CMSC 162
Introduction to Algorithmic Design II
Spring 2020

http://marmorstein.org/~robert/Spring2020/cs162.html

Lecture: 1:00 – 1:50pm MWF (Ruffner 350) **Lab:** 12:30 – 1:45pm R (Ruffner G56)

Instructor: Robert Marmorstein (marmorsteinrm@longwood.edu) **Phone:** (434)395-2185
Office Hours: 2:00-4:00pm MWF, 1:00-2:00pm T or by appointment **Office:** Ruffner 329

I am also available by appointment. My schedule is posted near my office door. To make an appointment, please check the schedule to see which times I am free, then contact me by e-mail and list some possible times we could meet. In general, I need at least 24 hours of notice to schedule an appointment.

Communications Policy
The best ways to get in touch with me outside of office hours are either to use Slack or to send e-mail to marmorsteinrm@longwood.edu. Typically, I will reply within 24 hours (often sooner) on weekdays. I often reply much quicker – even on weekends.

If you are asking for help with a project or homework problem by e-mail, you should attach your code or your work to the e-mail or copy/paste the part you are working on into the body of the e-mail. **Do NOT attach screenshots or pictures taken on your phone.** They are hard to read and take up too much space in my inbox. In general, e-mails containing images will be deleted unread.

An even better way to get in touch with me is to use **Slack**. Slack is a chat utility with clients for mobile devices and desktop computers. It will allow you to easily send me code snippets. Also, since I get notifications when a slack message comes in, I am more likely to reply to your message quickly if you use Slack than if you send me e-mail.

Slack is also a good way to communicate with other members of the class. Feel free to ask for help on the course Slack – as long as you stick to general questions about topics and do not share large blocks of code.

Course Description:
A continuation of CMSC 160. Topics include algorithmic design, complexity analysis, abstract data types, encapsulation and basic data structures. Advanced topics include using a modern high-level programming language, inheritance, overloading, and use of objects. 4 credits.

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

Student Learning Outcomes:
At the end of this course, the successful student will be able to:

– apply object-based principles to creating understandable and maintainable solutions to problems

– identify appropriate implementations for abstract data types such as stacks, queues, lists, sets, trees, and maps

– explain, implement, and use data structures such as linked lists, trees, and hash tables

– compare and contrast standard algorithms using complexity analysis

Course Structure and Student Expectations
This course is heavily project-driven, but also has a heavy theoretical component. In general, we will spend three hours a week in lecture and discussion sections and one hour a week in class working on laboratory projects. However, you should expect to spend at least nine additional hours each week reading the textbook, preparing for exams and working on assignments.

Textbook and Other Resources:
The textbook for this class is “C++ for Lazy Programmers”, 1st Edition, by Will Briggs, Apress, ISBN: 978-1-4842-5186-7. We will supplement readings from the textbook with readings from other sources such as the Unix Programmer's Manual (sometimes called the “man pages”) and the TexInfo documentation (the “info” command). 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 and quizzes(**25%**), participation(**5%**), the midterm exam(**10%**), and the final exam(**10%**).

University Policies:
This course adheres to the university policies found at http://www.longwood.edu/academicaffairs/syllabus-statements/.

Grading Policy:

Your final grade in this course is computed using a weighted average of your scores on each assignment. The weights for each category are given in the course requirements section of this syllabus and can be used by applying the following formula:

Final Grade = 0.50*Projects + 0.25*(Homework and Quizzes) + 0.05*Participation + 0.10*Midterm + 0.10*Final

Each of the category grades (such as Projects) can be computed by summing the points you’ve earned on each assignment in that category and dividing by the total number of points possible. Numeric grades are translated to letter grades using the following 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)			

Late Work:
In general, I do not accept late work or grant extensions on assignments unless you have a serious medical or family emergency which prevents you from completing the assignment on time (however, see “Slip days” below). In exceptional circumstances, I may be persuaded to grant extensions on one or more projects or assignments. In such cases, you do not need a doctor's note, but you must notify me of the circumstances within a reasonable amount of time (typically within twelve hours of the deadline).

All requests for extensions MUST be submitted by e-mail within a reasonable amount of time. This e-mail should outline in detail the reasons your work is late. Granting of extensions is entirely at my discretion – if you have not turned an assignment in on time, you should expect to earn a zero.

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. The number of slip days used is 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.

Slip days may not be used on homework assignments, quizzes, or exams.

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 twelve 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. In accordance with Longwood 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.

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. I will not interrupt class to notify you if you have been counted absent for use of a prohibited device. Feel free to contact me by e-mail at any point in the semester to check on the number of absences you have in my class.

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 will not interrupt class to notify you if you have been counted absent for violation of this policy. Feel free to contact me by e-mail at any point in the semester to check on the number of absences you have in my class.

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).

Honor Code and Collaboration:
I firmly believe 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.

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. You may not discuss them with anyone or use any resources except those specifically outlined on the exam handout.

2. You must give proper attribution.

Whenever you receive help or use an online resource, you should comment your code to give proper credit. The best way to do this is to place a comment ***above or on the same line*** *as the code on which you received help or used a resource. For example:*

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

or

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

is fine. You **DO NOT** need to cite material you obtain directly from me (in lecture, the assignment handout, or office hours). In general, you also **DO NOT** need to cite material taken from the textbook.

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

You **MAY** use web sites, books, and the man pages as reference materials. However, you must cite them appropriately and you **MUST** re-type any code you find and not just download it or copy/paste it.

At no point should another student touch your keyboard while helping you with a project. ***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.***

You **MAY NOT** share code with other students using flash drives, cell phones, e-mail, web sites, floppies, CDs, or other means 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 during office hours are fine).

