student advisors, probation status and graduate/undergraduate level should be captured in the schema design.
2. EER Diagram
   An EER diagram based on the above requirement specification is provided (see next page).
3. Add the following additional requirements to the EER diagram. Add new entity types and attributes as needed. Modifications can be made directly on the original EER diagram (please use a different color (e.g. purple) to make your additions explicit, also you MUST annotate your additions with the corresponding question numbers so it will be easier for me to grade, NO POINTS will be given if you fail to annotate your EER).
   (a) Departments have joint degree programs, that is, a degree program can be administered by more than one department.




Student
Degree Program
Dept requirement
N
1
offers
Undergraduate
Graduate
Probational Student
Department
N
offers
M
N
TA
M
1
chair
1
1
works for
N
1
coordinator
1
N
prerequisite
M
Course Description
1
has
N
Course Offering
N
instructor
1
Faculty
grade
N
taken
M
N
has
M
Textbook
1
advisor
N
1
counselor
N

(b) Graduate students can teach courses.
(c) Student can mentor other students (e.g., a senior student mentors some junior students). Assume that a student mentor can have several mentees. But a student mentee can only have at most one mentor.
(d) Each laboratory has a faculty as its director.
(e) A faculty can supervise multiple research projects.
(f) Students using certain labs for certain research projects.
(g) Every graduate student MUST have an advisor.

ANSWER

(A)
Department Offers Degree Programs
(Many to Many)

(B)
Graduate Teaches Course Offering
(Many to Many)

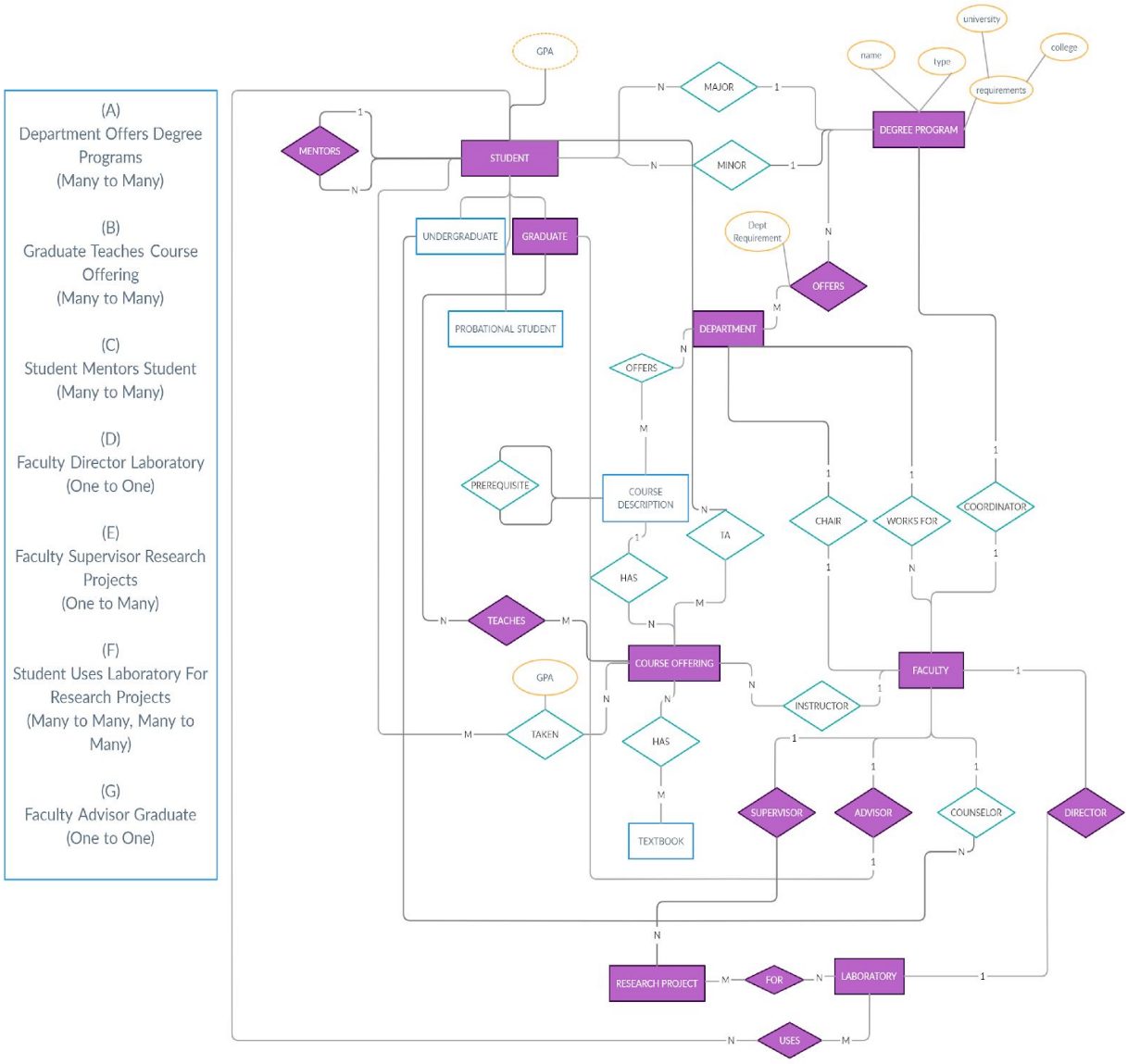
(C)
Student Mentors Student
(Many to Many)

