Grading Rubric for Programming Assignments, Spring 2010, CSCE 240

1.1 Introduction

The programming assignments for this course have point values 20, 35, 35, 35, 35, 45, 45. The first assignment is different from the rest in that the primary purpose of the first assignment is to ensure that you can access all the bits of software necessary to create and run programs for this course.

The general breakdown of point value for the latter six assignments is as follows:

- 15 points for correct execution (25 for the last two assignments);
- 10 points for correct style and structure;
- 10 points for correct documentation.

The purpose of this document is to indicate what you will or will receive positive points for in your code. This is intended to summarize in a concrete way the content of the (much-longer) style sheet.

1.2 Execution

You should not expect to receive *any* points for execution if your program fails to compile, if it crashes, or if it fails to produce any output. You will be given the standard script that will be used for compiling and executing your program, so there is no excuse for this kind of error.

You should therefore write your programs in such a way that the submitted program actually compiles and executes, even if it does not produce the correct output. You will be given credit for partially correct output that can actually be viewed and seen to be partially correct. If you have the same partially correct data, but your program crashes before displaying that data, there is no way to determine that you had something partially correct and you can expect to receive no points.

Most programs in this class have two or three fundamental issues of logic and program design. You will lose roughly 3 to 5 points for each failure of logic.

Each fencepost error, for example, is likely to cost you 3 points.

A logic error of nested `if` statements that don't quite do what you need to do is likely to cost you 3-5 points.

Improperly-aligned columns in an assignment where alignment was required is likely to cost you 3-5 points, depending on the severity.

Declaring a global variable instead of passing a parameter, if the parameter passing was a requirement, will cost you 3-5 points.

And so forth.

1.3 Style and Structure

Part of the lesson to be learned in this course is how to write software that you can hand off to someone else, software that can be modified and maintained, and software that is written in a way that is clearly not just a hack job.

Correct execution is therefore only a minimal requirement, and you will be graded on style and structure. The numbers in parentheses are an indication of the points you might lose if you fail to follow directions. I reserve the right to combine repeated failings into one greater failing, or to consider later in the semester that you should have learned by then and should therefore be penalized more for failing to have learned. For example, if the penalty for failing to indent consistently is 2 points, and you fail to indent consistently everywhere in several places, then you are likely to lose perhaps 5 points overall if this is Homework 3, but you might lose many more points if this is Homework 7.

For an indication of what is correct and acceptable, read the style sheet. This document only lists point values; it does not attempt a complete explanation of good and bad things to do or not do.

1.3.1 Purely Stylistic Things

You must indent consistently throughout a given homework assignment, and this indentation must be compatible with the standard Linux editors and printers in the labs. (2 points)

You must use consistent and appropropiate style for control structures, functions, braces, and such. (2 points)

Your program must display and print without ugly line wrap problems on the standard Linux editors and printers in the labs. (2 points)

1.3.2 Matters of Taste and Elegance in Structure

A number of things in this course have to do with The Right Way to Write Programs. You can expect to lose more points for the failings below later in the semester than you might lose early in the semester. In general, the best way to avoid losing such points is to think carefully whether your program structure is really the right structure. The following are some of the problems for which you can lose points.

All variables should be declared at the top of the block of code in whose scope they are contained. (2-5 points, depending on severity)

Magic numbers are forbidden. (2-5 points, depending on severity)

Variables should be declared in the proper scope, as local as possible. (2-5 points, depending on severity)

By the second time you write the same block of code, it should become a separate function. (2-5 points, depending on severity)

Variable and constant names should conform to C++ norms and should be meaningful and correct. (2-5 points, depending on severity)

If there is a C++ construct for doing something, and you use an older C construct (such as with type casting) you should use the C++ construct. (2-5 points, depending on severity)

Side or secondary effects or actions in a function should be avoided. (2-5 points, depending on severity)

1.4 Documentation

The model for documentation is that which is necessary in a Java program to be able to produce the same kind of documentation as is produced by `javadoc`.

The numbers in parentheses are an indication of the points you might lose if you fail to follow directions. I reserve the right to combine repeated failings into one greater failing, or to consider later in the semester that you should have learned by then and should therefore be penalized more for failing to have learned. For example, if the penalty for failing to document parameters passed to a function is 3 points, and you fail to document parameters in five functions, then you are likely to lose perhaps 6 points overall if this is Homework 3, but you might lose all points if this is Homework 7.

The following is not an exhaustive list of possible problems.

The files must contain sufficient documentation that they all be identifiable without resorting to file names. That is, there should be no loss of information if a reader is in possession of nothing but a printed copy. (3 points)

The overall program needs to have your name(s), an edit/submission date, and a brief description of what the program is or does. (3 points)

All functions must have a similar header line, some following text that describes in greater detail what the function does, and a bulleted list of input parameters, whether or not output is produced, and values returned by the function. (3 points)

All variables should be documented as to their use in the program. Inline documentation of blocks of code should be included but need not be excessive. (3 points)

Any "gotcha" situations should be explicitly documented, especially including such things as potential fencepost errors, clever tricks used in pointer or other arithmetic, etc. If it isn't routine and obvious, you should include at least one line to point out to a reader that there is something that isn't routine and obvious. (3 points)

Jargon terms must be spelled correctly. Typos can be excused; repeated misspellings of words used in the discipline are not acceptable. (1 point each word)