CS 2C Syllabus - Fall 2022

Algorithms and Data Structures in C++

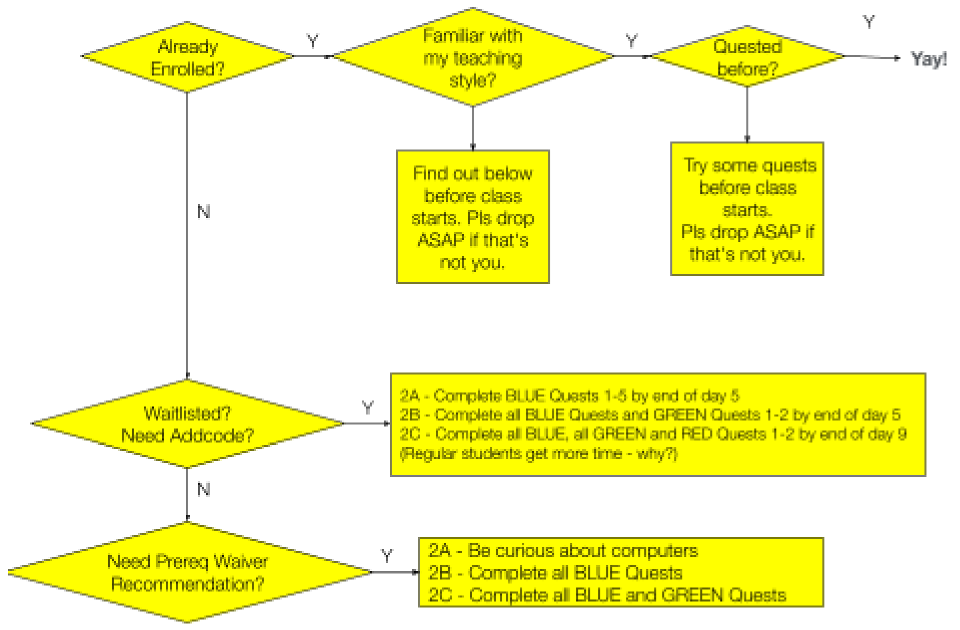
`“You cannot beast mode code your way through this class” - John Vicino`

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?


Already Enrolled?
Y
Familiar with my teaching style?
Y
Quested before?
Y
Yay!
N
Find out below before class starts. Pls drop ASAP if that's not you.
Try some quests before class starts. Pls drop ASAP if that's not you.
Waitlisted? Need Addcode?
Y
2A - Complete BLUE Quests 1-5 by end of day 5
2B - Complete all BLUE Quests and GREEN Quests 1-2 by end of day 5
2C - Complete all BLUE, all GREEN and RED Quests 1-2 by end of day 9
(Regular students get more time - why?)
N
Need Prereq Waiver Recommendation?
Y
2A - Be curious about computers
2B - Complete all BLUE Quests
2C - Complete all BLUE and GREEN Quests

Here is the way I approach teaching: I don't think I have ever been fond of stuffing knowledge down the throats of people who don't want it. But I believe that chances are high that if you've already sampled CS2A and CS2B and have come back for more, this is most likely your cup of tea and you want more of it.

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.

If you're just now venturing into real C++, and decided to dip your toes into CS2C, *be prepared to do a lot of self-learning*. If you think you are at risk of finding this class overwhelming, please consider restarting at the tiger asap and by solving each quest on your own.