(D)
Faculty Director Laboratory
(One to One)

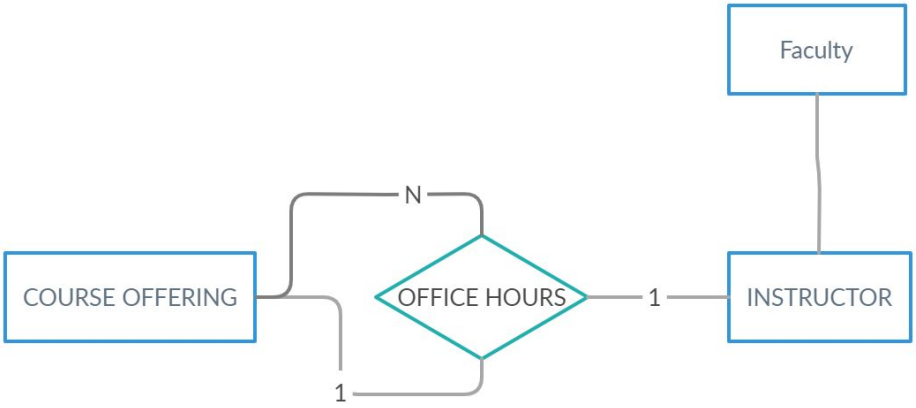
(E)
Faculty Supervisor Research Projects
(One to Many)

(F)
Student Uses Laboratory For Research Projects
(Many to Many, Many to Many)

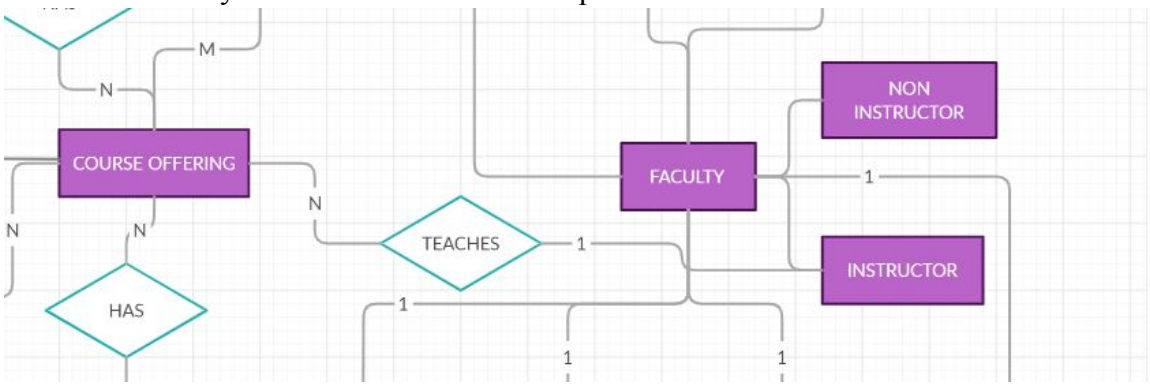
(G)
Faculty Advisor Graduate
(One to One)



4. An instructor may have office hours which
(a) corresponds to each course, or
(b) corresponds to each instructor (i.e., same office hours for all the courses that the instructor teaches)
How can each one of the above two cases be incorporated, independently, in the EER diagram?
Answer: You only need to show the relevant part of the above EER here.



