CS 2A Syllabus - Fall 2022

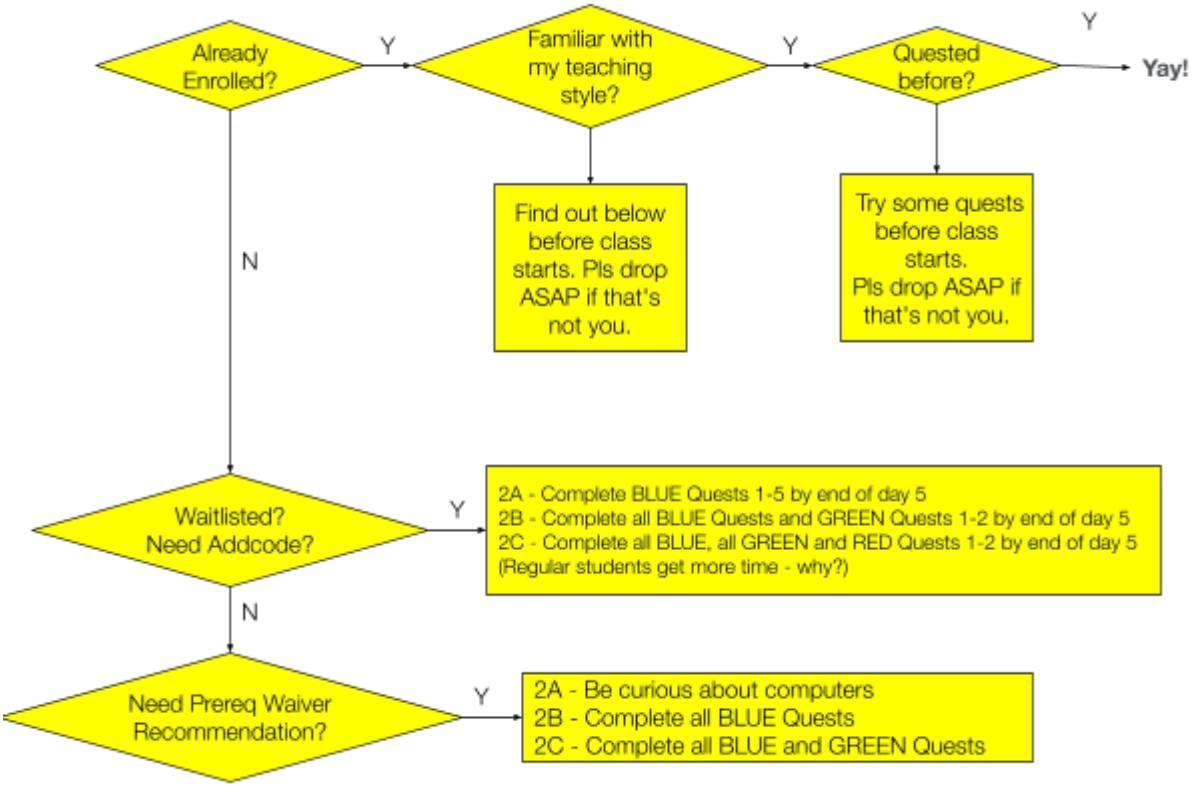
Introduction to Object Oriented Programming in C++

Hey there. My name is Anand. Please read this syllabus carefully. You should especially read it if you have never taken a class from me before.

Jump To

Section I - First Things

First things first - Should you enroll?



You will **not** be spoon-fed information in this class. You will have to learn to do things for yourself. You will not be given answers.

In the past many students who enrolled in this class expected to be given freebie answers and goodies earned through someone else's effort, as if that is a normal thing in their lives. If you have got used to getting things done for you, you will have a hard time. Many such students have dropped this class in frustration complaining that they received NO HELP, when it is more likely that they received NO FREEBIE ANSWERS they expected to arrive automatically.

With that out of the way, here's what you can get: Plenty of opportunities to enjoy finding answers yourselves with just a little bit of effort.

