Last Updated: January 16, 2018

Grading Policy

Course Information

Course Number	**CS 3377**
Course Title	**C/C++ Programming in a UNIX Environment**
Term	**Spring 2018**

Days & Times

Section 002	Tuesday & Thursday:	10:00am-11:15am	ECSS 2.306
Section 501	Tuesday & Thursday:	7:00pm-8:15pm	GR 3.302
Section 502	Tuesday & Thursday:	5:30pm-6:45pm	JSOM 1.217

Contact Information

Instructor	Dr. Stephen Perkins
Office Location	ECSS 4.702
Office Phone	(972) 883-3891
Email Address	stephen.perkins@utdallas.edu

Grading Policy

Projects and exams determine grades. The final grade will be composed as follows:

Programs	35%
Exams	60%
Attendance	5%

Letter grades will be assigned as follows:

97-100	A+
93-96	A
90-92	A-
87-89	B+
83-86	B
80-82	B-
77-79	C+
73-76	C
70-72	C-
67-69	D+
63-66	D
60-62	D-
Below 60	F.

All tests are closed book and closed notes. Laptop and electronic devices are NOT allowed.

There will be regularly assigned reading and homework problems. The homework problems will require the student to spend time programming a computer. Programming assignments should be turned in by means of eLearning. Assignment files contain:

A text copy of all source code
A text copy of any required supporting documentation or files
Specific details of deliverables are provided in each assignment write-up

All homework assignments will be graded by the TA. The instructor is responsible for grading the exams. **Therefore, if you have any question at all concerning the homework assignments, please speak with the TA about it first.** Even if you were to approach the instructor first, you would both still have to go back to the TA. It will save time to start with the TA first.

If you are dissatisfied with the result of your meetings with a TA, then please see the instructor about that issue. Together, you all can work to get it straightened out. You have every right to pursue any issue that concerns your grade in the course.

Homework Grading Criteria

All programming assignments and exams (other than the group project) are to be individual efforts. You are not to collaborate with other students. Copying of programming assignments or exams, in whole or in part, from other sources will be considered an act of scholastic dishonesty. This policy includes copying from other students, from assignments from previous semesters or from the Internet.

Programming assignments will be graded on a 100 point basis, utilizing the following criteria:

			Max Score
Source Code	Program Design	Partitioning	5%
		Organization	5%
		Efficiency	5%
		Coupling	5%
	Comments		10%
	Coding Style	Formatting	5%
		Naming	5%
		Capitalization	5%
Execution	Program Execution	No crashes	5%
		Error Recovery	5%
		Efficiency	5%
	Specification	Nominal cases	25%
		Special cases	5%
Documentation			10%
		Total	100%

Keep in mind that you always want to write code that is easy to understand and is easy to maintain.

Source Code: (45%)

Program Design: 20%

Many times you can write code that fulfills the requirements by just writing a single main() method. But most times, the problem is complicated enough to require several steps, which may be repeated one or more times. Please design your programs so that the functionality is spread across multiple methods that each accomplish a particular task, and name the method according to the task it performs.

Points	Criteria
5	Partitioning: Is the required functionality spread logically across multiple methods (or classes)?
5	Organization: Is the overall program flow easy to follow? Is it easy for an outsider to figure out how your software works?
5	Efficiency: Do the individual methods accomplish their given tasks as efficiently as possible? Are unnecessary variables, loops, methods, and classes eliminated?
5	Coupling: Are methods (and classes) "loosely coupled"? Does each method only receive the data it needs in order to accomplish its task? Are the "public" methods and variables appropriately so? Is information as hidden as possible?

Comments: 10%

- Every file should have a header that includes your name, course number, section number, & the homework number.
- Every class should have an extensive header comment explaining the purpose of the class.
- Every method should have comments explaining what it does, what its parameters are, and what values it returns.
- Significant variables and sections of code should have comments explaining their purpose. Avoid meaningless comments like "Declare the variables" or "This code adds one to the variable".

Coding Style: 15%

You should try to follow most of the coding standards provided in the textbook

Execution: (45%)

This section has to do with how well your program runs. If your program does not compile, please be aware that you may get no credit in this section.

Program Execution: 15%

Points	Criteria
5	No crashes: Does the program actually run all the way through the simplest possible test case without crashing?
5	Error Detection & Recovery: Does the program react well (does not crash) to unexpected or inconsistent events or input? Are exceptions handled appropriately?
5	Efficiency: Does the program finish executing in a reasonable amount of time?

Satisfaction of Specification: 30%

This is really the most important part: does your program do what it is supposed to do?

Points	Criteria
25	Nominal case: Does the software correctly fulfill the requirements of the assignment for the “expected” test cases?
5	Special cases: Does the software correctly handle unusual but legal test cases? Example: square root of a negative number, interest payment higher than loan payment

For programming assignments that seem arbitrarily restricted (“your program must save 4 runs”, “your program must accept up to 10 names”), points will be not taken off as long as your program meets at least those requirements. You may exceed them without penalty.

Documentation (10%)

All assignments must be submitted with documentation. At a minimum, this should include an algorithm report.

You should describe the overall flow of your program at least at a high level. If there are any parts of the program that are unusually complex, you should specify those parts in detail, using pseudocode or a flowchart.

Most of your documentation should be inside your code files. For particularly complex assignments, you may choose to include an external documentation file as well.