Question 1: ________ (30)
Question 2: ________ (20)
Question 3: ________ (10)
Question 4: ________ (10)
Question 5: ________ (20)
Question 6: ________ (10)
TOTAL: ________ (100%) * 1.5 = ________ Points (out of 150)
1. (30 points) A database is created that stores information about students, courses, and instructors. The database consists of five relations defined in SQL as follows:

CREATE TABLE Student (StuID INTEGER PRIMARY KEY, 
                       Name VARCHAR(50))

CREATE TABLE Course (CourseID VARCHAR(20) PRIMARY KEY, 
                      InstructorID INTEGER, 
                      Title VARCHAR(50), 
                      MaxEnrollment INTEGER, 
                      CurrentEnrollment INTEGER)

CREATE TABLE Student-Course (StuID INTEGER, CourseID VARCHAR(20))

CREATE TABLE Instructor (InstructorID INTEGER PRIMARY KEY, 
                         Name VARCHAR(50), 
                         Email VARCHAR(30))

CREATE TABLE Course-Instructor (CourseID VARCHAR(20), 
                                 InstructorID INTEGER)

Students can enroll in multiple courses. Each course has one instructor, but an instructor can teach multiple courses. The MaxEnrollment attribute for courses provides the maximum number of students that will be allowed to enroll in a course. The CurrentEnrollment attribute contains the number of students actually enrolled in the course.

Write an SQL statement that will:

(a) (5 points) increment the current enrollment for the course with CourseID MA1001

(b) (5 points) determine the average number of students actually enrolled in courses
(c) (8 points) list the names of all students enrolled in the course that has the title \textit{Intro to CS}

(d) (12 points) list the names of all students in all courses taught by the instructor named Hamel. Duplicate names are OK.
2. (20 points) Draw a set of relations (tables) to represent the following Java class definitions. You don't need to provide the SQL data types, just the table names and attribute names. For example, if you were asked to draw a table to represent

```java
class Customer{
    String name;
    String address;
}
```

you would draw this:

```
Customer
+--------+-----------+
<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>--------</td>
<td>-----------</td>
</tr>
</tbody>
</table>
+--------+-----------+
```

Here are the class definitions:

```java
class Book{
    private int isbn; // the key for identifying a book
    private ArrayList<String> authors; // authors’ names
    private String title;
    ...
}

class Library{
    private String address;
    private ArrayList<Book> books;
    private String director; // name of the library director
    ...
}
```

You may draw your tables on the next page.
(problem 2 answer goes here)
3. (10 points) Answer one of the following questions. If you answer both questions, only the first one will be graded.

- What is the meaning of the @Before annotation as it pertains to JUnit? What kinds of operations are typically done in a method annotated with @Before?

- In Homework 4, you created an enumeration type to represent a grade in a course:

  ```java
  public enum GradeType {A, B, C, NR, IP}
  ```

  What is the main advantage of using an enumeration type instead of simply using the type String to represent the grades?
4. (10 points) Assume you define a collection class as follows:

```java
class MyClass<T> implements Iterable<T>{
    ... // field and method definitions would go here

    // the Iterable interface requires implementation of this method
    Iterator<T> iterator(){
        ... // implementation would go here
    }
}
```

In a main program, you create a MyClass object, and an iterator for the object:

```java
MyClass<Integer> mcObject = new MyClass<Integer>();
// ... code that populates mcObject with Integers would go here

Iterator<Integer> mcIt = mcObject.iterator();
```

Write a snippet of Java code that would sum up all the integers in `mcObject`, and display the sum. You must use the iterator to cycle through `mcObject` (you may not use a for-each loop).
5. (20 points)

(a) A __________ object can create statement objects that are used to execute SQL commands.
   i. ResultSet
   ii. Connection
   iii. Statement
   iv. JDBC

(b) If your program needs to look at several result sets at the same time,
   i. you need to create multiple Statement objects
   ii. you can reuse the same Statement object
   iii. you should close one ResultSet before creating another one
   iv. there is no way to look at several result sets at the same time

(c) If you want to determine properties of a result set from an unknown table, you need to use
   i. the ResultSet interface
   ii. the ResultSetMetaData interface
   iii. the Connection interface
   iv. the Statement interface

(d) What is wrong with the following statement? (Assume a Connection has been made to the database described in Problem 1, and the Statement s has been created.)

   ```java
   ResultSet rs =
   s.executeQuery("INSERT INTO Student VALUES (143, 'Jones')");
   ```
   i. ResultSet should be changed to Statement
   ii. ResultSet should be changed to Connection
   iii. executeQuery should be changed to executeUpdate
   iv. the double quotes (" ) should be changed to single quotes (’ ).
6. (10 points) Answer these questions pertaining to the Model-View-Controller design pattern.

(a) For a GUI-based application, in which component (model, view, or controller) should the `actionPerformed` method(s) reside?

(b) True or false: the main reason for using the Model-View-Controller design pattern is to make the controller reusable in other programs.

(end of exam)