CSE 373 -- Autumn 1999 -- Programming Assignment Guidelines

Coding clearly: You should *always* make an effort to write code that is easy for other people to read. In the real world, this is an important skill that is valued higher than cleverness or using the least number of lines/characters possible. In this class, it is important so that we can understand your program and grade it fairly. By reading through your code and comments, we should be able to figure out how your code is organized and why it works. If we can't, we reserve the right to deduct points.

> *Commenting:* One aspect of writing clear code involves commenting your code so that it is clear what it does without reading it line by line (nothing is more boring). Comments should be used: (a) at the top of each file to tell what code it contains, (b) at the top of each function to tell what it does, (c) on any lines whose purpose is nonobvious.
>
> *Coding simply:* Although C and C++ allow it, there is no benefit to writing overly complicated statements such as:

```
if ((i = myfunc(counter++)) < num_iterations) {
   ...
}
```

> when they could easily be rewritten in a clearer manner:

```
i = myfunc(counter);
counter++;
if (i < num_iterations) {
  ...
}
```

> Coding should not be a macho activity. It should be like writing English sentences that you truly want someone to understand (including yourself).

Working Individually vs. with Partners: All of the programming assignments this quarter will be designed so that you can complete them individually. If you choose to work on a programming assignment individually, you may collaborate during the brainstorming stage as described on the course syllabus. Obviously, the actual coding must be done independently.

However, if it is your preference, you are welcome to work with a partner for any or all of the programming assignments. Ideally, I view this as a way for two people who aren't as confident with their programming skills to tackle the assignment together in order to deal with bugs in their code or problems with MSVC++. I've often found that two heads can often be better than one in these situations. My expectation is that both people will work on the program together rather than dividing the program into two halves and trying to sew the parts together the day before the project is due. For those of you who decide to work with partners, here is what I expect of you:

- Working with partners only applies to the programming assignments. Written assignments must be still be done according to the *Collaboration* section of the syllabus.

- Partners will turn in a single programming assignment that is the result of their combined efforts. Partners will generally receive the identical grade on the programming assignment.

- Each partner will also turn in a short, private statement of what role they took in the programming assignment and what role their partner took.

- Both partners are expected to understand the entirity of the code, and questions pertaining to the assignments may be assigned on the homeworks or on the midterm/exam to ensure this.

- You must work with different partners on each programming project. You may also choose to do some projects independently and others with partners. You simply may not work with the same partner twice.

- If you're working in a public lab such as MSCC, you must be sure not to disturb others around you as a result of working together.

I firmly believe that there is great value in programming in pairs -- it's amazing what two pairs of eyes can catch that one would've missed. The above rules are obviously set up to make sure that both people learn from the assignment and understand its point without one completely carrying the other. If you have any questions or problems with these policies, let me know or simply work by yourself.

Turn-in procedures:

For each programming assignment, there will be both an electronic turnin (so that we can compile and run your program) and a hardcopy turnin (so that we can scribble comments and grades on your code). Both of these will be due by the beginning of lecture or considered late. People working as partners should also turn in their private statements of who did what.

Electronic turnin: The electronic turnin will consist of mailing your code to c373@ms.washington.edu. Note that this is not the normal class mailing list, but a staff account on the MSCC UNIX machines. In your mail, you should:

- Have your name, UWID, and assignment number in the subject or close to the top of the mail message (as well as your partner's if you're working with one).

- Include all C/C++ code files and header files required to compile your project. Ideally, this should be done using attachments.

- Also attach any other files (such as sample input) necessary for correctly running your program. Do not send executables, or sample output.

Hardcopy turnin: The hardcopy should be a stack of paper that includes:

- A cover sheet with your name, UWID, and assignment number (as well as your partner's if you're working with one). The cover sheet should contain a brief (1 short paragraph) description of your assignment: what ADTs/algorithms you used, any bugs or features that are not working as well as it should (better to admit to them than have us discover them), any design decisions you made, or unique aspects of your code (especially anything above and beyond what was assigned). The cover sheet should conclude by giving a brief description of the code files contained in the hardcopy. For example:

main.cpp - our main C++ file for the program
list.cpp - our list ADT implementation
list.h - our list header file
hash.cpp - Weiss' hash table implementation with our changes
etc.

- After the cover sheet should come sample input and output demonstrating that your program works as it should. Be sure to include a few runs that test various aspects of your program rather than a single simple run that doesn't test much at all.

- Finally you should attach your actual code. You only need to supply code and header files that you created or modified. If you made only minor modifications to an existing file (such as the ADTs that Weiss provides), highlight the places where you made changes.

- For most assignments we will ask you to mark certain sections of your code with numbers. The idea here is just to help us find key areas of the code quickly during grading without reading the entire thing. So just use a big fat magic marker and scribble a big number with an arrow to point out those sections.