Syllabus
CMSC 162: Intro to algorithmic design II

Fall 2014

Lecture:	MWF 10am, Ruffner 352
Lab:	Thu 2pm, Ruffner G56
Website:	`http://cs.longwood.edu/courses/cmsc162`

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 such as inheritance, overloading, and use of objects. Prerequisite: Grade of C- or better in CMSC 160. 4 credits.

Professor:	Don Blaheta
Office:	Ruffner 337
Phone:	x2191
Email:	`blahetadp@blahedo.org` [1]
Office hours:	Mondays 4–5:30pm; Tuesdays 2–3:30pm; Fridays 11–noon; and by appointment

Overview

You have by now acquired some basic skills of programming and analysis, but the programs you've written have (of necessity) been small and the data uncomplicated. In this course you will continue to develop your programming skills, but more importantly, you will learn how to build layers of abstraction (and use abstractions that others have built) that will enable you to write and understand larger and more interesting programs and processes.

[1] Or `blahetadp@longwood.edu` if you'd rather, but I prefer the off-campus one and check it more frequently.

Textbook and resources

The required book for this class is *Data structures and problem solving with C++*, 6e, by Frank M. Carrano and Timothy Henry (ISBN 978-0-13-292372-9). You may share it, but there will be daily assigned reading, so plan accordingly.

The other main resource is provided by us: you'll be given an account on the department Linux machines (if you don't already have one), and you'll do your programming work there.

Course objectives

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

- explain, implement, and use data structures such as linked lists, trees, and hash tables;
- analyse and identify appropriate implementations for abstract data types such as stacks, lists, sets, and maps;
- perform basic complexity analysis on standard and novel algorithms; and
- employ the object-oriented paradigm to write understandable and maintainable programs.

Grading scale

I tend to grade hard on individual assignments, but compensate for this in the final grades. The grading scale will be approximately as follows:

A−	[85, 90)	A	[90, 100)	[2]	
B−	[70, 75)	B	[75, 80)	B+	[80, 85)
C−	[55, 60)	C	[60, 65)	C+	[65, 70)
D−	[40, 45)	D	[45, 50)	D+	[50, 55)

While there will be no "curve" in the statistical sense, I may slightly adjust the scale at the end of the term if it turns out some of the assignments were too difficult.

[2] Alas, no A+, unfortunately.

Content

Calendar

Wk	M	W	R	F
	August			
1	**25** — Introductions Policies	**27** §§1.0–1.3 Design and specification	**28** — Lab 1: File I/O, 2D arrays	**29** §§1.4–1.5 ADTs Bags
	September			
2	[Labor Day **no class**]	* **3** §3.1 Classes and methods .h files	**4** Ch. CI1 Lab 2: Classes	**5** §§3.2–3.2.2 Implementing an ADT
3	**8** §3.2.3 & TBA Testing	**10** §§3.2.4–3.2.6 More implementation	**11** — Lab 3: Class impl Unit testing	**12** §§2.1–2.2 Recursion
4	**15** §§2.3–2.4, 3.3 Recursion cont'd Binary search	**17** Ch. CI2 Pointers	**18** — Lab 4: Pointers	**19** §2.4.2 Binary search revisited
5	**22** §§2.6, 4.1 Classic recursion: Fibonacci Linked nodes	**24** §§4.2–4.3 Linked impl of Bag	**25** — Lab 5: Linked node methods	**26** §§4.4–4.5 Comparing impl's of an ADT
		October		
6	**29** §§5.1–5.2 Recursive algorithms	**1** §5.3 Recursive backtracking	**2** — Lab 6: Reading code `make`, `gdb` Backtracking	**3** §§2.5, 5.4 Towers of Hanoi Recursion / induction
7	**6** §§CI3, 7.3 Exceptions **Exam 1 TH out**	**8** **Exam 1**	**9** §6.1 Lab 7: Using STL `stack`	**10** §§6.2, 6.4–6.5 Uses of stacks Stacks and recursion

* **2 September**: Deadline to add/drop classes (5pm)

Wk	M	W	R	F
	October			
8	[Fall Break **no class**]	**15** * — Classic ADTs The "big picture"	**16** — Lab 8: Empirical efficiency	**17** Ch. 10 Algorithmic efficiency Big-O notation
9	**20** Ch. 8, §9.1 List ADT Array lists	**22** §§9.2–9.3 Linked lists Comparing impl's	**23** §CI4.1 Lab 9: Interfaces and multiple implementations	**24** Ch. CI4 Inheritance is-a / has-a Hierarchies
10	**27** Ch. 12 Sorted lists	**29** §11.1 Quadratic sorts	**30** Ch. CI5 Lab 10: Overloading operators	**31** §§11.2–11.3 Faster sorts Comparing alg's
	November			
11	**3** Ch. 13 Queues Priority queues	**5** §§15.1–15.2 Trees Traversals	**6** §§16.1–16.2 Lab 11: Linked trees	**7** — Tree implementation
12	**10** §15.3 Binary search trees	**12** §16.3 BST implementation	**13** — Lab 12: BST implementation	**14** Ch. CI6 Iterators
13	**17** §§18.1–18.3 Maps	**19** §§17.1–17.2 Heaps	**20** — Lab: DT/Alg implementation	**21** §18.4 Hash tables
14	**24** §18.4.1 Good hash functions		[Thanksgiving **no class**]	
	December			
15	**1** — Presentation work	**3** — Presentations	**4** — Lab: DT/Alg implementation	**5** — Presentations **Exam 2 TH out**

Exam 2: Wednesday, 10 December, 3–5:30pm

* **15 October**: deadline to withdraw from a class (5pm)

Grading breakdown

I figure that I have about 10–15 hours of your time every week, including class and lab time as well as reading, practice, homework, and projects. If you find you're spending more time than this, please do come discuss it with me, and we'll see what we can work out. The work you do for this course will be evaluated as follows:

Preparation and participation. In this class you need to be awake, alert, and prepared. That means both having your materials (something to write with, something to write on, your textbook) and doing any practice work leading up to the class. Reading quiz grades fall into this category. You also need to be actively engaged in whatever the class is doing that day, paying attention and at least occasionally answering my questions and asking questions of your own. Note that attendance is a prerequisite for these points: if you aren't there, you may or may not be prepared but you definitely aren't participating. Preparation and participation are collectively worth 5% of the grade.

