CSE 235 Homework 03 Spring 2006

Due 12:30AM, Friday, March 3rd, 2006

Name/CSE Login _____

Name/CSE Login _____

Instructions Follow instructions *carefully*, failure to do so may result in points being deducted. Clearly label each problem and submit the answers *in order*. It is highly recommended that you typeset your homework using LaTeX or a similar typesetting system. Staple this cover page to the front of your assignment for easier grading. Be sure to submit any programming files via the webhandin (`http://www.cse.unl.edu/~cse235/handin`). Late submissions *will not be accepted*. Be sure to show sufficient work to justify your answer(s). If you are asked to prove something, you must give as formal, rigorous, and complete proof as possible. You are to work individually, and all work should be your own. 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 (including programming) *together*. You may not simply partition the work between you.
2. You must use LaTeX, but 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 and one copy of your program files; do so under the first author's name.

Problem	Page	Points	Score
7.1.6abcd	480	5	
7.1.34abc	481	5	
7.1.46	481	5	
7.1.48	481	5	
7.3.14	495	4	
7.3.16	495	4	
7.3.32	496	4	
7.4.18	506	4	
7.4.28ad	507	4	
7.5.2	513	4	
7.5.32	514	4	
7.6.16ab	528	4	
7.6.28	529	4	
7.6.38	529	4	
BONUS: 7.2.14*	488	3	
BONUS: 7.2.18*	488	3	
Program			
Correctness		35	
Style/Documentation		5	
Total		100	

*-Material not covered in class, to read on your own.

Note: Exercise 7.1.49–52 may be useful.

Programming Assignment

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

Specifically, you will use a predefined class, `relation` in the files `relation.h` and `relation.cpp` available on the 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.

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 `relation.h`, `relation.cpp`, your `makefile` as well as any other files that your class depends on. The `makefile` will simply compile (but not link) all your classes into `.o` object files, it shouldn't make any executable (I'll take care of this).

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.