CSE 235 **Homework 07** **Fall 2008**

Due Oct 27, 2008

Name _____ CSE Login _____

Problem	Page	Points	Score
8.1.4	527	4	
8.1.6	527	8	
8.1.24	528	4	
8.1.28 bc	528	2	
8.1.32 aceg	528	4	
8.1.34 bdfh	528	4	
8.3.4 c	543	2	
8.3.10 ace	543	9	
8.3.12	543	2	
8.3.14 abc	543	6	
8.4.16	554	6	
8.5.2	563	5	
8.5.30	563	4	
8.5.46(abcde)	565	5	
Total for exercices		65	
Typesetting in LaTeX (bonus)		10	
Programming		See below	
Correctness		35	
Style & Documentation		5	
Grand Total		105	

Instructions Follow instructions carefully, failure to do so may result in points being deducted.

- The homework can be submitted on paper or via handin. Homework neatly formatted in LaTeX will receive a 10 point bonus. You will not receive the 10 points bonus if you work with a partner (see below).
- Clearly label each problem and submit the answers in order.
- Staple this cover page to the front of your assignment for easier grading.
- Late submissions will not be accepted.
- Show sufficient work to justify your answer(s). *Sloppy, hard to read papers will not be graded* (please have mercy...).
- When you are asked to prove something, you must give as formal, rigorous, and complete a proof as possible.
- You are to work individually, and all work should be your own. Check partner policy below.

- The CSE academic dishonesty policy is in effect (see `http://www.cse.unl.edu/undergrads/academic_integrity.php`).

Partner Policy You may work in pairs, but you must follow these guidelines:

1. You must work on all problems together. You may not simply partition the work between you.
2. You must use LaTeX and you may divide the typing duties however you wish.
3. You may not discuss problems with other groups or individuals.
4. Hand in only one hard copy under the first author????s name.

Programming Assignment

The goal of this assignment is to familiarize yourself with relations by designing and implementing a relation ADT in either Java or `C++`. In addition, you will write several functions that will support relation operations.

Specifically, you will use a predefined class with member methods which can be found in files available on the course web page. You will have to first decide how you intend to represent the relation. However you choose to implement the relation, it *must* be dynamic, not static (refer to the constructors).

You will then have to populate the various constructors and class methods. Details can be found in the actual files on the web page. All methods *must* be implemented and *will* be tested rigorously. You may *add* any member functions/variables and even use other classes if you wish, but you must *not* change the naming conventions or parameters of the methods already defined. *How* you implement these algorithms is up to you and is, in fact, dependent on how you decide to represent the relation.

It is *highly* suggested that you create your own test driver and test your class on as many different relations as you can since this is how I will be evaluating your program. However, you will only hand in the source code needed to compile your class. If you choose C++, then you will need to handin `relation.h`, `relation.cpp`, your `makefile` as well as any other files that your class depends on. If you choose to do the assignment in Java, then you'll only handin `Relation.java`, your `makefile` and any other support files.

For C++ users, the `makefile` will simply compile (but not link) all your classes into `.o` object files (using g++), it shouldn't make any executable (I'll take care of this). For Java users, your make file should compile it into a `class` file.

Correctness will be dependent on whether or not your algorithms and functions work; if your files compile etc. Style/documentation points will be awarded based on the readability of your code and how well it is commented.

Note: It may be helpful to make use of Exercises 8.1.49-52, and Exercises 8.4.25-26.