Syllabus
CMSC 162: Intro to algorithmic design II

Spring 2017

Lecture:	MWF 10am, Ruffner 352
Lab:	Thu 9:30am, Ruffner G56
Website:	`http://cs.longwood.edu/courses/cmsc162/1`

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` or `blahetadp@longwood.edu`
Office hours:	Mondays 1–2pm; Wednesdays 4–5:30pm; Thursdays 11–12:30pm

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.

Textbook and resources

The required book for this class is *C++ plus data structures*, 6e, by Nell Dale et al (ISBN 978-1-284-08918-9). 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)	[1]	
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.

[1] Alas, no A+, unfortunately.

Content

Calendar

Wk	M	W	R	F
	January			
1		**18** — Introductions Policies	**19** — Lab 1: Review and mastery	**20** §§1.0–1.3 Design and specification
2	**23** §1.4 Testing	**25** * §2.1 ADTs	**26** — Lab 2: Function design Unit testing	**27** §2.3 Classes and methods .h files
		February		
3	**30** §2.5 Exceptions	**1** §2.6 Algorithmic efficiency Big-O notation	**2** — Lab 3: Classes, I/O, 2D arrays	**3** §§3.1–3.2 Unsorted list ADT Array lists
4	**6** §3.2 Implementing an ADT	**8** §3.3 Pointers	**9** — Lab 4: Pointers	**10** §§3.4–3.5 Linked lists Comparing impl's
5	**13** §3.4 Linked impl of unsorted list	**15** §3.5 Comparing impl's of an ADT	**16** — Lab 5: Linked node methods	**17** §4.1 Sorted lists Binary search
6	**20** §§7.1–7.5 Recursion	**22** §§7.6, 7.8 Recursion cont'd Binary search revisited	**23** — Lab 6: Reading code `make`, `gdb` Backtracking	**24** ** §§7.10, 7.13 Tracing recursion **Exam 1 TH out**
		March		
7	**27** **Exam 1**	**1** §§5.1 Stacks	**2** — Lab 7: Using STL `stack`	**3** §§7.14–7.15 Stacks and recursion Recursive backtracking

* **25 January**: Deadline to add/drop classes (5pm)

** **24 February**: Deadline to elect pass/fail option (5pm)

Wk	M	W	R	F
	March			
		Spring Break		
8	**13** *	**15**	**16**	**17**
	§4.2	§§4.3, 7.7, 7.9	—	§§4.4–4.6
	Dynamically allocated arrays	Linked-list insertion	Lab 8: Empirical efficiency	Analysis and comparison of impl's
9	**20**	**22**	**23**	**24**
	§§6.1, 6.3–6.4	§6.5	—	§§6.7–6.8
	Templates Doubly-linked lists	Deep copying Operator overloading	Lab 9: Overloading operators	Inheritance Polymorphism
10	**27**	**29**	**30**	**31**
	§6.9	§§5.1, 5.3, 9.1	§11.1	§§5.2
	Iterators	Stacks, Queues, Priority queues	Lab 10: Interfaces and multiple implementations	Implementing stacks
	April			
11	**3**	**5**	**6**	**7**
	§§5.4	§§8.1–8.4	—	§§8.5–8.7
	Implementing queues	Trees Binary search trees	Lab 11: Impl'ing, traversing linked trees	BST implementation
12	**10**	**12**	**13**	**14**
	§§11.1–11.2	§§9.2–9.3	—	§11.3
	BSTs cont'd Sets and Maps `set` and `map`	Heaps	Lab 12: BST implementation	Hash tables and hash functions `unordered_set`
13	**17**	**19**	**20**	**21**
	§§12.1–12.4	§§12.5–12.6	—	§11.4
	Quadratic sorts	Faster sorts Comparing alg's	Lab: DT/Alg implementation	Good hash functions
14	**24**	[Showcase day **no class**]	**27**	**28**
	—		—	—
	Presentation work		Lab: DT/Alg implementation	Presentations
	May			
15	**1**			
	—			
	Presentations **Exam 2 TH out**			

Exam 2: Mon 8th, 8–10:30am

* **13 March**: 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. You need to be an active participant in this class: present, prepared, and on-task. The point for each day will be assigned using one of the following rubrics:

- Basic attendance: If you're there, you get the point!
- Participation: **1**: Attentive and on-task. **½ or 0**: Substantially late, sleeping, fussing with cellphone, etc.
- Reading quiz: Three questions, open-notes. **1**: Demonstrated that you read the assigned reading. **½**: At least some correct work on the quiz.

I will not, in general, tell you in advance which one I'll use on a particular day. These points are collectively worth 5% of the grade.

Labs and homework. An important part of learning happens when you try things outside of the classroom, i.e. home-work. In this course, it comes in two flavours: programming work, which will generally be connected to our once-a-week lab sessions and last about a week per assignment; and theoretical work, which will generally be due after just a few days but you'll have a chance to revise it. Programming work should be done basically on your own, but within limits you can talk to your classmates about it. (I call this work "collaborative" and go into much detail in my collaboration policy.) Theoretical homework will be group work, and you can hand in one copy for the whole group.

Labs and homework will be collectively worth 45% 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 late February 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 (§10.1)
B-trees (§10.3)
Linear probing (§11.3)
Heapsort (§9.4)
Splay trees
Shellsort
External sorting
234 and red-black trees (§10.2)
Treaps
Leftist heaps
Skew heaps
Radix sort (§12.7)
Parallel merge sort (§12.8)

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[2] 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.

[2] `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 contact me ASAP.)

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.