Lab/projects. Once a week we'll meet in the lab, and most of these labs will lead into or continue the writing of a related programming project. Some of the grade for those programs will be for design and testing, some for efficiency, and some for just plain "does it work?". These programs are collectively worth 35% of the final grade.

Homework. An important part of learning happens when you try things outside of the classroom, i.e. home-work. Much of that can and should be self-driven, but I will give some specific assignments throughout the term to help guide it. Each homework will proceed in two rounds: in response to your first handin, I'll give feedback (but no grade); after you have revised it, I'll assign a grade. Each problem will get 5, 3, or 0 points. The homeworks are group work: you can work with anyone in the class (or on your own if you prefer), and mark the names of the whole group at the top of the handin. They will be collectively worth 10% of the final grade.

Presentation. At the end of the term, you'll give a presentation about a data structure or algorithm not otherwise covered in the course (details below). This will be 10% of your grade.

Exams. There will be two exams, one in early October and one during the finals period. Each will have a take-home component and a sit-down portion. The final will not be explicitly cumulative, though of course the material from the second half of the course builds on the earlier stuff. **You are not permitted to discuss the exams, *at all,* with anyone other than me.** Each exam is worth 20% of the grade.

Presentations and final project

In the last weeks of the term, each student will, with a partner, give a presentation about a data structure or algorithm as well as writing an implementation relevant to it. The presentation will be 12–15 minutes and should include:

- Accurate example diagrams
- Pseudocode and tracing using the example
- A demonstration of either correctness or efficiency

Both partners must participate in the presentation but may divide the time as they see fit.

The implementation will be relevant to the presented topic but you'll negotiate the exact requirements with me. Both partners must contribute to the implementation, but it is group work (jointly submitted).

Your topics will be chosen from the following list:

AVL trees (§19.5)	Shellsort
B-trees (§21.3.3)	External sorting (§21.2)
Linear probing (§§18.4.2–18.4.3)	234 and red-black trees (§§19.3–19.4)
Circular arrays (§14.1.3)	Treaps
Heapsort (§17.4)	Leftist heaps
Splay trees	Skew heaps

Regardless of whether the topic is (partially) covered in the textbook, you will be encouraged to seek out other resources to research and understand it better.

Policies

Support

This is an introductory course. That means that what is covered is an important basis for other work in the field, *not* that it is supposed to be obvious, or easy. So don't feel bad if something doesn't click right away. Never hesitate to ask me a question; I'll usually at least give you a hint as to where to look next.

I'm in my office a lot (not just during posted office hours). Feel free to come in and ask questions (or just to talk). If you can't catch me in my office, email is probably your best bet.

Honor code policy

Above all, I ask and expect that you will conduct yourself with honesty and integrity—and not to ignore the other ten points of the Honor Code, either. Take pride in what you are capable of, and have the humility to give credit where it is due.

The two main forms of academic dishonesty are "cheating" and "plagiarism". "Cheating" is getting help from someplace you shouldn't, and "plagiarism" is presenting someone else's idea as if it's your own. If you ever find yourself inclined towards either of these, know that there are always other, better options. Persevere! See my website[3] for some discussion and examples of how to steer clear of these problems, and feel free to come talk to me if you need help finding some of those other options (even if it's for another course).

Cheating or plagiarism (on any assignment) will normally receive a *minimum* penalty of a lowered *course* grade, ranging up to an F in the course. Cases will also be turned in to the Honor Board. But: I believe in your potential, and I hope that you will, or will grow to, observe this policy not simply to evade punishment but positively as a matter of character.

[3] `http://cs.longwood.edu/~dblaheta/collab.html`

Accommodations

If you have any special need that I can accommodate, I'm happy to do so; come speak to me early in the term so we can set things up. If you have a documented disability, you should also contact Longwood's Office of Disability Resources (Graham Hall, x2391) to discuss some of the support the college can offer you. All such conversations are confidential.

Attendance and late policy

Attendance is required, and assignments must be turned in on time. That said, if you have a good reason to miss class or hand something in late, I tend to be fairly liberal with extensions if you ask in advance. (Good reasons do include assignments due for other classes.) (And medical and family emergencies are exempted from the "in advance" part, of course. But get a note from a dean.)

Frequent absence will result in a lowered participation grade; habitual absence may in extreme cases result in a failing grade for the class. *Unexcused* late assignments will normally be given a zero.

Inclement weather policy

I don't plan to cancel class for weather unless the entire college shuts down. If you are commuting or are otherwise significantly affected by a weather event, use your own best judgement; and if you do miss class for this reason, contact me as soon as possible to make up missed work.

Early bird policy

Nobody's perfect, and on occasion an assignment gets written a little unclearly (or, once in a while, with an actual error in it). If you catch one and bring it to my attention early, so that I can issue a clarification or correction to the rest of the class, there'll be some extra credit in it for you.