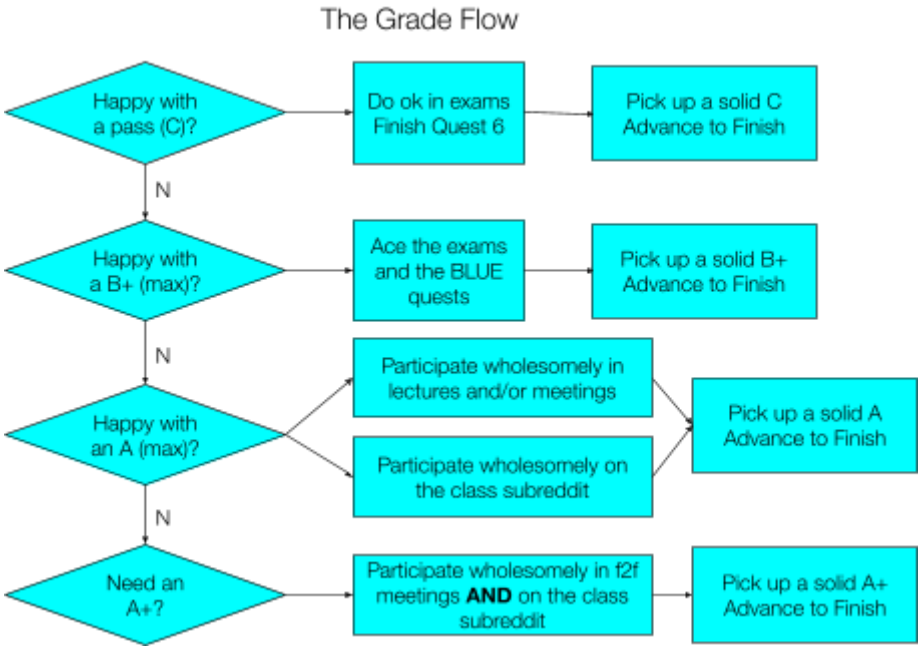
Note the following **essential** prerequisite skills.

- Looking up, evaluating, and using information on the net
- Following simple directions correctly (e.g. creating a subreddit user according to some requirements)

If you don't have these skills yet, take some time to learn them first (usually, quite easy) and then enroll.

If you're doing the quests for your own edification at your own pace - awesome. That's the way to be. No fun being full of stress, either before your tests or during your quests.

First Things Second - General Matters



YES! Whether you're doing this class to just get a feel for this thing called C++, whether you want to be good at it, but not necessarily make it the center of your programming world, or whether you want to learn from this class and prepare to dive into the depths of CS and swim with the sharks… this class has a track for you.

If you just want a smattering of programming knowledge in C++, you should complete quests 1-6, and do ok in the quizzes and exams. No need to participate in the forums. You'll get your passing grade.

For a B+, you should do well in your quizzes and exams, and also complete all 9 quests, without necessarily acing anything.

To get to A Territory, you absolutely need to earn struggles/mastery (SM) points, which you can assume to be a proxy for evidence of productive collaborative programming, which is an important skill to have in the real world. Section IV in this syllabus shows you how to get them.

Note: *I try to keep everything public, including 1-1 meetings you may have with me unless it involves confidential matters. Foothill offers a number of qualified counselors who can discuss confidential matters with you to suggest good solutions.*

Any class-related discussion that may be generally useful will be made available as transcripts or recordings at publicly accessible media sites (e.g. YouTube or Reddit). Make sure you are ok with this policy before enrolling (or make special arrangements with me before class starts).

Section II - Administrative stuff

CS 2A is an introduction to object-oriented programming using the C++ language. Absolute beginners or students already familiar with other programming languages will learn how to write C++ programs that cover a wide range of applications. The ability to work with computers and access to the internet are the only prerequisites.

