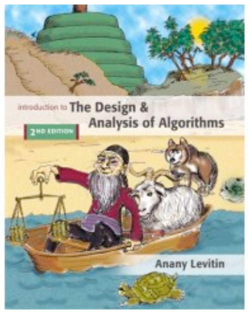
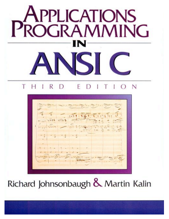
Syllabus for **Algorithms** COMP 215

Instructor:	Mark LeBlanc (mleblanc)	Office Hours: by appt. *or* TR 2-3:30; W 2:30-3:30 or by appt.
Office:	SC-B103	Meeting: TR 9:30-10:50am (A118)
Phone:	286-3970 (on campus: x3970)	W 3:30-5:20pm (A102)

Required Text:
The Design & Analysis of Algorithms
by Anany V. Levitin. Addison Wesley;
2nd edition (2007).



Strongly Recommended:
Applications Programming in ANSI C
by Richard Johnsonbaugh & Martin Kalin.
Prentice Hall; 3rd edition (1996).


APPLICATIONS PROGRAMMING IN ANSI C
THIRD EDITION
Richard Johnsonbaugh & Martin Kalin

A 3-ringed binder:
I will pass out many pages of notes. <u>Many</u> …

" ... *then (the algorithms of today) will become 'simple' problems and a new generation of challenges, which we can now only barely imagine, will take their place on the frontier of what it is possible to do with computers*."
Aho and Ullman, Foundations of Computer Science, p95.

Content of the course:
An introduction to the mathematical foundations, design, implementation and computational analysis of fundamental algorithms. Problems include heuristic searching, sorting, several graph theory problems, string matching, and the theoretical expression of their orders of growth. Out-of-class assignments and hands-on labs emphasize the balance between theoretical hypotheses (that is you figure out on a napkin how long an algorithm will take to run) and experimental verification (if your napkin math says the algorithm will finish in a reasonable time, then you implement a program to actually run an experiment).

Curriculum:
This course is one of four core courses that are required for a computer science major or minor at Wheaton. In many ways, Algorithms is the first course that challenges students with the subtle rigor of the discipline. It seems that everyone loves computing and the associated gadgets today, but the requirements for reliable and efficient software go far beyond the glitz of the web or your cell. The sections in an Algorithms text, for example: Big Oh notation, growth rates, combinatorial search, heuristic methods and intractable problems are themes that emerge throughout most of the sub-disciplines of computer science.

As always ...

"In computer science, if you are almost correct you are a liability."
Fred Kollett (1941-1997), Professor of Math and CS, Wheaton College

Your grade:	(Takehome) Quizzes	**20%**	continual throughout the semester as assigned in lectures
	Weekly Labs	**5%**	continual throughout the semester
	Programs	**45%**	continual throughout the semester as assigned in lectures
	2 In-Class Exams	**20%**	TBD
	1 Final Exam	**10%**	Thu, Dec 16, 2:00 – 5:00pm

Honor Code Revisited:

It goes without saying that all submitted work will be the student's own, in keeping with the Wheaton Honor Code. Use discretion; don't ask your colleague for "the" answer on a takehome quiz question or programming assignment (and raise an eyebrow if someone asks *you*). However, I do encourage you to discuss problems in general, such as the type of data structure one might use or to help with a syntax error. Labs are open for discussing solutions, even getting hints. Your on-paper quizzes, exams, and your programming solutions must be your own from beginning to end.

Here is an analogy. Almost no one would ever "use/steal" a line or two from another person's **poem**. Consider it the same with your programs. Do *not* "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.

LATE SUBMISSIONS:

Programming Assignments -- Due is due. Always turn in whatever you have on time. Something turned in on time is much better than receiving no credit because it is late. If the program is "due" by Tuesday, 5am, I will turn OFF the onCourse (moodle) dropbin, that is, you will not be allowed to submit your source code after that point. If your code does not compile/run/work right, submit what you have and tell me in a README file. So let us agree together: all assignments are due as stated. (Exceptions granted for serious reasons). If your professor is your boss (and he is of sorts) and your boss wants it when he said he wants it (and he does), then you submit what you have and explain what it does and does not do in your README file. **Late is not an option**. (*Good, glad we can all agree with this*).

Note: If a programming assignment is "due" on, say a Monday, I will actually allow you to submit your programs up to 5am of the following day. Thus, a program due on a Monday can safely be submitted up until Tuesday 5am. 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*.

Submitting code: All programs submitted will include a **README.txt** file that explains the status of your work, e.g., "all is working", "everything but X works", etc. All programs will be submitted via our onCourse site. When submitting your coding solutions, you *must* store all your files (.c, .cpp, .h, README.txt, sample output if appropriate) in one folder and **.zip** that folder into one (1) file. You then will submit only one file (a .zipped file containing a folder with all your source, header, input, and README files). Unless otherwise noted, I will compile code on a Window's box (DevC++ uses the gcc compiler), thus *you* must make sure your solution compiles on Windows *before* you submit it.

All source code *must* be printed ***in landscape* mode** and **STAPLED** and submitted to me by the day after your code is electronically submitted. I will deduct points for source code that is not stapled.

Pencil-paper take-home quizzes – Any take-home quiz that is due on a particular day, e.g., Tuesday, will be collected *at the start of lecture*. Work submitted at the end of class or later in that day will lose points. Again, it is better to submit unfinished work rather than nothing. Leave me a note, e.g., "I got *this* far on #3, but I couldn't get the final solution."

Your handwritten work ***must* be neat**. I recommend that you work on your solutions on scratch paper throughout the week, not trying originally to be neat. Then, when it is time to submit your work, you can transcribe your solutions to new pieces of paper. Note: use *lots* of space on your paper; that is, use lots of pages of paper so your solutions appear neat and professional looking. This implies that you show *how* you arrived at a solution, that is, **show *all* your work**, **not just "the answer."** And of course, **STAPLE** your pages. I will deduct points for work that is not stapled and not neat.

OUT-OF-CLASS WORKLOAD:

It is expected that you spend at least 2-3 hours on reading and practice problems for every 50 minutes of lecture. This computes to at least 6-9 hours of work in the text per week. This should be done throughout the semester and not just when studying for exams. The material is cumulative in a big way; for example, week 5 depends heavily on weeks 1 through 4.

It is also expected that you spend at least 6-10 hours per week on your current programming assignment. WARNING: Programmers typically underestimate the time it takes to complete a software project; 6-10 hours per week on your programming assignment may be one of those "underestimations." This is *not* an introductory computer science course. No whining!

QUIZZES

There will be 7-10 quizzes throughout the semester, many of them take-home quizzes. Quizzes may be based on pages of reading that were assigned in the previous lecture. You may be quizzed on material in class that has not yet appeared in lecture, that is, **you *must* read *before* the material is presented in class**. Quizzes may be used to get a pulse of the class, i.e., to see if you get the gist of the material in the reading.

EXAMS

There are two in-class exams during the semester and a final exam during finals week. There will be no makeup exams, nor will the lowest exam grade be dropped. If you are an athlete and/or you have a conflict with an exam date, please see me within the first week of classes.

HELP

I have listed my office hours on the syllabus. But you know I'm always willing to meet you! Don't let this material bury you. Study, study, study and talk about it with me as often as you can. Homework and programming assignments can be especially challenging. Visit me; ask questions!

Important: You must use your office hour visits judiciously. While I may have allowed it in the past, in this 200-level course, you may *not* just "hang out" in my office and work on your programming assignments. In short, you must come to my office prepared with your question(s).
But you know … I *do (still) love ya* …