# CMSC 471: Information on Homework Grading

# Eric Eaton, Fall 2007

**(Adapted from Marie desJardins' Grading Policy)**

## From the Course Syllabus:

All homework solutions must be typeset. Any solutions requiring math or proofs must use proper mathematical notation. Similarly, graphs should be well-constructed with all axes labeled.

All solutions are required to be your own, individual work. You may, and are encouraged, to discuss methods, concepts, and assignments with anyone; however, the solutions turned in must be your own work. A good rule of thumb is to be alone when you sit down to actually generate solutions to the assigned problems, and to not show your solutions to anyone else.

At the top of your submission, you must include a clear statement specifying the source of any assistance you received on this assignment. This includes any websites you consulted, other students with whom you discussed any of the problems, etc. If you did not receive any assistance, you must say so.

## Grading of Written Assignments

Some of the written assignments will have a "right" answer. Those will be graded according to correctness, with partial credit given for incorrect solutions to the extent that you've shown your work and indicated why you believe your answer to be correct.

All answers should include a clear justification: a correct answer with no explanation of how the answer was obtained will only receive partial credit.

Other assignments will require you to express opinions in short answers or in essays. As with programming assignments, a portion of your grade will be given for the content of the essay, and a portion will be given for readability and style. As with programming assignments, the approximate distribution of credit will be as follows:

| | |
|---|---|
| 80% | Content. (i.e., well thought-out and well-reasoned answers; answers that are "correct" to the extent that there is a correct answer, which there often is not.) |
| 10% | Readability. (i.e., correct grammar and spelling, correct mathematical formatting, readable structure.) |
| 10% | Elegance. (i.e., well-expressed thoughts in a well-structured essay) |

Although this is not a writing class, success in any scientific discipline requires the ability to effectively communicate thoughts and ideas.

If you have difficulty writing, whether it's because English is not your first language, or because you haven't taken many writing classes in your undergraduate program, I highly suggest that you take advantage of the UMBC Writing Center, in the main library. This is a free tutoring service that will help you prepare essays and papers for any course.

UMBC Writing Center
Location: UMBC Library – Lower level
Hours: MTuWed 10:00 am - 7:00 pm; Th 10:00 am - 5:00 pm; F: 10:00 am - 2:00 pm
Phone: 410-455-3126

# Grading of Programming Assignments

Here is the approximate breakdown of how programming assignments will be graded. This breakdown may vary from assignment to assignment, but it gives you the rough idea:

| | |
|---|---|
| 80% | Correctness. (i.e., your program returns the right answer in all cases. Note that you must have complete error checking to receive this credit.) |
| 10% | Readability. (i.e., your program is commented with a header comment, documentation line, and/or inline comments as appropriate, and properly indented) |
| 10% | Elegance. (i.e., your program is simple, clean, efficient, and understandable) |

### Correctness

Your solution should be robust against errors. No matter what arguments are given to the program code, the program should not break. You may return an error code or default value, or use *error* or *cerror* to signal the error and enter a break level. The behavior of your program in handling an error should be clearly documented.

### Readability

For my code, I typically provide header comments, a documentation line, inline comments, and properly indent the code. This is perhaps more commenting than a simple program needs, but it's a good habit to get into anyway.

For larger programs, you may want to group functions together into categories and then include a header comment for each category. In these cases, if the functions are small, you may not need to include header comments for every function.

My rule of thumb for comments is that you should be able to throw out all of the code, leaving only the comments, and be able to regenerate the code from the comments. Similarly, my rule of thumb for header comments is that someone else who has never seen your code before should be able to pick it up and use it easily.

Please think carefully about how to organize and document your code so that it is readable. Developing software is about more than just getting the right answer: it's about writing code that is understandable, extensible, maintainable, and modifiable by yourself and by others.

### Elegance

Examples of inelegant solutions include: solutions that are inefficient, take many more lines of code than necessary, or use lots of temporary variables when just a few would do. Inefficient solutions are not as easy to understand or maintain as elegant solutions. Elegance does not just mean “short functions,” since making functions very short may also make them obscure and difficult to understand. Elegance in programming is an art; the only way you can learn it is to think carefully about how you formulate your solutions, and to study examples of well-designed code.