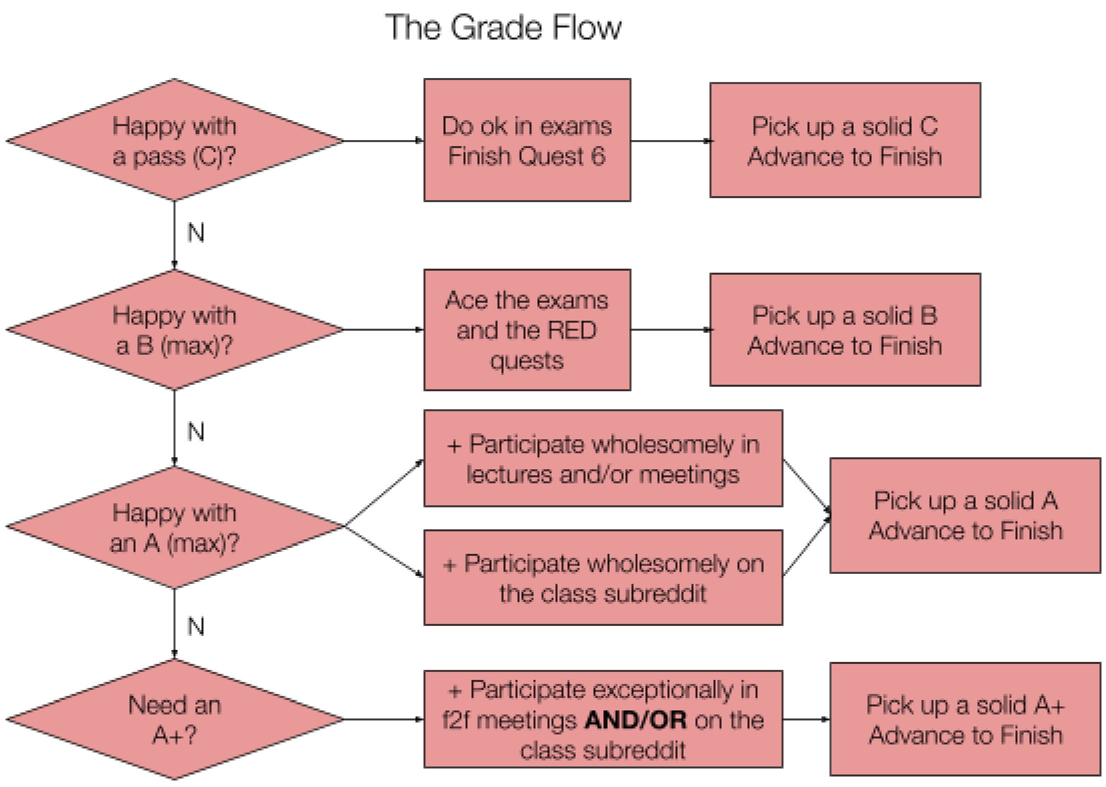
On the positive side, you CAN always try to get the help you need by posting in our moderated 2A or 2B subs in addition to our section's 2C sub. They're not restricted to active Foothill students. But use them to ask for help, not look up past solutions.

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)

In my classes, you will largely self-teach yourself most of the material with help from your classmates. Your tutors and I will be watching, careful not to step in and give you the answer. Yes - You will NOT be given the answer. You cannot ask anyone to debug your programs for you. You have to find it yourself, although we will give you all the help we think you need to succeed.

First things second - General Matters


The Grade Flow
Happy with a pass (C)?
Do ok in exams Finish Quest 6
Pick up a solid C Advance to Finish
N
Happy with a B (max)?
Ace the exams and the RED quests
Pick up a solid B Advance to Finish
N
Happy with an A (max)?
+ Participate wholesomely in lectures and/or meetings
+ Participate wholesomely on the class subreddit
Pick up a solid A Advance to Finish
N
Need an A+?
+ Participate exceptionally in f2f meetings AND/OR on the class subreddit
Pick up a solid A+ Advance to Finish

If you just want to be able to fill ordinary (resource-type) programming jobs that don't require heavy-duty knowledge of CS fundamentals or Data Structures, you can pass this class by completing Quests 1-6, and doing ok in the exams/quizzes. At the end of Quest 6, you should be intimately familiar with at least the hash table ADT, which will be behind many applications you may create (e.g. using a python dictionary).

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

To get to A Territory, you absolutely need to earn concept demonstration points. Even if you know all the concepts flawlessly, you will NOT get an A in this class without demonstrating an ability to explain your concepts and help those trying to learn 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.