4. 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, such as entire classes or functions, from a web site or from another student. How much code is “too much” depends partly on context, but a good guideline of what "large" means is that copying more than three complete programming statements is usually too much.

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 account 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.

If you have questions about the honor code policy, PLEASE ask me. It is much better to receive a late penalty on a single assignment than to fail the course and face honor board charges.

You may find the scenarios at https://integrity.mit.edu/handbook/writing-code helpful in understanding this policy. While their honor code policy is not identical to mine it is similar.

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.*** There are two options for doing this:

A. You can install Linux directly onto your hard drive in a "dual-boot" configuration. This means that you can have both Linux and another operating system on your computer and can reboot to switch between them.

B. You can install Linux in a "Live" configuration, where it runs off a DVD or external drive (such as a USB flash drive). You will then need a way to save your work to another device (such as a second flash drive or in the cloud).

C. You can install Linux in a virtual machine – a program which “simulates” a computer and allows you to run another operating system within your own environment. Virtual Machines run a bit slower than physical ones and you may experience some difficulties with permissions and transferring files. However, if you are unable to set up a dual-boot environment this can be a good way to go.

Any of these options can work in this course, though students who successfully configure their systems to dual-boot usually experience fewer problems in the long run.

Other Software
In addition to the Linux operating system, you should have (at a minimum) the following software tools installed: the vim editor, the “gcc” and “g++” compilers, a web browser (such as chromium or brave), and the SDL development libraries. On Debian-based distributions such as Ubuntu, you can use the following command to install the software for this course:

```
sudo apt install build-essential vim libsdl2-dev splint strace
```

On Redhat-based distributions such as Fedora and CentOS, you can use:
```
sudo yum install g++ gdb valgrind strace splint vim SDL2-devel
```

What if I have a Mac?
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.

You may also have to install either Homebrew (“brew”) or MacPorts (“port”) or both in order to get SDL and other tools working.

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. 15 – 17	Introduction, UNIX review Introduction to SDL Review of Arrays and Vectors Review of Structs **Read Syllabus** **Read Chapters 10 and 11**
Jan. 16	**Lab 0: Writing C++ Programs in Linux using Vim**
Jan. 22 – 24	Files and Streams String Streams Libraries, Makefiles, and Linking Testing **Read Chapters 13 and 24**
Jan. 23	**Lab 1: C++ Review** **Lab 0 Due**
Jan. 22	**Last day of Add/Drop (by 5 pm)**
Jan. 27 – 31	Pointers, Lvalues, Rvalues, and References Dynamic Memory and Dynamic Arrays Testing and Debugging Smart Pointers **Read Chapter 14**
Jan. 30	**Lab 2: Dynamic Memory** **Lab 1 Due**
Feb. 3 – 7	Classes, Constructors and Destructors, Initializer Lists Design Principle: Top-Down Design Mutators and Accessors Encapsulation (public and private variables) Implementation and Interface Constants and Mutability In-line Functions Static Members Documentation **Read Chapters 15 and 16**
Feb. 6	**Lab Day: Continue Lab 2**
Feb. 10 – 14	Operators and Exceptions Design Principle: Code Reuse and Modularity Abstraction and Polymorphism

	Operator Overloading Move constructors and move assignment The swap and move functions **Read Chapter 17**
Feb. 13	**Lab 3: Object-Oriented Programming and Inheritance** **Lab 2 Due**
Feb. 17 – 21	Algorithmic Analysis, Big-O Searching: Linear and Binary Search Version Control **Read Chapter 18**
Feb. 20	**Lab Day**
Feb. 24 – 28	Inheritance, Composition, and Polymorphism Virtual and Pure Virtual Functions Casting Objects Abstraction and Polymorphism Virtual functions and classes **Read Chapter 19**
Feb. 27	**Lab 4: Linked Lists, Stacks and Queues** **Lab 3 Due**
Mar. 2 – 6	**Spring Break: NO CLASS**
Mar. 9 – 13	Catchup and Review, **Midterm Exam**
Mar. 12	**Lab Day**
Mar. 16 – 20	Linked Lists, Array and Pointer based Lists, Abstract Data Types, Iterators Stacks and Queues **Read Chapter 22**
Mar. 19	**Lab 5: Stacks, Graphics, and Exceptions** **Lab 4 Due**
Mar. 23 – 27	Trees, Tree Traversals, Recursion Balanced Binary Search Trees
Mar. 26	**Lab Day**
Mar. 30 – Apr. 3	Priority Queues (Heaps) Templates, Vectors, Lists, and Maps
Apr. 2	**Lab Day** **Deadline to withdraw without an ‘F’**

Apr. 6 – 10	Hash Functions, and Hash Tables, Open and Closed Hashing Using static analysis tools
Apr. 9	**Lab 6: Hashing and using an IDE (integrated development environment)** **Lab 5 Due**
Apr. 13 – 17	Elementary Sorting: Selection Sort, Bubble Sort Insertion Sort
Apr. 16	**Lab 7: Review** **Lab 6 Due**
Apr. 20 – 24	Advanced Sorting: Quick Sort, Merge Sort, and Heap Sort
Apr. 23	**Research Day: NO CLASS**
Apr. 27 – 29	Exam Review **Read Chapters 8 and 9**
Apr. 29	**Lab 7 Due**
May 4 (Monday)	**Final Exam** (3:00 – 5:30pm)

Major Assignments:

In addition to the two exams, there will be several laboratory projects in this course.

Exams:

The midterm exam will be taken in class on March 13th and is worth 10% of your final grade. The final exam will be held on Monday, May 4th at 3:00pm and is also worth 10% of your grade.

Projects:

Projects are worth 50% of your grade. There will be six to eight laboratory projects. For tentative due dates, see the course schedule above.