CS 1C/2C Course Syllabus

Overview

Welcome to CS 1C/2C, a data structures and algorithms course that is the most important part of your CS education so far as it provides profound intellectual growth in making decisions that optimize computing resources. In this course, you'll develop the knowledge and intuition to choose the best data structure or algorithm for the problem given. You'll learn about space and time complexity and trade-offs between them as you design not only a solution to problems, but the optimal solution.

Roadmap

The prerequisite for this course is CS 1B/2B.

In CS 1A/2A, you learned some fundamental topics such as control flow, loops, arrays, and an introduction to object-oriented programming. In 1B/2B, our biggest themes were inheritance and memory management; we used these concepts in developing some basic data structures toward the end of the course. In this course, you will develop more complex data structures such as trees, hash tables, and graphs. You'll learn different ways of representing the same data and how to make design choices in your code.

Course logistics

Time, location, and use of time

We will meet Tuesdays and Thursdays, [TIME/PLACE REDACTED]. We will have a short break in the middle of each class.

Our class session time will include both lecture and other in-class learning tools. I author all the lecture materials, with input from some of our official resources. These materials will be made available to you in the "Files" section of Canvas as we progress through the course.

Important dates

Our final is scheduled for Thursday, December 13, [TIME/PLACE REDACTED]. The midterm is scheduled for Tuesday, October 30 during the full class time.

The tentative schedule is listed at the bottom of the syllabus in full and on the home page in part.

Text and references

The textbook recommended for our course is:

- For 2C: Data Structures & Algorithm Analysis in C++ by Mark Weiss. The 4th edition is the most recent.
- For 1C: Data Structures & Algorithm Analysis in Java by Mark Weiss. The 3rd edition is the most recent.

(Fun fact: Weiss studied with Robert Sedgewick, who studied with Donald Knuth!)

I have also posted modules that were authored by Michael Loceff, another faculty member here at Foothill College. The modules were based on the textbooks by Weiss. You may want to buy the Weiss book if you like reading a physical book (sometimes I find this easier), but it's optional.

In creating the lectures, I've also referenced:

- Introduction to Algorithms by Cormen et. al., an excellent and well-known algorithms text used at many schools. It is a bit more in depth than what we need, but I particularly liked the big-O explanation, as well as detail given in several sections.
- grokking algorithms by Bhargava. It covers only a few of the topics we'll cover and only in very little detail; however, it explains in intuitive ways and sometimes with cartoons.
- A Stanford CS 106B course reader :)

You don't need to buy any of these additional books -- this is just FYI. Where materials differ in opinions, those differences will be presented in class.

Instructor

My name is Joanna Lankester. I have worked as a software engineer and data scientist in multi-national corporations, startups, and my own small company, in addition to teaching. I love all things tech, have multiple projects going (always!), and write code every day. I also love teaching, mentorship, and Foothill College in general!

I prefer to go by Professor in class (Professor Joanna or Professor Lankester, doesn't matter which). She/her/hers.

TA

We will have a teaching assistant who took this class in a prior quarter: Ian Cramer will be an embedded tutor in our classroom. Please feel free to ask him questions during our in-class activities.

Communication

Please post questions about the assignments in the discussions. If you have a question that is specific to you (e.g. registration issues, etc), you can message me on Canvas or email me at [EMAIL REDACTED]. Although not required, I appreciate it when you let me know if you expect to miss class.

Posting in Canvas discussions

The discussion board provides a great way to get quick help from each other and from me when completing the programming labs. Do make use of it!

Please do not post homework code, whether a question or an answer, in Canvas. Learning to write code involves synthesizing information, trying out examples on your own, and figuring out why the code may not be working the first time. Important to that process is your (and your classmates') opportunity to figure out the code without seeing the answer ahead of time.

When asking a question in the discussion board, make the question as specific as possible. If it's an error in your code you haven't been able to solve, describe what you've tried. The more specific your question, the more likely someone will be able to help.

Feel free to answer other students' questions; this is an encouraged and positive way to interact with classmates. Do not post homework solution code, but you can refer each other to where in the modules or book you found a similar example, or paste in some code from lecture that helped.

Optional but highly recommended workshop

[NAME REDACTED], a tutor in the STEM Center, will be giving a weekly workshop on Wednesdays, 4-5:30 pm at room 4218 in the STEM center (next to the CS lab). The workshop will cover algorithms related to the course material and/or the weekly assignment; the material will be presented from a language-agnostic viewpoint since it's for both 1C and 2C students.