CS 2C Readiness Self-Calibration

This is a **challenging** class according to most former students. How do you know if you will be successful? Try this:

In this class, there are two required technical tasks to test your preparedness. If you cannot complete them both in 10 days you will be dropped for your own benefit, so you can return when you are better prepared to succeed.

- 9 BLUE quests (essentially ALL the homework of CS2A) is due on day 3 (Friday of week 1). You only get 2 days to complete what CS2A students get 3 whole months for.
- 9 GREEN quests (essentially ALL the homework of CS2B) is due on day 10 (Friday of week 2). You only get 7 days to complete what CS2B students get 3 whole months for.

Even though you may already be proficient in these concepts or have passed CS2A and CS2B (or their equivalents) elsewhere, you still need to complete these quests. I allocated grade points to them commensurate with the amount of effort they should take CS2C beginners.

If it takes you longer than 2 days to complete all the BLUE quests, or longer than 7 days to complete all the GREEN quests, you should seriously reevaluate if you want to do CS2C this quarter or later or review the amount of effort you are willing to put in each day.

Important note: To keep your self-evaluation meaningful, you must **NOT** refer to ANY past-subreddit posts, nor search for solutions online. You may converse freely with any *current* quester in the appropriate subreddit. Each quest is designed to be solved using only the tools you should already be familiar with when you reach them (plus your own creative thinking). *This is the way the very first batch of questing students had it, and I don't believe that you are less capable than any of them.*

What does the data say?

Students who complete GREEN 9 by day 10 have an 67.5% chance of a A- grade or better

One by-product of this necessary exercise (if done sincerely) is that you will get a taste of the pace of work required for this class. You cannot return to the class from time to time to make progress. It has to become a central theme of your life for the remainder of this quarter and you must either be programming or thinking about programming at least 4+ hours each day. To make it more worthwhile, my participation points award system is tuned to reward predictable periodic forum behavior and non-bursty progress.

Section II - Administrative stuff

CS 2C continues where CS 2B left off. Students already comfortable with intermediate-level C++ programs will have an opportunity to play with and master important data-manipulation algorithms.

This course provides a systematic treatment of advanced data structures, algorithm analysis and abstract data types in the C++ programming language. Coding topics include building ADTs on top of the STL templates, vectors, lists, trees, maps, hashing functions and graphs. Concept topics include searching, big-O time complexity, analysis of major sorting techniques, top down splaying, AVL tree balancing, shortest path algorithms, minimum spanning trees and maximum flow graphs.

A strong 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 involving advanced data structures such as Lists, Trees, Graphs. They will be familiar with the use and implementation of algorithms for balancing binary trees, creating splay trees, minimum spanning graphs, finding the shortest path through a graph, and maximum flow through a network. They would also be familiar with the most common sorting algorithms and know the advantages and tradeoffs of each.
2. A successful student will be able to reason about the running time and derive properties of computer programs using precise mathematical terminology. Specifically they will be conversant with the Big-O notation and be able to craft efficient algorithms using the appropriate data structures to solve non-trivial computational problems.

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 - 9 BLUE Quests (**10** points)
- REQUIRED - 9 GREEN Quests (**20** points)
- 9 RED Mystery Quests you must solve starting Thu week 2, at the rate of about one per week - refer to the freeze schedule (total capped at **250** points for 2C this quarter)
- 1 Midterm on the Thu of W6 - (**25** points)
- 1 Final exam on the Thu of W12 - (**50** points)
- Struggles/Mastery points aka C-Score - See below (**50** points)

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**).

