Programming Rules

NOTE. A rule is a *requirement*; it must be followed. A guideline is a *suggestion*; it is strongly encouraged but does not have to be followed. The following are rules; to receive full credit on an assignment they must be followed.

1. You must make sure that your program is free of all errors when it is compiled, linked, and executed on the department's network gateway machine, named `eniac.geo.hunter.cuny.edu`, prior to submitting it. Sometimes a program will run correctly on one machine but not another, for one reason or another. This requirement stipulates that it must run correctly on `eniac`.

2. You must submit all of the source code and nothing else, unless the assignment states otherwise. Do not submit an executable, or data files.

3. For full credit, an assignment must be submitted to me in the manner described in the assignment by the end of the day on the due date. Each day after the due date, the program loses 20% of its total value. A program that is five days late will be scored 0.

4. *The program must be your work, and your work alone.* If you do not understand what it does or why it works because someone else's hand is in it, I will discover that one way or another. You are forewarned that I think it is reasonable for you to explain to me how your program works if I ask you to do so. If you cannot explain it, then it is not "yours". Attributing someone else's work as your own is plagiarism, and it is a violation of Hunter College policy. I will file an official complaint against any student who I believe has committed plagiarism.

5. *Every program must be professionally documented.* Every distinct source code file must contain a preamble with the file's title, author, brief purpose and description, date of creation, and a revision history. The description must be a few sentences long at the minimum. A revision history is a list of brief sentences describing revisions to the file, with the date and author (you) of the revision. This is an example of a suitable preamble:

```
/******************************************************************************
Title :        drawing_stars.c
Author :       Stewart Weiss
Created on :   April 2, 2010
Description :  Draws stars of any size in a window, by dragging the mouse to
               define the bounding rectangle of the star
Purpose :      Reinforces drawing with the backing-pixmap method, in which the
               application maintains a separate, hidden pixmap that gets drawn
               to the drawable only in the expose-event handler. Introduces the
               rubberbanding technique.
Usage :        drawing_stars
               Press the left mouse button and drag to draw a 5-pointed star
Build with :   gcc -o drawing_demo_03 drawing_stars.c \
                  `pkg-config --cflags --libs gtk+-2.0`
Modifications :
******************************************************************************/
```

   All function prototypes in your program, whether members of a class or not, must have a prologue containing comments for each parameter and appropriate pre-and post-conditions. These prologues must not be in the implementation files of classes, but in the class interfaces. All non-trivial algorithms must be documented in plain English in a multi-line comment block. All non-trivial declarations must have adjoining, brief comments. Documentation is usually worth 10% of the grade.

6. Every program must follow commonly accepted stylistic guidelines regarding the use of blank lines, white space, indentation, and naming of program entities such as variables, classes, functions, and constants. Your program must be consistent in its use of typographical format for distinguishing types, variables, functions, and constants. For example, you might decide that all type names should begin with an uppercase letter and all variables begin with lowercase letters, or that all variables use underscores to separate the words in the name, as in `number_of_scores` and `first_author`, or that they use changes in case, as in `numberOfScores` and `firstAuthor`. Style is usually worth 10% of the grade.

7. Every program must be correct to receive full credit. "Correct" means that for every possible input, it produces output that is consistent with the specification. If the program produces correct results for some, but not all, inputs, it is not correct. Since there may be infinitely many possible inputs, you cannot possibly establish your program's correctness by running it on all inputs. You must use a combination of sampling (i.e., testing) and logical analysis to convince yourself of its correctness. Correctness is usually 50% to 70% of the grade. A very common mistake I have found is for students to hand in programs that do not even run correctly on the input file I gave out. In other words, students have failed to check the outputs of their programs before submitting them. This is just laziness.

8. Every program must satisfy specified performance requirements if these are stated. This means that it uses an amount of storage and running time within specified or reasonable limits. Performance may be worth between 10% and 30% of the grade, depending on the assignment.

9. There are no rigid rules. All rules can be broken, but it is best to check with me before taking a big chance.