Written Assignment #1
Due Wednesday April 15

⇒ **Please make yourself a copy of your solution to this assignment before turning it in. You will need it for Written Assignment #2.**

On-campus students: You may turn in your written assignments during class, or you may give them to Sharon Lambeth, the course secretary, in Gates 419. If Sharon is not in, please slip your assignment under her door. Written assignments are due at 5:00 PM on Wednesdays. Assignments less than 24 hours late will be penalized 10% or will use one of your four "chits". Assignments more than 24 but less than 48 hours late will be penalized 30% or will use two of your four "chits". No assignments will be accepted more than 48 hours late—i.e., no written assignments will be accepted after 5:00 PM on Fridays.

SITN students: For written assignments, assignments due on Wednesday must be timestamped by the Thursday courier. Late assignments timestamped by the Friday courier will be penalized 20% or will use two of your four "chits". Assignments timestamped later than Friday will not be accepted. Assignments not sent by SITN courier are subject to the late policy for on-campus students specified above. No late written assignments are permitted for remote SITN students.

---

1. You are to design a database that maintains information for producing a weekly television guide for a given region (such as Northern California). The data should include information about television shows, television networks, cities, channels, show times, etc. For starters, you may make the following assumptions:

   - A given channel in a given city is associated with one network.
   - A given show is either owned by a network (and shown on a channel associated with that network) or is a local show and may be shown on any channel.
   - Not all shows are shown in all cities, and the days and times for a given show may differ from city to city.
   - You may ignore cable channels, which generally are not city-dependent.

   Please feel free to make additional assumptions about the real world in your design, as long as the assumptions are reasonably realistic and are stated clearly as part of your solution.

   (a) Specify an ODL schema for your database. In addition to class (`interface`) definitions with attributes and relationships, don't forget to include keys and inverse relationships.

   (b) Specify an entity-relationship diagram for your database. Don't forget to underline key attributes and include arrowheads indicating the multiplicity of relationships.

   Note that this question is fairly open-ended and there is no single right answer, but some designs are better than others.

2. Find a site on the World-Wide Web for which you're quite sure there is a database management system operating behind the scenes. While we will be more impressed if you find your own site, here are a few possibilities: The All-Music and All-Movie Guides (both at `www.allmusic.com` and `www.allmovie.com`), `www.cbooks.com`, `www.amazon.com`, `www.cdnow.com`. Rest assured that there are many, many more DBMS-powered sites out there.

   (a) What is the URL for the site you selected?

   (b) Briefly describe, in English, the data managed by the site and the queries available to the user.

   (c) Specify a possible schema for the database. You may use ODL class definitions or an entity-relationship diagram, whichever you prefer (but you will have to choose the other for problem #6). If you choose ODL, don't forget to include keys and inverse relationships. If you choose E/R, don't forget to underline key attributes and include arrowheads indicating the multiplicity of relationships.

3. Specify an ODL class definition for any real-world domain of your choosing such that the class realistically includes a relationship that is its own inverse. Please do not use an example that was given in lecture or the help session.

4. Consider an entity-relationship diagram consisting of two entity sets E_1 and E_2 connected by a binary relationship set R. Suppose that R has a single attribute A. Under what conditions can attribute A be moved to become an attribute of entity set E_1 instead of an attribute of relationship set R, without fundamentally altering the design?

5. This question is based on Figure 2.12 (page 45) and Figure 2.15 (page 48) of the Ullman/Widom textbook. Figure 2.12 depicts a four-way relationship set *Contracts*. The arrows specifying multiplicity encode the following two constraints: (1) Given a star, a movie, and a producing studio, there can be only one studio that "owns" the star; (2) Given a star, a movie, and the star's studio, we can uniquely determine the producing studio.

   (a) In Figure 2.15, relationship set *Contracts* is replaced by an entity set and four binary relationship sets. Do the arrows specifying multiplicity in Figure 2.15 describe the same set of constraints as Figure 2.12? If yes, briefly state the reason. If no, describe a scenario that satisfies the constraints specified in one figure but not the other.

   (b) Now suppose we are only interested in keeping one contract for each movie (perhaps just the contract for the "hottest" star in the movie). In other words, we want to specify the following constraint: Given a movie, we can uniquely determine the (hottest) star, the studio of this star, and the producing studio. Can we modify the arrows in Figure 2.12 to encode this constraint? What about Figure 2.15? Do not modify other aspects of the E/R diagram except the arrows.

6. **Personal Database Application (PDA)**

As the course progresses, your programming project will be to build a substantial database application for a real-world scenario of your choosing. You will design a schema for the database, and you will create an actual database using a relational database management system. You will populate the database with sample data, write interactive queries and modifications on the database, create programs that manipulate the database, and develop user-friendly tools for interacting with the database.

Your first step is to identify the scenario you would like to manage with your database, and to construct an ODL or entity-relationship schema design for the data. We suggest that you pick an application that you will enjoy working with, since you'll be stuck with it for the whole quarter! In previous years, students who built a database about something they were interested in—a hobby, a favorite Web site, material from another course, a research project, etc.—got the most out of this part of CS145.

Try to pick an application that is relatively substantial, but not too enormous. For example, when expressed in the entity-relationship model, you might want your design to have in the range of five or so entity sets, and a similar number of relationship sets. Note that this is a ballpark figure only! You should certainly include different kinds of relationships (e.g., many-one, many-many) and different kinds of data (strings, integers, etc.), but your application need not necessarily require advanced features such as subclassing in ODL, or weak entity sets or roles in E/R.

(a) Write a short (approximately one paragraph) description of the database application you propose to work with throughout the course. Your description should be brief and relatively informal. If there are any unique or particularly difficult aspects of your proposed application, please point them out.

(b) Specify a schema for your proposed database. Use either ODL class definitions or an entity-relationship diagram, whichever you did *not* use in problem #2. As always, when using ODL don't forget to include keys and inverse relationships, and when using E/R don't forget to underline key attributes and include arrowheads indicating the multiplicity of relationships.

If you're having trouble thinking of an application, or if you're unsure whether your proposed application is appropriate, please feel free to consult with one of the course staff.