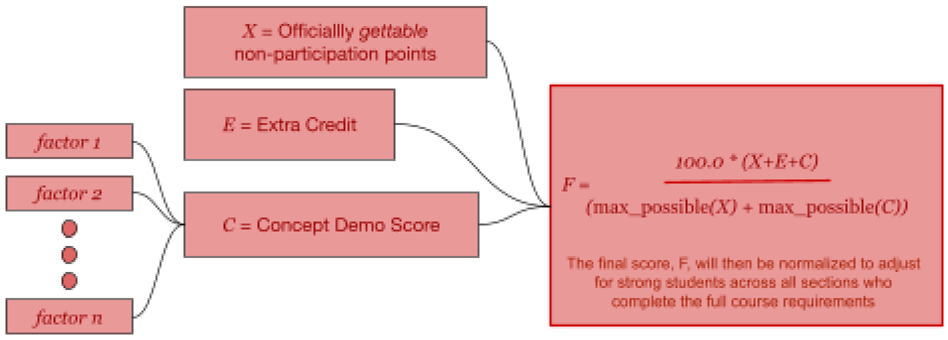

X = Officiallly gettable non-participation points
E = Extra Credit
factor 1
factor 2
factor n
C = Concept Demo Score
F = 100.0 * (X+E+C) / (max_possible(X) + max_possible(C))
The final score, F, will then be normalized to adjust for strong students across all sections who complete the full course requirements

Figure G: Formula for final grade

According to this syllabus, max_possible(X) should be **360** - Yes?

max_possible(C) = **50**. Find details in Section IV.

Virtual Catch-up Circles

Even though your section may be purely online, we will have **weekly check-in virtual meetings.** These meetings may last from anywhere between 1 to a couple of hours. During the first week of class, a time slot that accommodates the most number of students will be chosen for this meeting.

Consider this meeting the equivalent of lectures for those who are familiar with my online lectures. Professionals in the field taking this class out of interest may find these meetings similar to their Weekly Status meetings, except that ours will take place in a supportive, moderated environment in which we can point you to information or lessons that can help unblock you.

Interested to see what these meetings may be like?

See https://www.youtube.com/watch?v=qdHskvhqHH8&list=PL7PAwfkmE8mqnokMLBJaZAwJ6F9V9Poen

What if you cannot make these online catch-up circles via zoom? Then you must schedule private 1-1 meetings with me during which I'll get a chance to review your progress, give pointers on what to do next, and enter a suitable proxy for this score in my private spreadsheet.

Communication

Please use our sub for any question or comment that relates to the quests (except questions of a private nature). If you have a confidential question (grades or registration) you can email me. If you have a question that only makes sense of material you can find in Canvas (e.g. modules, syllabus, exams, etc.) then it makes sense to post that question in Canvas rather than our sub.

Try to meet with each other after class (even if virtually), set up private study and programming groups and work on independent (non assignment) programming challenges outside of class. I'll give you a few interesting challenges from time to time. Some of these may earn you extra credit.

You can reach me via messaging in Canvas, Reddit or by email. While on campus, my room number is `0x113d` (in hex). In week 1 of CS2A, you would have learned how to decode that into decimal.

Infinite Office Hours!!!

Well, within reason, ofc.

One time I happened to be in my office at 2AM. Someone knocked on my door.

> "Who is it?" I asked.
>
> "Quick question prof" said a female voice.
>
> I opened the door. "Hey sorry. My office hours are at 10am. See the sign?" I pointed at the print out I had stuck to my door. It clearly said 10AM - 11AM in **BIG BOLD BLACK** letters.
>
> "Yeah" she said. "I read it. But I'm a binary janitor. So I waited up until now to come here."
>
> "That's cool. You definitely deserve an answer" I said. "What's your question?"
>
> "Would you like me to come back later to empty your trash can?"

This quarter, I've decided to do away with this confusion. I have open office hours ANYTIME (virtual). I am very flexible in being able to adjust to your availability. But you have to pre-arrange it with me via email first.

I will likely be able to point you in the right direction for further experimentation. But I will not look at your questing code directly, nor debug it for you. Nor can you ask anyone else to do it (honor policy).

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 **250** for CS2C 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 2C (RED 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. Keep moving when you get a password. You can always come back to polish your previous quests when you have free time before the freeze date. You get about ONE week (7 days) before each quest freezes.

At the end of the quarter, your total trophy count will be capped at **250** 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. 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*).

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. You're best to finish them in a single sitting.

What does the data say?

