CMSC 162 (Spring 2015)
Introduction to Algorithmic Design I (4 credits)
http://marmorstein.org/~robert/Spring2015/cs162.html

Instructor: Robert Marmorstein **Office:** Ruffner 329
Office Phone: 434-395-2185 **E-mail:** marmorsteinrm@longwood.edu
Office Hours: 1:30pm-3:00pm MWF, 10:00am-11:00am T *or by appointment*

Lecture:

Section 1: 10:00am - 10:50am MWF (Ruffner 352)
Section 2: 12:00pm - 12:50am MWF (Ruffner 352)

Lab:

Section 1: 12:30pm – 1:45pm T (Ruffner G56)
Section 2: 2:00pm – 3:15pm T (Ruffner G56)

Course Description:
A continuation of CMSC 160. Topics include algorithmic design, complexity analysis, abstract data types, and encapsulation and basic data structures. Advanced topics using a modern high-level programming language inheritance, overloading, and use of objects.

Prerequisite:
Grade of C- or better in CMSC 160.

Course Objectives:

The student will be able to :

1. create programs using objects, classes, and modules.

2. apply and implement fundamental data structures, such as lists, stacks, queues, and hash tables.

3. analyze the time-complexity of algorithms.

Textbook and Other Resources:
The textbook for this class is "Classes and Data Structures" by Jeffrey S. Childs, Pearson, 2008, ISBN: 978-0-13-158051-0.

We will supplement readings from the textbook with readings from other sources. In particular, you will need to learn how to use the Unix Programmer's Manual (sometimes called the “man pages”) and the TexInfo documentation (accessible through the “info” command on any Linux system). These are an invaluable resource for any programmer that provide information about the standard programming libraries on the system and the UNIX programming environment we will be using to develop software. Both of these documents can be downloaded for free and installed on your computer. Alternatively, you can use the on-line versions which are linked from the course web site (though these may be out of date or intended for use with a slightly different environment than we will be using). Other readings will be posted to the course web site.

Course Requirements:
Your grade will depend largely on completion of the weekly lab sessions. These projects will comprise 50% of your grade. The remainder of your grade will come from homework assignments (20%), pop quizzes (10%), the midterm exam(s) (10%), and the final exam (10%).

Grading Policy:
Late work will not be accepted unless you have a serious medical or family emergency which prevents you from completing the assignment on time. In such cases, you do not need a doctor's note, but you must send me ***e-mail*** within a reasonable amount of time (typically twelve hours from the assignment due date) to explain your circumstances and to make arrangements for the work to be completed. All other late work will receive a grade of 0%.

Slip Days:
You will be allocated a fixed number of slip days at the start of the semester. You may use your slip days to extend the due date of one or more *programming projects*. You can use all of your slip days on one assignment or you may use them over multiple assignments.

Slip days are calculated from the minute the assignment is due until you turn it in and are rounded *up* to the nearest integer value. That means that if you turn an assignment in 24 hours and 1 minute after the due date, you will use up *two* slip days. The slip day clock runs over weekends and holidays. If a lab is due on Friday and you turn it in on Monday, you will have used three slip days, not one. Slip days cannot be shared, traded, bought, or sold, but can occasionally be earned by participation in relevant campus activities I select.

Grading Scale:

		100-91:	A	90:	A-
89:	B+	88-81:	B	80:	B-
79:	C+	78-71:	C	70:	C-
69:	D+	68-64:	D		
63 or lower: F		(There is no grade of D- in this course. Anything below a 64 is failing.)			

Attendance:
I expect you to attend class unless you are sick or engaged in a school-sponsored sport or extracurricular activity. Please do NOT come to class if you are sick. Instead, contact me within 12 hours of the absence to check whether you've missed any work and make arrangements to make up any missed quizzes. You should also make arrangements to get notes from another student in the class.

You should also check the course web site for announcements, new assignments, and other important updates.

I will rely primarily on your honor for enforcement of the attendance policy. However, I will keep a record of your attendance as required by Longwood policy. In accordance with that policy, missing more than 10% of scheduled class time (5 class sessions) to unexcused absences may, at my discretion, result in loss of one letter grade and missing 25% of class or more (14 sessions), whether excused or not may result in an automatic failing grade.

Food and Drink:
You may bring non-alcoholic beverages, including soft drinks, to class. However, please do not eat in class (it distracts me and the other students). Violations of this policy will be considered an unexcused absence.

I occasionally grant exceptions to this rule for students who must otherwise forgo lunch or have medical needs that require them to eat in class. If you feel that you need such an exception, you must make arrangements with me in advance (i.e. before bringing food to class).

Cell Phones and Laptops:
Cell phones, music players, and laptops are to be turned off and put away during class, except as needed for the lab sessions. Violations of this policy will be considered an unexcused absence.

Honor Code and Collaboration:
I am a strong believer in the honor code. As such, I encourage you to actively collaborate with other students and to discuss homework problems. However, there is a point at which collaboration becomes cheating and I deal harshly with cheating in my courses. To help you understand the line between acceptable discussion of a project and dishonorable behavior, I ask you to observe the following rules:

1. Exams and quizzes are to be completed entirely on your own.

2. For homework and projects, everything you turn in should be something YOU have personally typed or hand-written. You may NOT copy code electronically from other students or the Internet.

The work you submit should, in general, be your own original work or material which I have provided and you have suitably modified by yourself.