A familiarity with algebra and good written English comprehension skills are both advisories. Please arrange tutors/helpers right away if you need assistance with either. Contact the STEM Center or the TLC.

Course outline and SLOs

You can access the official course outline of record for all CS courses here. Student Learning Outcomes for this course are:

1. A successful student will be able to write and debug C++ programs which make use of the fundamental control structures and method-building techniques common to all programming languages. Specifically, the student will use data types, input, output, iterative, conditional, and functional components of the language in his or her programs.
2. A successful student will be able to use object-oriented programming techniques to design and implement a clear, well-structured C++ program. Specifically, the student will use and design classes and objects in his or her programs.

By enrolling in this course in **Fall 2022**, you are implicitly agreeing that this syllabus provides a bare minimum of what you may experience during a one-quarter run of this class. However, experimental variations may gradually be introduced on a per section basis. You agree to be part of these too and to meet reasonable (as determined by me) additional flexible learning requirements that may be incorporated into this class before the finals (totals may be scaled appropriately if you're doing it for a grade).

Canvas and **required tasks**

Students at Foothill College doing this class for credit should use Canvas to coordinate some activities and take online exams.

Most of the rest of our work will happen at other public locations, including youtube, zoom, quests, reddit, etc.

You will start your adventure in Foothill's class by completing the following **required** tasks.

1. posting an introductory note about yourself in Canvas (You can simply reply to my own introductory post if you prefer) and
2. scoring at least 90% on the syllabus quiz (in Canvas). This doesn't need knowledge of CS or c++.

These **must be completed before the end of the first Friday** of the quarter to prevent your enrollment being reassigned to someone else on the waiting list.

Assessment

If you're doing this course for kicks, or other fun reasons, you can skip this section.

In this course there will be:

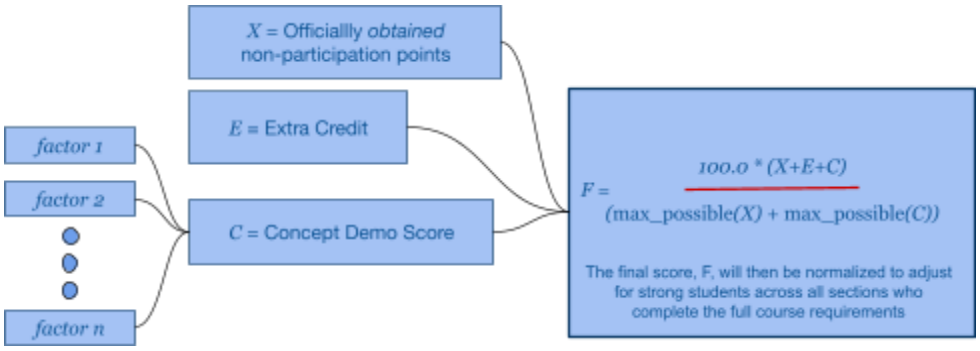
- REQUIRED - One syllabus quiz (**5** points)
- REQUIRED - One data representation quiz (**20** points)
- 9 BLUE Mystery Quests you must solve starting week 2, at the rate of about one per week - refer to the freeze schedule (total capped at **180** points for **2A** this quarter)
- 1 Midterm on the Thu of W6 (**25** points)
- 1 Final exam on the Thu of W12 (**50** points)
- Struggles/Mastery (**50** points - see below)

Exams

Exams are objective style and will be administered via Canvas. You will typically have a window of time (18+ hours) during which you can begin these exams. But once you begin, the current version of Canvas does not allow you to *pause* your exam and come back to it. The 1h (or 2h for final) timer cannot be stopped once you start it, until you hit finish.

They can be taken anywhere you get a decent Internet connection. But I don't recommend taking them on mobile devices. I also suggest that you don't actually try out code-fragments from your question sheet in an IDE.

You may get various extra credit goodies, including grade points and sometimes cash prizes, during the quarter. In the end, this is how your final score will be determined (**and reviewed/verified by me**).



