Name: _____

CSSE 373—Formal Methods in Specification and Design

Exam 1, Apr. 9, 2009

This exam is open book and open notes. You may also use your laptop if you wish to check your solutions in Alloy or review past homework, though you should budget your time very carefully. You may *only* use your laptop to access data on your local hard drive or directly accessible from the course ANGEL or web pages.

All of the problems on the exam concern the example specification given on page 5. You may carefully remove that page from the exam, but should turn it in when done.

If you need additional paper to complete a problem, please use one of the provided blank sheets and attach it to your exam. Please note on the printed exam when you do so.

Problem	**Poss. Pts.**	**Earned**
1	12	_____
2	12	_____
3	12	_____
4	8	_____
5	20	_____
6	20	_____
7	16	_____
Total	100	_____

1. (12 points) Suppose for a particular instance of the model, the enrolled relation had the value:

```
{C0->S0->T1,
 C0->S1->T0,
 C1->S0->T0,
 C1->S1->T0,
 C1->S1->T1}
```

Give the value of each of the following Alloy expressions:

a. C0.enrolled[S0] = _____

b. C1.enrolled[S1] = _____

c. Course.enrolled.Term = _____

d. #(Course.enrolled) = _____

2. (12 points) Write an Alloy function that returns all of the prerequisites of a given course, both the direct prerequisites, and the prerequisites of the prerequisites, and so on.

```
fun allPrerequisites[c: Course]: set Course {
```

3. (12 points) Write an Alloy function that returns the grade earned by a given student in a given course during a given term. The function header is provided

```
fun grade[s: Student, c: Course, t: Term]: Grade {
```

4. (8 points) What constraints are imposed by the fact below (assuming a correct implementation of the grade function from problem 3)?

```
fact {
  all c: Course {
    c not in c.^prereqs
    all s: Student, t: Term |
      some grade[s,c,t] => s in c.enrolled.t
  }
}
```

5. (20 points) Write a fact that states that a student cannot retake any course for which they earned a C or higher. (Note that Terms are totally ordered. You may assume that a correct implementation of the grade function exists, even if you did not complete problem 3.)

```
fact OnlyDeficientRetakes {
```

6. (20 points) Write a predicate hasPrereqs that takes a student, a course, and a term. The predicate checks whether the student has satisfied the prerequisites for the course. Assume that "satisfying the prerequisites" means earning a *non-failing* grade in all of the prerequisite courses prior to taking the given course. (Note that Terms are totally ordered. You may assume that a correct implementation of the grade function exists, even if you did not complete problem 3.)

```
pred hasPrereqs[s: Student, c: Course, t: Term] {
```

7. (16 points) Suppose we wanted to extend the model to also track the class period and room used for each offering of a course.

a. Give the additional signature(s) and relations necessary for this extension:

b. In this extension we want to make sure that two courses aren't scheduled for the same room at the same time. Write a fact that ensures this (or argue how any multiplicity constraints in your relations already do so):

```
module exam1/registrar

open util/ordering[Term]

sig Course {
  -- the prerequisites of this course
  prereqs: set Course,
  -- the students enrolled in this course for a given term
  enrolled: Student -> Term,
  -- the grades earned by the students in this course for a given term
  grades: Student -> lone Grade -> Term
}

sig Term {
}

sig Student {
}

abstract sig Grade {}
one sig A, B, C, D, F extends Grade {}
```