5. Not all faculty members are teaching. Consider creating a new entity Instructor and making it a subclass of Faculty. Make necessary modifications to the EER diagram.
Answer: You only need to show the relevant part of the above EER here.



6. In this part of the lab assignment, you will learn to implement 1:1, 1:N and N:M relationships in MySQL.

   (a) We implement the 1:1 relationship c*hair* between Department and Faculty in the EER diagram. Since Department totally participates in the relationship, we implement the foreign key for the relationship in the Department table. Why?
   Answer:

   The details of the Chair, in this case their ssn, is the primary key of the Faculty Entity and it is what links the two ENTITIES so the foreign key had to be created in Department Table since the other holds the primary key

   Open your MySQL Workbench and connect to MySQL in VMWare, then write the following (Replace *YourName* with your real name):

```
CREATE DATABASE IF NOT EXISTS UNIVERSITY_YourName;
 USE UNIVERSITY_YourName;
```

   Since the foreign key in the Department table is referencing the Faculty table, we need to create the Faculty table first. Create a file called lab3.sql. Add the following SQL commands into lab3.sql.

```
/*drop all tables if they exist ignoring the foreign key
constraints*/
SET FOREIGN_KEY_CHECKS=0;
DROP TABLE IF EXISTS faculty;
DROP TABLE IF EXISTS department;
DROP TABLE IF EXISTS courseoffering;
DROP TABLE IF EXISTS coursedescription;
DROP TABLE IF EXISTS dept_offers_coursedesc;
DROP TABLE IF EXISTS textbook;
```

```
SET FOREIGN_KEY_CHECKS=1;
 create table faculty (
   facssno varchar(9) primary key,
   facname varchar(30)
   );

 insert into faculty values ('000000000', 'John Doe');
 insert into faculty values ('111111111', 'Mark Smith');
 select * from faculty;

 create table department (
   did integer(7) primary key,
   deptname varchar(30),
   chair varchar(9),
   FOREIGN KEY (chair) REFERENCES faculty(facssno));

 insert into department values (1, 'CS', '000000000');
 select * from department;
```

Run the SQL statements in lab3.sql in MySQL workbench. Based on the statements in lab3.sql, who is the chair of CS department?

Answer:

JOHN DOE

Append the following statements in lab3.sql:

```
insert into department values (2, 'MS', '000000000');
select * from department;
```

Who is the chair of MS department?

Answer:

JOHN DOE

The insertion of the second tuple is semantically wrong. Why?