According to this syllabus, $\text{max\_possible}(X)$ should be **280** - Yes?

$\text{max\_possible}(C)$ = **50**. Find details in Section IV.

Are there lectures?

For CS2A, I offer both pure virtual and synchronous classes whenever possible. Synchronous may be either face to face on campus, or via zoom..

Lectures are mandatory for those in synchronous (Z) sections. Class cuts, including being habitually unresponsive when spoken to during class, eat into your participation score (C) as you can see above. CS2A students who are enrolled in a sync section can earn a significant amount of Concept Demo points by attending class *actively*, especially if you get an opportunity to *live-code* in class.

Want to see other students live-coding in action? Check out our past recordings. I made a list (about 3 quarters of CS2A classes) at:

https://www.reddit.com/r/cs2a/comments/o2vpki/complete_list_of_recorded_classes

If you're in my non-synchronous section, you will have to post a weekly update of progress made in your blue trail under the Foothill flair on our forum. These posts will be graded into my secret spreadsheet with scores between 0-10 for each update. At the end of the quarter, the total points for such posts will be scaled into the same range as the lecture points for the sync students. Virtual students can also choose to instead attend sync lectures *actively* whenever possible.

However, the above is NOT ENOUGH to ace your **C** score. It only gets you halfway to the max.

Informative, helpful or insightful posts in the class sub are an essential component of the concept demo score and is worth the other half of the total (25 points). These will start expiring in week 2.

Personal Tutorial Hours

Hooray! You have an official tutor who can give you at least 8 and UP TO 12 hours of his time each week this quarter. His name is Walter Bergstrom. He completed CS2C in the spring. You can see Walter on our previous class videos. We have been discussing having recorded zoom group office/tutorial hours with Walter each week.

We can decide on a schedule for this with Walter on the sub once the quarter starts.

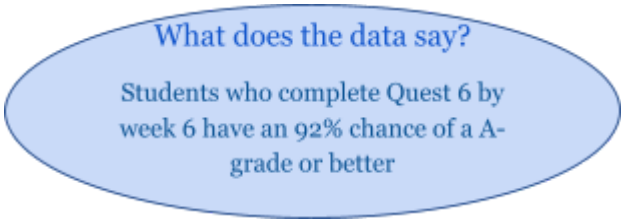
Section III - Informative stuff

This part of the syllabus is optional. It is meant to ease your journey on the way to a successful completion of this class.

Mystery Quests (capped at **180** for CS2A this quarter)

You will solve these at the public questing site (https://quests.nonlinearmedia.org).

Each quest will give you a certain number of trophies. You can check your total trophy count at any time by visiting your personal scoreboard at the /q site (It will be wiped on the 1st of Jan, Apr, Jul, and Oct). *It will show you all your trophies, but only the ones you earned for 2A (BLUE ones) count for this course.* Your secret handle is your Student ID.



The quests are set up such that the password to each quest is given out upon scoring a certain number of trophies in the preceding quest. However, I found that a few students were getting stuck in the lower numbered quests pounding away at them to eke out every remaining trophy before moving on, even though they had already earned the password. This is a bad strategy. Resist your temptation to hack more and keep moving when you get a password. P*upping*[1] a quest is sufficient to get full points for grade purposes. You can always come back to polish your previous quests when you have free time before the freeze date. You get about one week on average before each quest freezes.

At the end of the quarter, your total trophy count will be capped at **180** and fed into the grade crunching system. If you spend a lot of effort getting up to high numbers by the time you get to Quest 7 already, then you'll be close to getting burned out right in time for two of the funnest quests of all. So plan your time and effort wisely. It's not like your old quests are going to disappear when you move on.

What is a Quest Freeze?

A freeze is when I will review your submissions and transfer scores from a particular quest into Canvas.

If a quest has frozen, you still have to complete it in order to move forward into the next quest, but your trophies for it will not be counted towards your grade!

