University of North Texas
CSC 330: Advanced Data Structures
Prof. Stephen R. Tate

Handout 3
August 26, 2009

Submission Procedures and Grading Guidelines

Submission Procedures: In this class you will be using a program called "`330submit`" to turn in your programs electronically. This program sends copies to me so that they can be checked and graded. `330submit` is run from the command line on the `cmpunix` system. The general form of the command is "`330submit programno files`", where "`programno`" is of the form "`program1`", "`program2`", and so on, and "`files`" is a list of files which may contain wildcards.

I highly recommend that when you do an assignment, you have a directory just for that assignment. When you're finished, clean out all object files and any extra stuff that's in the directory (this is really easy if you put a "`clean`" target in your `Makefile`), and then run `330submit` with the wildcard "`*`" as the file name. This submits all files in your current directory, which should include all the files that make up your program. For example, to do this for Programming Assignment 2, you would type

```
330submit program2 *
```

One important thing to note is that the case of the assignment name is significant. It will *not* work if you use the upper case version "PROGRAM2". As another example, if you wanted to explicitly give all the file names, then the files for programming assignment 2 might be submitted with the command:

```
330submit program2 Makefile ListStack.* BlockStack.*
```

Note that you should be sure to turn in your `Makefile`!

There's also a special assignment name of "`question`" that you can use to submit code to me that you have a question about. If you need to ask me a question about a program (and have tried to contact the grader first!), you can send me a copy of your code by using

```
330submit question *
```

Grading Guidelines: Your programs will be judged on several criteria, which are shown below, along with the questions that they address. In addition to the percentages below, the style guidelines give some "major rules" with specific point-value penalties for violating — these will be applied regardless of whether the total penalty exceeds the amount for "style" given below.

Correctness (50 percent): Does the program compile correctly? Does the program do what it's supposed to do?

Design (20 percent): Are operations broken down into functions and procedures in a reasonable way? Are ADT's clearly defined, implementation independent, and sensible?

Style (10 percent): Is the program indented properly? Do variables have meaningful names?

Robustness (10 percent): Does the program handle erroneous or unexpected input gracefully?

Documentation (10 percent): Do all program files begin with a comment that identifies the author, the contents, and the compiler used for that particular file? Are all the procedures and functions clearly documented using Doxygen comments? Are unclear parts of code documented?

Notice that correctness is only half of the grade. Design and style are very important: you write an English paper by first laying out how you want the information presented, then writing a rough draft that makes all your points, and then polishing the style of that rough draft. Writing a program is much the same: first design how things should be organized, then do a "rough draft" that consists of a working implementation, and then go back and polish your code, adding all the required comments and make sure style guidelines are followed. Unfortunately, many people hand in their "rough draft" programs, thinking that a working program is all they need. If you do that in this class, you probably will not pass.

A program that does not compile will get at most 50% of the possible points for an assignment. If you add comments after getting your program working, then be sure you test it again before submitting it. It is actually fairly easy to make working code inoperable when adding comments (by forgetting to close a comment, for example). Even if your program is entirely correct except for a small mistake of this type, you will still get a maximum of 50%. We will use exactly the same environment to grade your assignments as you use for testing, so there is no excuse for a program that doesn't compile.

In addition, most programming assignments are given with some sample inputs and outputs. If your program does not work correctly on those sample inputs, you will get at most 75% of the possible points for an assignment. Your programs will also be tested on other inputs, but be sure, at the very least, that your programs work correctly on the supplied examples.