Software needed

The recommended Integrated Development Environment (IDE) for this course is:

- 1C: Eclipse for both Windows and Mac.
- 2C: Visual Studio for Windows and Xcode for Mac; the modules include instructions for setup.

Instructions are provided for setup of these IDEs. Of course, if you prefer a different IDE and can set it up on your own, you are welcome to use that instead.

Office Hours

- Tuesdays, [TIME/PLACE REDACTED]
- Or by appointment.

If you need a 1:1 meeting (e.g. other topics that don't pertain to other students), you can either email me to set up an appointment, or wait around until others leave -- whichever you prefer.

Assessments and grading

The following assessments will constitute the indicated percentage of your final grade in the course:

Category	Percentage
Final	25

Midterm	12
max(in-class questions, final)	10
Programming assignments	53

Importantly, note that the exams are mandatory. A passing score will not be assigned if an exam is missed, even if the rest of your percentage score is over a passing grade.

Programming assignments

Programming assignments will usually be due on Wednesdays at 11:59 am (right before the afternoon). For some, we may have extra time beyond a week, resulting in a due date on a different day of the week.

Please submit code in the following format:

CS 1C: Please make a new folder named with something that starts with your last name, then copy and paste all the .java files you created into that folder. Zip (compress) the folder and submit that. Include your Foothill.java testing file. Please include a run of your code as a .txt file in the folder. Include any provided course code that you used in the format specified on the assignment.

CS 2C: Please make a single text file containing all prototypes, class definitions, and static constant definitions. Please comment out the main function entirely and include a commented-out run of your code at the bottom of the file. No need to include provided course code that you used.

Both: Please include a commented-out run of your code as described (Ok to paste several together as needed).

You will learn the most if you can fully debug your code and get it to run. Therefore late work will be accepted (at a 10% penalty per day) for up to 3 days.

Although Canvas has a resubmit button, it would be impossible for me to both accept late work and grade your most recent code, since I would not have a way of knowing whether you would resubmit. Therefore, while you may resubmit multiple times before

the deadline, be advised that after the deadline I may start grading at any time and cannot take a more recent submission if I've already started grading. Check with me if you have any questions. Unfortunately, a revise and regrade is super time consuming and thus not possible.

Code should be submitted according to the style guide.

The programming assignments will constitute the majority of your effort in this course. The effort you put in will pay off, just as it does in other skills you learn, such as sports, arts, or musical instruments. Please set aside enough time each week to complete these assignments; make sure to start earlier than the night before. I encourage you to put in the effort to think through the problems on your own, and then ask questions in the discussion forums as needed.

I will ensure that the assignments are graded in a consistent way between students.

In-class questions

We will do some in-class questions or activities to help enhance your learning. The in-class questions give you a chance to try a piece of code or concept in class that relates to the prior lecture, but that is much smaller in scope than the programming assignments. They are great practice for the exam. The questions are usually given at the beginning of lecture, so please come on time. Note that the lowest two scores are dropped, so there should be no need to email me to ask about making up these questions.

If your score on the final is higher than your in-class question score, I will replace it. For the vast majority of students, the in-class question score is higher, so get to class! :)

Midterm and final exam

Please plan ahead for the exams. Exams will be given on site (in the classroom). The exams cannot be missed; they are mandatory to pass the course. You will be allowed to bring in some sort of reference sheet on exams; I'll announce information closer to the exam dates.

I author my own exams.

We'll have a time to review midterm exams in class.

Regrades

I spend quite a bit of time reading and testing your code and providing you a high-quality, detailed review. I also give consistent grades across similar programs. For these reasons, I don't reconsider the number of points awarded based on the error made. I don't accept regrade requests on the basis that your code works on your computer, but not on mine. If you thoroughly test and debug your code, it will work on my computer as well.

However, please *do* let me know by email if you ever believe that I haven't seen one of your files or part of your code, for example, or if there is any ambiguity in the feedback provided.

NOTE: the official policy of this course is that any regrades, if granted, are subject to a full regrade, which could result in a lower score. Please send requests by email within 72 hours of a graded item being returned to you.

Questions about feedback

You are always welcome and encouraged to ask questions about the feedback I have written for your code. Some of the best learning happens during these discussions!

Extraordinary circumstances

Any extraordinary circumstances that interfere with your ability to complete work for this course will require official documentation.