Bummer... Yes? If I were you I would try my best to avoid that situation by staying on top of things and taking this class ***seriously***.

However: There is one glimmer of hope if you missed a freeze deadline. You can plow through and complete Quest 9, earning the password for the first GREEN quest. Then all quests automatically become unfrozen, and all points, including for your frozen quests, will get transferred to Canvas towards your grade (*except for ones that were flagged by the bursty-progress indicator - this means you can't just cold-ace multiple quests in one week for points*).

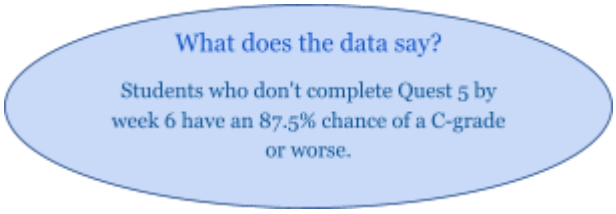
---

[1] Progress Until Password

Getting started on your mystery trail

The password for the first quest is A Tiger named Fangs. Passwords for subsequent quests will be automatically revealed upon *satisfactory progress* (as the machine sees it) in each preceding quest. The first five BLUE quests range from trivial to simple and are meant to give you an opportunity to get familiar with basic control structures and the questing system.

If you've never coded in c++ before, expect that you may retire for the night without feeling any closer to the solution for several days initially. This feeling of making no progress and suspecting that you're not cut out to be a computer scientist *is an important feeling* to feel SEVERAL nights until you get used to seeing it bubble up and settle down on its own. This takes both time and exposure, and you cannot shortchange yourself on either.



In order for rewards from a quest to count towards your total, you must have completed all previous quests. If you leave a hole in your trail of completed quests, then your total reward earnings is the sum of all rewards you earned before the first incomplete quest.

You can quest as much as you want. Everything you do is logged. Quest with your ID and we can watch your struggles and progress, and know when it is best to help.

Bugs in your code?

Getting your code debugged by someone else is NOT allowed. That includes me, tutors, teachers, friends, enemies and relatives. Debugging your own code is an essential skill that aspiring programmers must learn and enjoy - Yes, enjoy!

Of course, I can't police this. But your enrollment in this class signifies acceptance of this condition (in addition to being bound by Foothill's Academic Integrity Policy). You cannot send your code to me, a tutor, or someone else and ask them what the issue is. What you can do is:

1. Explain (in our **subreddit**) what you're trying to do
2. Describe in English the steps you think might work, or if you have no idea and would like someone to explain the requirement better.
3. Or describe the behavior of your program and ask why it is not working as expected.

Sometimes it is also ok to post your code on our **subreddit**. Mostly, exercise good judgment regarding what can be shared. You want a fun and fulfilling learning experience. The best way to get it is to keep it fun and fulfilling for everyone. You wouldn't give away a movie's ending to a friend who's going to watch it. Why give them the solution to a problem when they can feel good finding it themselves?

Suggested Schedule

Programming, like all art, is not a 9-5 job. Sometimes you're on a roll and killing it. Other times, not so much. I know how it is.

So there are no regular papers or labs due every day or week in this course. Rather, like real projects, there are deadlines you should strive to meet. You can plan your own time in your own way. Here is one suggestion:

Week	Read References	Complete	Notes
1	Data Representation Quiz (on Canvas)	DR Quiz	
	Vars, Exprs, Streams	Mystery Quest 1	
3	Control Structures	Mystery Quest 2	Q1 Freezes (Mon AM)
	Functions, Param passing	Mystery Quest 3	Q2 Freezes (Mon AM)
5	Arrays and Vectors	Mystery Quest 4	Q3 Freezes (Mon AM)
	Review/Midterm (on Canvas)	Mystery Quest 5	Q4 Freezes (Mon AM)
7	Objects, Classes	Mystery Quest 6	Q5 Freezes (Mon AM)
	Methods & Params	Mystery Quest 7	Q6 Freezes (Mon AM)
9	Pointers, Mem Mgmt.	Mystery Quest 8	Q7 Freezes (Mon AM)
	Algorithms, Searching	Mystery Quest 9	Q8 Freezes (Mon AM)
11	Sorting		Q9 Freezes (Mon AM)
	Final Exam (on Canvas)		