Students who don't complete Quest 4 by week 6 have an 87.5% chance of a C-grade or worse

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. Below is one suggestion:

Week	**Read References**	**Complete**	**Notes**
1	Algorithms and ADTs review	All BLUE some GREEN	
	Algorithm analysis	WRAP UP GREEN	
3	Time complexity and Big-O	Mystery Quest 1	Q1 Freezes Mon 12am
	General trees (and BSTs)	Mystery Quest 2	Q2 Freezes Mon 12am
5	AVL Balancing and Splaying	Mystery Quest 3	Q3 Freezes Mon 12am
	Review/Midterm (Canvas)	Mystery Quest 4	Q4 Freezes Mon 12am
7	Hash tables Quadratic probing	Mystery Quest 5	Q5 Freezes Mon 12am
	Sorting	Mystery Quest 6	Q6 Freezes Mon 12am
9	Priority Queues, Heaps, Heapsort	Mystery Quest 7	Q7 Freezes Mon 12am
	Dijkstra's and Kruskal's algorithms	Mystery Quest 8	Q8 Freezes Mon 12am
11	A Maxflow algorithm	Mystery Quest 9	Q9 Freezes Mon 12am
	Final Exam (Canvas)		

You only get 1 week to complete each quest - hard-deadlines in CS2C

Every week, give yourself one or more topics to study and one or more programming quests to complete. Make sure you have adequate time to code, experiment, revise and recode. This takes a LOT longer than many students estimate. Also see: The bell rings at 11:59.

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).

Don't be an *unceremonious drop*

An unceremonious drop is when a student enrolls in a class, discovers they are out of their depth, and decides to silently exit the class without telling or thanking anyone. This helps no one, especially those in the waitlist who might have got into class if you had not enrolled.

Get a taste of how this challenging class will be by reading this syllabus and checking out the comments left by past students on our subreddit (They usually have titles like "Tips" or "Advice" for future students).

For most people, CS2A is NOT challenging even if you are not interested in computer science.

CS2B requires active interest in the subject and a desire to spend your spare time coding. All others will find CS2B moderately challenging.

CS2C requires more than an active interest in the subject. You must be passionate about computers, logical problem solving, critical thinking and programming, and be keen to learn as many new tricks as possible. CS2C will be easy for such students, but VERY challenging for all others, regardless of where and how long you have been programming in your life.

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 $\boldsymbol{C}$ 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.

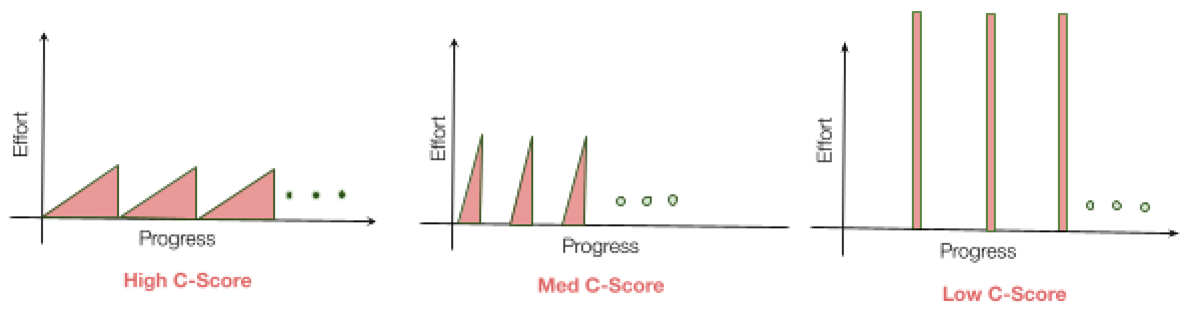
It does not matter if you're *grinding* in anonymous mode without a Student ID. The system can flag your code if it suspects grinding behavior or code similar to any code that was *ground by anyone in the past*. In that case, your concept demo points will take a serious hit and you may not know.

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 2 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.[1]

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.

[1] 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: *Data Structures and Algorithm Analysis in C++*, any edition ≥ 2nd, by Mark Allen Weiss, Pearson.

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 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.

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 CS2C, 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!

&