Academic integrity

Programming assignments should be 100% your own work. Although you are free to discuss concepts with classmates, you should write up your own solution. A good test to make sure you're working adequately on your own is if, after a discussion and not writing down anything, you can go back to your computer and code the solution on your own. No one, including any tutor, should ever be typing into your code. Please do not send your code to someone else in the course.

Any violation of the academic integrity clause of our course may be penalized up to and including a zero on the assignment and a referral to the dean of students.

I reserve the right to ask you to explain your code.

I diligently screen for matching code between classmates. This should be a relief to most students: you are working hard to get the work done, and I want to make sure you are getting the credit you deserve. You may find this surprising, but even within completely correct code on the same assignment, you all have an individual coding style, similar to a writing style or speaking style.

On-campus resources

Disability-Related Accommodations

If needed, please contact the Disability Resource CenterLinks to an external site. as early as possible by visiting the DRC in room 5400, emailing the DRC at adaptivelearningdrc@foothill.edu, or calling DRC at 650-949-7017 to make an appointment.

The STEM Center

The STEM center in room 4213 offers free help to all students in science/math/technology courses and even has a separate Computer Science lab in room 4204. Please see more information including open hours at the STEM center

websiteLinks to an external site.. We also have online tutoring for CS (Links to an external site.)Links to an external site. later at night.

Although I regularly work in the STEM center, I will need to help students in the order they arrive (including students from other courses) when I am scheduled there. Please sign in/sign up for help through all the usual procedures.

Fall 2018: I am scheduled to staff the CS lab in the STEM center on [TIME REDACTED].

Psychological Services and Personal Counseling

Many students find themselves in a difficult or stressful time at some point. We are fortunate to have a great team of counselorsLinks to an external site. who can talk with you about whatever you're going through. Please don't hesitate to contact them.

Questions?

If you have any questions about this course, please don't hesitate to message me and ask. If you have any other general questions, things you want to know about our campus, request for extra topics, general career questions, etc, just let me know anytime.

The advanced class is an incredible time of intellectual growth; you are going to see the world in a different way after this course. I'm looking forward to this class, and I hope you enjoy it too.

Schedule content

Here is the tentative schedule of content for our course. Except for exam dates, this schedule is always subject to modifications in order to remain flexible; we may need more or less time on topics, and I like for the time to be somewhat guided by student interest as well.

Date	Topic
9/25/18	Course intro. Timing linear search. Template and STL use review. Timing vector insert*. Subset sum. Vim intro.
9/27/18	Timing bubble sort*. Iterator vs const iterator. Building out a vector class.
10/2/18	Building out a linked list. Implementing iterators. Matrix representations. Sparse matrices.
10/4/18	Building a stack from our other data types*. Big-O, theta, omega. Identifying runtimes of linear and quadratic.
10/9/18	Matrix multiplication. Logarithmic and exponential time. Pros and cons of recursion: fibonnaci.
10/11/18	Review of tree traversals. Templated BSTs. Functors. Lazy/soft deletion in trees.
10/16/18	Review general trees. Tree search methods: breadth-first vs. depth-first; introduce AVL.
10/18/18	AVL trees. Rotation.
10/23/18	Splay trees.
10/25/18	Dynamic programming. Knapsack problem, maximum substring problem. Greedy algorithms*. Review for midterm
10/30/18	MIDTERM
11/1/18	Intro to hash tables, collisions; separate chaining.
11/6/18	Hash tables; quadratic probing.
11/8/18	Priority queues. Heaps and heap sort.
11/13/18	Insertion sort, selection sort, shell sort.

11/15/18	Merge sort, quick sort.
11/20/18	Practice with STL containers. Intro to graph representations. Traversing graphs with BFS, DFS.
11/27/18	Path finding: Dijkstra's algorithm, Euler circuits
11/29/18	Minimum Spanning Trees
12/4/18	Maximum Flow
12/6/18	Review for Final
12/11/18	n/a finals week
12/13/18	FINAL

Other items I'm required to include in this syllabus

Number of hours that are considered hybrid per week: 2 (relates to credit hours for course; does not relate to amount of time labs will require, which is generally more)

Hybrid hours are: lab

Student attendance during hybrid hours is mandatory (this just means that your programming assignments are a component of the course).

Activities students need to do for hybrid course: programming labs

Student learning outcomes: See course outlines (Links to an external site.)Links to an external site.