Col 2 is also the order in which the topics will be covered in lectures (for sync classes)

Every week, give yourself one or more topics to study and one or more programming quests to complete. If you have some programming experience already, expect to spend about 8-12 hours per week reading and/or attending lectures or watching videos. Budget an additional 10-15 hours for working on programming quests. To be on the safe side, budget about 25 hours per week (initially) for this course. Many students have underestimated the time required.

Extensions

Extensions don't make sense because the quests are self-paced. You just have to complete each by their "freeze" date to get credit. After their freeze dates, you should still complete them, but not for credit. There's a LOT of time to complete these quests even if you have to take some breaks. So, there is no need to ask for extensions.

Programming style and compilers

My personal preference for program formatting is the C++ equivalent of the classic K&R style for C. It's not imperative that you follow the K&R style. I'm ok with any consistent and clean styling/formatting of your programs.

Use an IDE/compiler of your choice. But you'll find better support from me and the STEM center if you stick to one of the environments we know about (ask).

Section IV - Struggles/Mastery Points

These points are estimated by a multi-factor system and will contribute to the C score in Figure G.

You won't have access to the full details, but if it helps, the top few positive factors *1...n* that go into my predictive model for ***C*** are:

1. Number of interesting or helpful posts each day (lost points for previous days cannot be recouped)
2. Number of nontrivial comments made on posts **NO OLDER than 2 days**. That's correct - comments on posts older than 2 days don't count
3. Total of presence and participation scores during *online catch-up circles* (active = 10, passive = 0 per session)
4. Normalized score received from class tutor
5. Ratio of *receive* vs *give forum posts/comments*
6. Number of insightful CS observations
7. etc.

The top few negative factors are:

1. *Bursty progress* meter score - this would be returning to the questing system once in a while to make some progress and disappearing for days at a stretch (tagged by a system that doesn't need your Student ID)
2. Count of off-topic, self-promotional, mean and/or boastful posts (tagged by me)
3. Count of posts or comments or server log entries showing evidence of actual solution lookups (not of concept pages and media) - Examples are references to cheat-code websites, links to code very close to solutions, or links to subreddit posts *from a previous quarter* (tagged by me)
4. Score from grinding-behavior detection sub-system
5. Score from plagiarism-detection sub-system
6. etc.

Note - Asking a question in the forums may also count as being *helpful to others*. Surprised? Weigh in on any of the many valid reasons on our sub.

To be on the safe side, avoid referring to past posts on the subreddit ever - until AFTER you have completed a particular quest. And make it a point to code *every single day* to avoid the risk of burst-coding.

Essentially, points are obtained for evidence of concept mastery. The only two accepted ways to do this are:

1. Incremental progress by questing with your ID and following up with posts relating to your struggles and/or victory over them in the sub every day.
2. Alternatively, if the quest was too simple for you, posting a significant number of helpful hints (without giving away the solution) and/or tip sheets for completed quests.

Students who show no incremental effort and progress, who don't support their successful solutions by tip sheets or other posts corroborating their learning, and those who miraculously submit a correct or close-to-correct solution shortly before the deadline will be deemed to have copied the answer from the internet or past posts. They can arrange to have a private 1-1 exam with me over zoom when I can test their conceptual knowledge and award points in lieu.

Grinding and Bursting

I define *grinding* as many rapid submissions to the questing site without much thought going into exactly what change you're making to your code or why, in the blind hope it would work. It is like spraying bullets in the dark hoping one of them will hit your target. If this tendency is not nipped early, it will not make for a successful builder of anything, let alone programs.

