University of Massachusetts Dartmouth
Department of Electrical and Computer Engineering

ECE 160 – Grading Rubric

To give you an understanding of how your projects/labs will be graded, I distribute this rubric. Please read this, and occasionally re-read it. There are 100's of points lost on projects and labs because students do not pay any attention to various parts of this rubric. The last two pages contain specific instructions for how points are awarded and how points are taken away for most programs labs. Exceptions will be noted in the problem handout. I know there are some items in this list which may seem "trite" or otherwise overly "picky". As I mentioned in class, if everyone in the class did not name the submitted project correctly, it ADDS 4 HOURS to the grading time. Hence, the somewhat "picky" requirements

NOTES ON TESTING/VERIFICATION

Generally, all of the programs you are required to write for this course have a very tight specification. For most labs/projects, there are sample run(s) provided. To insure a good grade, your code should at a minimum match the sample run(s) as closely as possible. In addition to the sample test data, you should test several additional inputs. A successful engineer knows how to pick test data. The sample data that I supply is not an exhaustive list of possible inputs...When I grade your program, I will try several additional data sets...you should too.

RESUBMITS

Occasionally (rarely), I will indicate that you should fix a particular error and resubmit a program. Unless invited, a resubmit is not allowed. Generally, I only invite resubmits if it looks like there was an error introduced just prior to submitting the program.

WHAT GETS SUBMITTED AND WHERE?

You should only submit the files specified in the handout (generally these are .txt, .cpp and/or .h files, depending on the lab/project). All files should be submitted in the M:\ECE-160\yourname folder. You should NOT create any subfolders. Submitting extra files may result in loss of points. Submitting a project in a subfolder is considered naming it wrong (resulting in loss of points).

NOTE: By default, Windows hides file extensions for known file types. I suggest you disable this. On Windows 7, click on "Computer", then on the menu, click "Tools", and "Folder Options". Click on the "View" tab. Under "Advanced settings:", scroll down and uncheck the "Hide extensions for known file types" option. Click "OK". At this point extensions are shown for all files. NOTE: if clicking on "Computer" gives you a sub menu of drives, then right click on "Computer", and select "Open".

USING scanf()

All input shown on a single line in the sample runs must be done with a single scanf() statement. Input shown on multiple lines must be done with multiple scanf() statements. While sometimes (in the VS2013 environment), you can put two pieces of input on one line, and have it read by one scanf(), or vice-versa, this is not always the case. This will be checked when grading; failure to do this will result in loss of points.

ON THE PLUS SIDE:

Correct submittal name.....20%

Each project and lab has a specified name that must be used to submit the file. This is generally given on the assignment and on the syllabus in the form "`projname.cpp`", "`labname.txt`" or "`labname.cpp`". The name you use must match the given name exactly (pay attention to upper/lower case, etc). The quote marks are not considered part of the name.

Required Certification 5%

Each project and lab submitted should include one of the following statements.

A.
```
// I, <name> certify that this is my own work and have
// not collaborated with anyone else.
```
OR

B.
```
// I, <name> state that I collaborated with
// <name1>,...,<namen> on this project or lab,
// but still developed the details of my own code
```

Appropriate documentation (includes required certification; see above).....5%

At a minimum, the main function must have a header which includes your name, the date, and a description of the program. Each variable used in the main function must be described. Each and every function must have a header which includes the name of the function, a brief description, and a description of the “pre-conditions” (On Entry), and “post-conditions" (On Exit). Each variable used in a function must be described.

Appropriate looking code 10%

Appropriate looking code means code which (in my opinion) looks like it is related to the given problem.

Successful compilation 10%

The code compiles without any fatal errors or warnings which prevent the code from working.

Correct output.....50%

Output is as specified in the problem statement. Incorrect output results in lowering of score.

EXAMPLES:

- You write a bunch of code (with errors) which looks like you've been thinking about the problem and hand in a correctly named program with OK documentation, and the required certification...you would earn a score of 40 points.
- You write a bunch of code which looks like you've been thinking about the problem (and it compiles) and hand in a correctly named program with OK documentation, the required certification, and no output...you would earn a score of 50 points.
- You hand in a perfect program with the wrong name...75 points
- You hand in a perfectly running program with no documentation and no certification...90 points

ON THE MINUS SIDE:

Incorrect name for associated data file ..... -10 points

Some labs/projects will require your program to read and/or write a file. The name of the file will be specified in the handout. Your program must use this exact file name.

Submission after due date ..... -2^{n-1} points

Where n is the number of days late. i.e. a program that is one day late looses 1 point; a program which is 4 days late looses 8 points. A program is considered "submitted" when it is placed in your folder on the M: drive. Any program handed in more than 1 week late receives a grade of 3 (just as an indicator they were turned in). Unless otherwise specified, projects are due on date specified by 11:59:59pm. A "day" is defined as a 24 hour period beginning at midnight, and includes Saturdays, Sundays, and holidays. If the server (M: drive) is down, the deadline will be extended. Given the redundant nature of the network, this RARELY happens (maybe once every three or four years). Any down time is logged, and the system administrator sends an email to all ECE faculty.

Incorrect command structure ..... -10 to -20 points

In most projects there will be some type of command interface. This command interface is part of the specification, and must be followed to avoid loss of credit. Making the program more "user friendly" will not result in a higher score (it will likely lower your score). Following the specification will earn you the maximum number of points. Any "pausing" at the end of the code (a "pause" command or a "press any key to continue" followed by an input statement) will result in loss of 20 points.

Incorrect header files ..... -30 points

This is a course emphasizing C, not C++. Use of the file <stdafx.h> results in a lowering of score by 30 points.

Unaligned output..... up to -20 points

Several projects require that your output be in tabular (table) format, with right justified values, and or with decimal points aligned.

EXAMPLES:

- You hand in a perfect program...
  - 1 day late: 99 (2^{1-1}= - 1); 2 days late: 98 (2^{2-1}= -2)
  - 3 days late: 96 (2^{3-1}= - 4); 4 days late: 92 (2^{4-1}= -8)
  - 5 days late: 84 (2^{5-1}= -16); 6 days late: 68 (2^{6-1}= -32)
  - 7 days late: 36 (2^{7-1}= -64); 8 days late: 00 (2^{8-1}=-128)
- You hand in a program 3 days late, and the decimal points in your output do not line up as the problem specifies...86 (-4 late, -10 output)
- You hand in a program which runs perfectly, but uses `#include <stdafx.h>`...70 (-30 for using stdafx.h)
- You hand in a program which compiles, but several test cases work incorrectly...60-80 (-20 to -40 depending on how many test cases are incorrect)