This doesn't mean you can't look online for help with a project. It just means that you must re-type any code you find and not just download it or copy/paste it. You may not share code with other students using flash drives, cell phones, e-mail, web sites, floppies, CDs, or any other electronic storage or communication device unless I specifically direct you to do so. You may not print out copies of your code to share with other students (personal copies or copies to show me are fine).

3. Do not copy large blocks of code from other students or the Internet.

You MAY assist other students or get assistance with simple problems like syntax errors, but you may NOT copy large blocks of code from each other. A good guideline of what "large" means is that copying one or two lines of code is usually okay, but copying more than three complete statements is usually too much.

4. You must give proper attribution.

Whenever you receive help or use an online resource, you should comment your code to give proper credit. A simple comment like:

/ based on http://codewarrior.com */*

or

/* Jessica helped me with the curly braces here */

is fine. This comment should go directly above or on the same line as the code on which you received help, so that it is clear exactly which parts of your program are original and which are not.

5. You are responsible for securing your code.

Helping other students to cheat is also cheating. Furthermore, it is your responsibility to make sure that other students do not use your work to cheat. Be careful with who you let access your computer and report any missing files, flash drives, or other devices to me promptly.

Infractions of these policies will be dealt with harshly under the Longwood Honor Code. Any student convicted of an honor offense involving this class will automatically receive a final course grade of **F** in addition to any penalties imposed by the Honor Board. You should consider all work in this class to be pledged work, whether or not the pledge appears on the assignment.

Computing Environment:
In order to complete the programming assignments, you will need to use a Unix-based open-source operating system such as Linux or BSD. ***You are responsible for getting a development environment set up and working correctly on your system.***

To do this, you have two options. One option is to install Linux directly onto your hard drive and dual-boot.

Another option is to use a Live CD or Live USB disk to run Linux without any modifications to your hard drive. **I highly recommend that you install Linux directly to a partition of your hard drive or use a Live USB disk.** The Live CD option requires the use of a flash drive to save your work and can introduce permission problems that make compiling and running the projects more difficult. **I do not recommend the use of virtual machines to run Linux for this class.**

If you have a Macintosh, you have an additional option. Your operating system already provides many Unix tools through the terminal utility. Most of the projects in this class can be completed directly from the Mac Terminal. To do that, you will need to install the XCode developer tools, which are available free from Apple. Be aware that you may also need to adapt the instructions of some of the programming labs to account for differences in the programming environment. If you elect to choose this option, it is your responsibility to get everything set up properly and make things work.

Tentative Course Schedule:

Jan. 13	Lab 0: Writing C++ Programs in Linux using Vim
Jan. 14-16	Introduction, UNIX review, C++ Review, Design: Code Reuse and Code Modularity Types, Loops, Functions, Blocks and Scope, **Read Syllabus**
Jan. 19	**Holiday: NO CLASS**
Jan. 20	Lab 1: Unix Review **Last day of Add/Drop (by 5 pm)**
Jan. 21-23	Structs and Classes, Encapsulation and Abstraction, Methods and Operators, Accessors and Mutators Design: Top-Down Design, Implementation and Interface Version Control **Read Chapter 1**
Jan. 26-30	Linking Libraries, Makefiles, Testing, In-line Functions Templates, Vectors, Lists, and Maps **Read Chapter 2**
Jan. 27	Lab: Catchup and Review
Feb. 2-6	Constants and Mutability, Pointers and References, Dynamic Memory, Constructors and Destructors, Initializers Dynamic Arrays **Read Chapters 3, 4, and 5**
Feb. 3	Lab 2 : Dynamic Arrays

Feb. 9-13	Inheritance, Composition, and Polymorphism, Casting, Static Members, Using the Debugger, Virtual functions and classes **Read Chapter 6**
Feb. 10	Lab: Catchup and Review
Feb. 16-20	**Catchup and Review, Midterm Exam**
Feb. 17	Lab 3: Debugging and Inheritance
Feb. 23-27	Linked Lists, Array-based Lists, Pointer-based Lists, Using Valgrind **Read Chapter 7**
Mar. 2-6	**Holiday: NO CLASS**
Mar. 9-13	Stacks and Queues, Time Complexity, Using profilers **Read Chapters 8 and 9**
Mar. 10	Lab 4: Linked Lists, Stacks and Queues
Mar. 16-20	Abstract Data Types, Iterators, Hash Functions, and Hash Tables, Using static analysis tools **Read Chapters 10 and 11**
Mar. 17	Lab: Catchup and Review
Mar. 23-27	Priority Queues, Trees, and Heaps, Tree Traversals **Read Chapter 12**
Mar. 24	Lab 5: Heaps
Mar. 30-Apr. 3	Recursion, Searching, Binary Search **Read Chapter 13**
Mar. 31	Lab: Catchup and Review
Apr. 6-10	Sorting (Selection Sort, Insertion Sort, Merge Sort, Heap Sort) **Read Chapter 14**
Apr. 7	Lab 6: Sorting
Apr. 13-17	Binary Search Trees, Application of Balanced Binary Search Trees, Graphs **Read Chapter 15**
Apr. 14	Lab 7: Binary Search Trees
Apr. 20-24	Catchup and Review
Apr. 30	**Section 1 Final Exam** (3:00pm-5:30pm, Thursday) **Section 2 Final Exam** (11:30am-2:00pm, Friday)