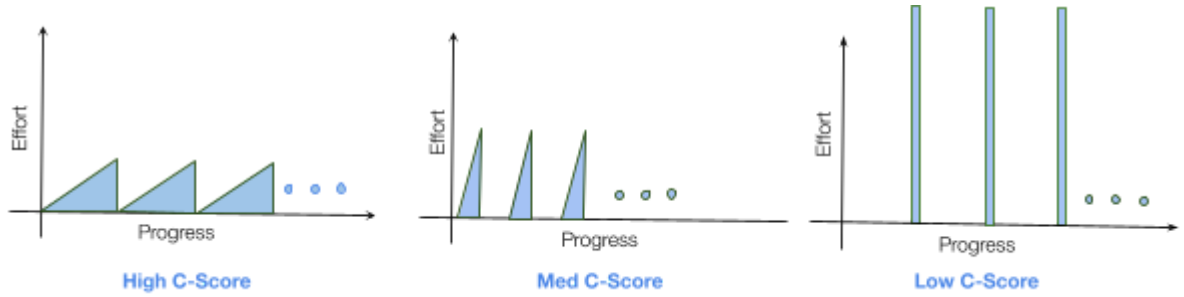
It does not matter if you're *grinding* in anonymous mode without a Student ID. The system can flag your code if it suspects grinding behavior or code similar to any code that was *ground by anyone in the past*. In that case, your concept demo points will take a serious hit and you may not know.

Bursting (Burst-mode coding) is when you are absent from the questing system for 2+ days, then return to quest intensely for a day, and then disappear again. Repeated manifestation of this behavior causes the system to flag your code with the burst-mode tag which is unfavorably graded.

To avoid this, quest and make at least some progress (1+ trophies) every single day. This is favorably graded by the system.

A Questing Tip: One way to boost the likelihood of getting participation points is to quest in non-anonymous mode with your Student ID. Your increasingly successful attempts will be better evidence of concept mastery than a sudden, non-anonymous, but successful solution, such as you can buy online.

As a quick reference for what constitutes bursty progress refer to the following picture:



Three kinds of progress charts and corresponding SM points rates
(example shown for the first 3 quests)

Summary: Avoid Bursty Progress - Aim to get at least 4 trophies EVERY SINGLE DAY

I suspect that the C-Score can be approximated by the formula:

C = 50 * **Probability(**solution was original**)**

Low probability scores suggest that the solution was a direct lookup online or on our subreddit, or has significant code written by someone else which the submitter does not actually understand. You can increase this value by either talking about your struggles as you overcome them in the sub, or by helping others who are struggling, or by posting interesting observations/tips about some aspect of the quest you found salient or hard - these need to happen on our official subreddit.

Reddit Recommendations

Don't say anything in the forums that you'll end up regretting later in your life. OTOH do try to let your natural genuine curiosity shine through for others to seek out in a sea of wannabe programmers. Maintain your profile on our subs as you would if you were a professional and it will free up a lot of your time.

I will try to remove posts that I deem (in my subjective opinion) to be a liability to your future self. But you can't rely on it. Best to be helpful, courteous, informative and only post useful tips, tricks and observations. They usually have more lasting value.

ANY user anywhere in the world can quest and post/discuss in our subreddits. So you may see posts and replies by users with anonymous names like *coding_lion*, *bat_girl* and such. All posts are subject to the same rules like *Johnny be good*, but only the ones with avatar names matching the spec in this syllabus will get participation credit: Your reddit avatar name **must** start with your first name (as on Canvas) and an underscore, followed by your initial (or full last name) + some optional digits (example: *ramanujan_s1729*).

Can you review past subreddit posts?

Access to past posts in the subreddits is off-limits to current students (also recommended for current non-student questers). For students this is partly by honor code, but also by way of a slight negative reinforcement via a weight applied to posts and comments referencing actual solutions (instead of hints, tips and other pointers).

Imagine you are on equal footing with past students who did NOT have these resources but were still able to do well in this class.

