syllabus for
data structures
comp 116

Instructor: **Mark LeBlanc**
mleblanc@wheatoncollege.edu
http://cs.wheatoncollege.edu/mleblanc

SC-1322 508.286.3970
Hours: **MON 10:30-11:30**
TUE and WED 1-2pm
or appointment

Meeting Times: **MWF** 9:30-10:20 Mars 1349(lecture)
Tue Lab 3:30-5:20 SC 1315

How to think like a computer scientist (C++ Version). Allen B. Downey (2012).
C++ Programming. Wikibooks.org
Data Structures Using C++ (2nd Edition). Malik, D.S (2010).
I also strongly recommend that you buy a 3-ring binder. I'll pass out lots of handouts.

Content: This course is the second half of a year-long introduction to computer science for majors and minors. The content includes an introduction to the theoretical and practical aspects of data structures. Emphasis is on abstract data types and the use of the class mechanism to support object-oriented implementations, including exposure to the C++ Standard Template Library (STL). Examples include stacks, queues, linked lists, and general trees and their applications. Pointers, memory management, and recursion are discussed in length and used in some implementations.

Content revisited: *Ok, now I'll say that in a less geeky way.* In your first programming course, probably using Python, you were exposed to programming and introductory ways to solve computational problems by writing software. This course is more of the same, although a few items will change. First, the programming language here will be C++, a language that shifts more of the burden of correctness to you, the programmer. That's right, there is no sugar coating things if we choose to use a systems-programming language like C++. When programming in C++, you must be explicit about the types of data that you will store in memory. You have near-complete access to the memory used by your program, down to the bits if you want (although we won't go there in this course) and failure to access that memory in an appropriate manner will cause your program to crash. Since C++ is often used in mission-critical applications, 'code that crashes' is not an option. And so, with much power comes much responsibility. The second change from the first course will be our study and application of more complex ways to "structure your data", a.k.a. "data structures". We will use the C++ object-oriented features to both implement our own data structures as well as apply higher-level objects made available in libraries. Third, we will repeatedly emphasize good software engineering skills, including the steps that lead to verify "program correctness." Specifically, you will learn how to write good (internal) documentation in your programs, including pre- and post-conditions for your functions and methods and logical invariants for your loops. In summary, this second course is where you really appreciate and practice the art of writing software, of programming, of scripting. Writing software is a craft and given the evasive nature of software in our world, programmers have an obligation to be efficient *and* correct. In short, I often tell students that once they complete this second course, they'll "really (start to) get it." Are you ready to "get it?"

In computer science, if you are almost correct you are a liability.
Fred Kollett (1941-1997), MathCS, Wheaton College, Norton, MA

Your Grade:

Things to do	Grading	Frequency
Labs	10%	weekly
6 Programs	45%	TBA
Quizzes (approximately 5)	10%	TBA
Exams (2)	20%	TBA
Final Exam	15%	Thurs, Dec 14, 9:00

onCourse (moodle):
All daily deadlines and readings will be **written on the board in class** and listed on our class online website in onCourse. I have *not* attempted to list a full 15-week schedule on this paper syllabus.

Late Submissions:
Due is due. Always turn in whatever you have on time. Something turned in on time is much better than not having it accepted because it is late. Late is not an option. (*Good, glad we can all agree with this*). Note that the moodle (onCourse) page associated with this course will have submission areas that are time triggered; if you are late, submissions will be blocked. Note: If a programming assignment is "due" on, say a Monday, I will actually allow you to submit your programs up to 4am of the following day. Thus, a program due on a Monday can safely be submitted up until Tuesday 4am. In short, if you are willing and/or need to work on and test your solution late into the night that it is due, you are granted some grace time. Note: See the paragraph above: *due is due*.

Honor Code Revisited:
It goes without saying that all submitted work will be the student's own, in keeping with the Wheaton Honor Code, unless the assignment has assigned groups. For labs, you may get "help" from fellow classmates, but remember that all completed work must be your own. Use discretion; don't ask your colleague for "the" answer or for lines of code. However, I do encourage you to discuss the problem in general, such as the type of statements or functions one might use. For programming assignments, your answers and software must be your own from beginning to end. Here is an analogy. Almost no one would every "use/steal" a line or two from another person's poem. Consider it the same with your programs. Don't "borrow/use" lines or sections of code from another classmate. Your program is (like) your poem; everyone's program should be unique. Be wise. If a colleague is asking you for too much help, be honest and remind them your program is just that, *your* program.

Homework: It is expected that you spend at least 2^{++} hours on reading and practice problems for every 50 minutes of lecture. This computes to at least 6 hours of work in the Data Structures texts per week. This should be done throughout the semester and not just when studying for quizzes or exams. The material is cumulative in a big way; for example, week 5 depends heavily on weeks 1 through 4. It is expected that you spend at least 6 hours per week on your current programming assignment. WARNING: Programmers typically underestimate the time it takes to complete a software project; 6 hours per week on your programming assignment may be one of those "underestimations."

Labs: The labs are a critical part of the course. In almost all cases, the current lab will be preparing you for the current programming assignment. That is, if you complete and understand the lab, you should be well on your way to a solution for the programming assignment. In order to best grasp the material in some labs, I strongly suggest that you completely redo any labs that you find difficult. (Read that last sentence again, unless of course you've already reread it once).

Quizzes: In addition to two exams spaced evenly over the semester, you will be quizzed regularly (the dates TBA). Each quiz may address material that has been assigned in the text, but not yet presented by the instructor. NOTE: you'll have to read the material and practice the sample problems BEFORE coming to class in order to be successful with the quizzes. Quizzes start at the beginning of a lecture and last approximately 10-15 minutes. After some quizzes, immediate (peer) grading will be followed by a discussion which will focus on the concepts in the quiz.

Quizzes and Exams: There will be no makeups, nor will the lowest grade be dropped. If you are an athlete and/or you have a conflict with a quiz date, please see me. When studying for exams and quizzes, have someone pick problems from the notes and text, mix them up, and present them to you as if you were taking a quiz (not telling you from what section a problem comes). While practicing, do not peek at the textbook or an earlier solution. NOTE: its easy to "read" code; its much more difficulty to "write" code from scratch. Quizzes and exams will sometimes ask you to write code from scratch; practice ownership of the concepts!

HELP

I have listed my office hours on the syllabus, but be bold: schedule a time to meet. Study, study, study and talk about the material with me as often as you can.

Please don't wait too long before you see me;
a quick chat in my office can often clear things up.
I'm here a lot…

Accommodations for Disabilities

Wheaton is committed to ensuring equitable access to programs and services and to prohibit discrimination in the recruitment, admission, and education of students with disabilities. Individuals with disabilities requiring accommodations or information on accessibility should contact Abigail Cohen, Assistant Dean for Accessibility and Assistive Technology at the Filene Center for Academic Advising and Career Services. ~ cohen_abigail@wheatoncollege.edu or (508) 286-8215 ~

Wondering what you'll be doing with a degree in computer science?
*Check out this website (**computingcareers.acm.org**) for the exciting potential of a career in computing ….*