Syllabus tl;dr

CMSC 162: Intro to algorithmic design II

Spring 2021

Meets:	MWF 1:10, Rotunda 115 (or via Zoom); and Thu 12:30pm, via Zoom
Websites:	`https://canvas.longwood.edu/courses/5251553`
	`http://cs.longwood.edu/courses/cmsc162`
Professor:	Don Blaheta, Rotunda 334, `blahetadp@longwood.edu`
Office hours:	Mondays 4–5pm; Tuesdays 11–noon;
	Wednesdays 2:30–3:30pm; Thursdays 2–3pm

Textbook and resources

CS2 Software Design & Data Structures by the OpenDSA project.

`https://opendsa-server.cs.vt.edu/ODSA/Books/CS2/html/`

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.

Graded work

- Engagement 5%
- Labs and homework 45%
- Presentation 10%
- Exams 20%

  Exam 1 is out Monday, 22 February
  Exam 2 is out Monday, 26 April

Zoom attendance quick links

CLASS	OFFICE HOURS

Presentations and final project

In the last weeks of the term, each student will, with a partner or two, 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 needs to include:

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

Both/all partners must participate in the presentation but may divide the time as they see fit. More details will come later in the term.

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, 95)	A+	[95, 100]
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. Final grades of A+ are recorded as an A in the grading system. Final grades below the minimum for D− are recorded as an F.

Note that *individual* grades recorded in Canvas should be accurate (and you should let me know if there's a data entry error!), but *averages* as computed by Canvas sometimes are not, if the averaging is complex or (especially) if an individual student has a special case scenario. The reference gradebook is my own spreadsheet, and while I will try to make Canvas reflect it (including averages) as well as I can, Canvas can't always handle it.

Special note re mastery lab: You must eventually complete the first lab satisfactorily in order to get higher than a D+ for the course. See details in the syllabus and in the Lab 1 handout.

Calendar

Wk	M	W	R	F
	January			
1		**13** A — Introductions Policies	**14** — Lab 1: Review and mastery	**15** B §1.1 What is a Data Structure? Design and specification
2	[MLK Day **no class**]	**20** A §§2.1–2.1.1.1 Object-Oriented Design Classes and methods	**21** * — Lab 2: Classes, I/O, 2D arrays	**22** B §2.2 .h files Templates UML
3	**25** A §§1.2, 3.1 ADTs Lists	**27** B §§3.2–3.2.1 Implementing an ADT	**28** — Lab 3: Function design Unit testing	**29** A §§3.2.2 More implementation append, remove
	February			
4	**1** B TBA Pointers “Smart” pointers	**3** A — Dynamic allocation	**4** — Lab 4: Pointers	**5** B §§7.1–7.2 Recursion Fibonacci Binary search
5	**8** A TBA Linked nodes	**10** B §10.1 Linked List	**11** — Lab 5: Linked node methods	**12** A — Linked List implementation, ctd
6	**15** B §7.7 Tower of Hanoi Recursive algorithms	**17** A TBA Recursive backtracking The call stack	**18** — Lab 6: Reading code `make`, `gdb` Backtracking	**19** B** §6.1 Other uses of stacks Array-based stack implementation
7	**22** A §6.2 Stacks and recursion Linked Stacks **Exam 1 TH out**	[**Exam 1 no class**]	**25** — Lab 7: Using STL `stack` **Exam 1 due**	**26** B TBA Exceptions

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

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

Wk	M	W	R	F
	February			
8	[March Break **no class**]	**3** A — Classic ADTs The "big picture"	**4** — Lab 8: Empirical efficiency	**5** B §§4.2, 4.5 Algorithmic efficiency Big-O notation
9	**8** A §10.2 Comparing implementations Array List, Linked List revisited	**10** B §2.1 Inheritance is-a / has-a Hierarchies	**11** — Lab 9: Interfaces and multiple implementations	**12** A Ch. 8 Quadratic sorts
10	**15** B TBA Faster sorts comparing alg's	**17** A TBA Faster sorts, ctd	**18** TBA Lab 10: Overloading operators	**19** B §§9.1.1, 9.2 Queues Linked Queue
11	**22** A §§11.1–11.3 Trees Traversals	**24** B TBA Tree implementation	**25** §§16.1–16.2 Lab 11: Linked trees	**26** A — Tree implementation, ctd
			April	
12	**29** B §§11.4–11.4.2 Binary search trees	**31** A* §11.4.3 BST remove	[April Break **no class**]	[April Break **no class**]
13	**5** B TBA Maps/Dictionaries	**7** A — BST analysis, balance, rotation	**8** — Lab 12: BST implementation	**9** B TBA Heaps
14	**12** A TBA Hash tables	[Research Day **no class**]	**15** — Lab: DT/Alg implementation	**16** B — Model presentation Presentation debrief
15	**19** A — Presentation work day	**21** B — Presentations	**22** — Lab: DT/Alg implementation	**23** A — Presentations
16	**26** B — Presentations **Exam 2 TH out**			
	May			

Exam 2 due: Wed 5th, 5:30pm
Reserved for (online) presentation overflow if needed: Wed 5th 3–5:30pm

* **31 March**: Deadline to withdraw from a class (5pm)