You are expected to use the subreddit to communicate with your current questers, and to demonstrate both your conceptual understanding and willingness to help others technically.[2]

After you successfully complete a quest on your own (with possible contemporary help), you can refer to all past posts for that quest and link to them in your tip sheet.

Tip sheets don't need to contain tips for ALL minis. Just the few salient ones you discovered are fine to mention.

[2] If you absolutely must, allow yourself the luxury (no more than a few times) of looking up past posts, but ONLY **AFTER** you have pupped the corresponding quest.

Section IV - Resources

My first resource suggestion is the book: Absolute C++ by Walter Savitch, any edition > 2. You can borrow free copies from the library.

I also have a fork of CS2A/B/C modules that ex-prof Michael Loceff created when he taught this course. Thanks to Michael, I'm able to make these available to everyone completely free.

Click on the appropriate link to access them: cs2a, cs2b, cs2c

Although a couple of revisions behind, much of it is still relevant to this course. It is essentially a *distillation* of selected topics from the text. But be aware of salient differences between the content of his modules (or the text) and what some of our quests require. This shouldn't be a problem if you understand the concepts. But it will be a problem if you don't. ,

> As always, hit our sub, when in doubt.
> Actually, that's not quite it.
> When in doubt, try it out.
> If you still just don't get it. Then hit our subreddit (to ask - not to look up)

Lane's Lane

The Foothill STEM Center already provides fantastic assistance by making experienced CS tutors available for 1-1 real-time (synchronous) assistance almost 24/7 (via zoom) and generous hours in the STEM Center when the campus is open. Within the STEM Center, Lane Johnson hosts two special workshops each week focused especially on helping questers. Look for their actual hours on our **sub**, or simply check into the STEM Center sometime and ask for Lane.

Tutor Walter

Stay tuned for announcements from your class tutor regarding group tutoring sessions he may organize.

Disability Resource Center

Foothill College is committed to providing equitable access to learning opportunities for all students. Disability Resource Center (DRC) is the campus office that collaborates with students who have disabilities to provide and/or arrange reasonable accommodations.

If you have, or think you have, a disability in any area such as mental health, attention, learning, chronic health, sensory, or physical, please contact DRC to arrange a confidential discussion regarding equitable access and reasonable accommodations.

If you are registered with DRC and have a disability accommodation letter of accommodations set by a DRC counselor for this quarter, please use Clockwork to send your accommodation letter to your instructor and contact your instructor early in the quarter to review how the accommodations will be applied in the course.

Students who need accommodated test proctoring must contact the DRC immediately if they cannot find or utilize your MyPortal Clockwork Portal. DRC strives to provide accommodations in a reasonable and timely

manner. Some accommodations may take additional time to arrange. We encourage you to work with DRC and your faculty as early in the quarter as possible so that we may ensure that your learning experience is accessible and successful.

To obtain disability-related accommodations, students must contact the Disability Resource Center (DRC) as early as possible in the quarter. To contact DRC, you may:

Visit DRC in Building 5400, Student Resource Center (physical visits suspended during college closure).

- On the web: http://www.foothill.edu/drc/
- Email DRC at drc@foothill.edu
- Call DRC at 650-949-7017 to make an appointment

Other Resources

The department maintains a blog called Opportunities for CS students. It has announcements of internships, scholarships, free software offers, public lectures, etc. You can also check out our numerous SLI opportunities.

Important Dates

For a list of important dates for the fall quarter, see the official college page here.

Bottom line

Computer science is a **hard science**, not a soft science. A significant investment of your time and effort will be required in this class. To succeed in becoming a software engineer or computer scientist,, you must expect to code at least 2 hours EACH and EVERY single day for the next 12 weeks. Make sure your schedule allows it before you start. Now smile, and get ready to get wasted (in a good way). If you apply yourself sincerely, you will learn a REALLY USEFUL skill for a happy life.

Happy Hacking!

&