Ostermann's Programming Project Guidelines

(*version 4.2 - Mon Mar 31, 2008*)

Originality

Anything that you turn into my class is assumed to be solely the result of your own thought processes, research, trial and error, and creativity. The concept of *intellectual property* is particularly important in the sciences and engineering and will be emphasized in this class.

If parts of the program are from someone else (including me), then those parts of the program **must** be clearly marked as being not your own work. You *may* not get credit for parts of a project that you were supposed to write but got from somebody else.

If you turn in a project including work which is **not** your own original work and which you did **not** clearly mark as being the intellectual property of someone else, then you will fail the course.

Project Submission

All project submissions in my classes will be done electronically. To make this work, you need to do the following:

- Add the program directory /home/osterman/classbin to the end of your PATH. To accomplish this:
  - If you're running csh, which is the default:
    1. Use your favorite editor to edit the file `.login` in your home directory
    2. Make sure that the PATH section reads as follows:

       ```
       ## If you want the current directory included in your PATH variable,
       ## uncomment out the next  lines
       ## setenv PATH ${PATH}:.
       ```

    3. Just below that, add the lines

       ```
       ## add Dr. Ostermann's class bin
       setenv PATH ${PATH}:/home/osterman/classbin
       ```

  - If you're smart enough to **not** be running csh, then you already know how to make the appropriate changes in whatever shell you're running instead!!
  - *Note that there are security considerations when adding things to your PATH. I've asked you to add this to the END of your path. If you're not comfortable doing that, then you're being cautious, and that's to be commended. I recommend that you look in that directory and either make an alias to the commands that you need or make a symlink. I don't recommend that you copy them, however, as I may need to change them.*
- To turn in an assignment, do the following:
  - Set the current directory to be the directory containing the project that you want to turn in (and **ONLY** that project, please adopt a hierarchical classwork storage structure!!)
  - Run the program `xyzsubmit`, where `xyz` is replaced by the class number. For example, if you're in cs444, then you'd run the program `444submit`. Note that this program is in directory `/home/osterman/classbin`. If the shell responds with "not found", then you didn't add that directory to your path. The arguments to that program are:

    PROJNUM The required argument to the program is the project number. For the first project, you'll turn it in by typing, for example, `444submit 1`

    -f The "`-f`" argument means *force*, meaning that the program will **delete** all previous submissions of the project before submitting the new files.

-l The "`-l`" argument means *large*, meaning that the projects whose size exceeds 1MB can be submitted. This should not normally be necessary.

- Turn in requirements
  - You **MUST** use the program `make` to manage your projects. the `xyzsubmit` program will refuse to submit a program without a `Makefile`
  - You must use the gcc or g++ compiler and (at least) the options `-g -Wall -O2 -Werror`
  - Programs that do not compile will not be graded and will receive zero credit (see -Werror!).

Main program information

The file that contains the subroutine `main` must contain a banner comment at the top listing at least:

- Your name
- The class number (542, 444, etc)
- The name of the project
- A description of what the program does and how it works (at a high level)
- A listing of any known bugs/problems in the program

Alternately, this information can be put into a file called `README`

Programming Style

This is a course in *Computer Science*, not *Computer **Programming***. A Computer Scientist treats every programming project as a chance to learn something new and to improve his/her craft. How the program is written is at least as important as whether or not it works. To encourage the use of good programming style, all projects in this class must adhere to the following rules:

- Numeric constants are not allowed inside expressions, they must be declared as constant types (or MACROS) in all upper case, as in:

```
const int MAX_STRING 256;
```

- Each procedure definition must contain at least a one line comment telling what it does.
- Blocks of code with a single purpose inside a subroutine should be commented using some standard format.
- Programs must use standard indentation. I'm not going to specify which form to use, that's a religious issue. If you're not already comfortable with one style or another, I recommend the style in the Kernighan and Ritchie book.
- If a procedure is longer than 1 or 2 screens (or one printed page), it should probably be broken up into subroutines.
- If the lines of a procedure are indented by more than 4 or 5 "tab stops", then that probably means that the "deeply indented" part needs to be rewritten as a subroutine.

Good Advice

Modifying your program to meet these standards just before turning it in defeats the purpose of these guidelines and is a waste of your time. By this point in your careers, you should be working on understanding the *artistic* value of much of what we do and cultivating a sense of pride in your work. Write all of your programs correctly the first time and then you can reap the debugging rewards as you test them.