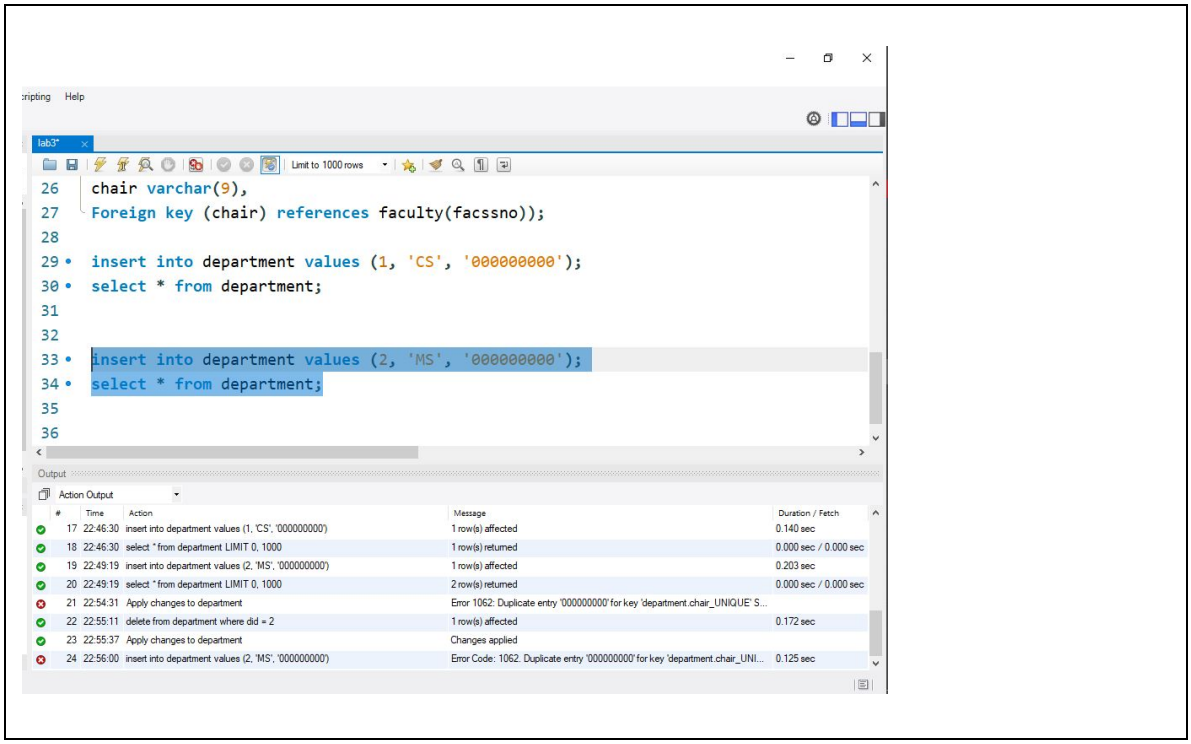
Answer:

Every Department must have one Chair and every Chairperson must Chair only one Department. In this case JOHN DOE is chairing both CS and MS department and that makes is semantically wrong

The MySQL system does not detect the problem! To prevent inserting the second tuple, you need to make the foreign key attribute (chair) in Department UNIQUE.

```
create table department (
  did number (7) primary key,
  deptname varchar (30),
  chair varchar (9) unique,
  FOREIGN KEY (chair) REFERENCES faculty(facssno)
  );
```

Go back to MySQL Workbench and rerun the SQL statements in lab3.sql. Observe that the second tuple cannot be inserted now.
Screenshot:



Comment out the second insert using /**/

```
/*
insert into department values (2, 'MS', '000000000');
select * from department;
*/
```

(b) 1:N relationship is implemented almost the same as 1:1 relationship except that UNIQUE is not needed. Foreign key should be defined in the table on the N side.
Append the following SQL commands in lab3.sql.

```
create table coursedescription (
  cno varchar (10) primary key,
  title varchar (50),
  credits integer (1),
  description varchar (200));

create table courseoffering (
  seqid integer (5) primary key,
  semester varchar (6),
  year integer (4));
```

Now implement in lab3.sql the 1:N relationships *has* between CourseDescription and CourseOffering and *instructor* between Faculty and CourseOffering. Make sure that your implementation in lab3.sql works in MySQL workbench.

Answer (copy your SQL implementation of the above description here):

HAS

```
alter table courseoffering
add column cno varchar(10)
references coursedescription(cno);
```

INSTRUCTOR

```
alter table courseoffering
add column instructor_num varchar(9)
references faculty(facssno);
```

(c) For an N:M relationship, a new table should be created. For example, the N:M relationship *offers* between Department and CourseDescription can be implemented in lab3.sql as follows:

```
CREATE TABLE dept_offers_coursedesc (
  did INTEGER (7),
  cno VARCHAR (10),
  PRIMARY KEY (did, cno),
  FOREIGN KEY (did) REFERENCES department(did),
  FOREIGN KEY (cno) REFERENCES coursedescription(cno)
);
```

Go back to MySQL workbench and rerun lab3.sql.

Add the following statement in lab3.sql.

```
create table textbook (
  isbn varchar (10) primary key,
  textname varchar (30),
  publisher varchar (50)
);
```

Now implement in lab3.sql the N:M relationship *has* between CourseOffering and TextBook.
Answer (copy your SQL implementation of the above description here):

```
create table courseoffering_has_textbook(
isbn varchar(10),
seqid integer (5),
primary key (isbn,seqid),
foreign key (isbn) references textbook(isbn),
foreign key (seqid) references courseoffering(seqid)
);
```