Comp Sci 214 — Data Structures — Fall 2024

1 Course Description

Comp Sci 214 teaches the design, implementation, analysis, and proper application of abstract data types, data structures, and their associated algorithms. We will explore a wide variety of data structures both conceptually and concretely via implementation. The class involves a significant hands-on programming component; you will both implement and use several data structures yourself.

2 Course Staff

Instructor	Prof. Vincent St-Amour (`stamourv@northwestern.edu`) (Call me Vincent)
Teaching Assistant	Tochukwu Eze (`TochukwuEze2025@u.northwestern.edu`)
Peer mentors	Adedamola Adejumobi, Alison Bai, Brock Brown, Nicholas Di Girolamo, Kaylie Mei, Didier Munezero, Ethan Piñeda, Joanna Soltys, Shreya Sridhar, Mattie Poelsterl, Sydney TerMolen, and Kris Yun

3 Prerequisites

- Comp Sci 111 AND (Comp Sci 211 OR Comp Sci 150)
- Comfort with programming.
- Basic discrete math. See Chapter 2 of the textbook.

4 Learning Objectives

- **Functional correctness**: can you apply data structures and algorithms concepts to produce implementations that display correct behavior and produce correct results?
- **Non-functional correctness**: can you produce programs that are efficient, robust, and maintainable, and demonstrate good programming habits and hygiene?
- **Theory**: can you understand, apply, and contextualize definitions and techniques related to the theoretical aspects of data structures and algorithms: computational complexity, invariants, etc.
- **Evaluation**: can you make informed decisions when picking data structures and algorithms and evaluating alternatives?
- **Integration**: can you combine multiple data structures and algorithms to decompose and solve large and complex problems?

5 Communication Channels

In this class, we will rely on a variety of communication channels, each serving a different purpose.

5.1 Lectures

Lectures will be synchronous and in person. **Attendance is mandatory: we expect you to come to lecture and participate.** Per registrar policy[1], excessive absence is grounds for failing a class.

This is consistent with my observations teaching thousands of students over a number of years: not coming to class has been the #1 cause of failing grades, and the vast majority of students who skip class end up doing poorly. I don't want this to happen to you. However, we do understand that life happens, and that coming to class may not always be possible for everyone.

As a compromise, attendance[2] to **least 50% of lectures** will be required to pass the class. Insufficient attendance will automatically result in a final grade of **F**. Because this is a very generous buffer, we will not distinguish between justified and unjustified absences: you should be nowhere near that threshold, so that a sudden mishap or illness would not push you over the edge.

To track attendance, we will be using the Youhere platform.

- Install the Youhere app on your phone: `https://www.youhere.org/app`
- In the app, enroll in: `nu-cs-214-f24-lecture`
- **Within 5 minutes** ot the start of each lecture, check in using the app **from inside the classroom**

Recordings

We will make a best effort attempt at recording lectures, and the recordings will be available from Canvas under the "Panopto" tab. Please see the recording policy at the end of this document. Recordings are **not** substitutes for attending lectures, merely supplements.

To ensure the health and safety of our community, however, *DO NOT* come to lecture if you have reason to believe you may be sick with COVID-19 (or anything else, really). Please watch the recordings instead and contact the instructor.

Questions

I love questions! Keep 'em coming. During lecture, please ask questions in one of two ways:

- Raise your hand and ask verbally.
- If you prefer to ask anonymously and/or textually, reply to the Piazza (see below) post of the current lecture. A member of the course staff will then ask your question in your stead.

5.2 Piazza

We will use Piazza for discussion, Q&A, and announcements. You can find the class's Piazza instance here: `https://piazza.com/northwestern/fall2024/comp_sci214`

We expect you to subscribe to the 214 Piazza board ASAP and monitor it regularly.

The entire course staff, as well as many of your fellow students, will be monitoring Piazza. As such, it is the most effective way to get answers quickly; more effective than contacting individual members of the course staff!

[1] `https://catalogs.northwestern.edu/undergraduate/requirements-policies/enrollment/exams-attendance/`

[2] By which we mean: coming to class on time and being present and engaged for the entire duration of the meeting.

A few guidelines for using Piazza productively:

- **Look before you post**: The main advantage of Piazza is that common questions are already answered, so search for an existing answer before asking a question. Posts are categorized (e.g., "hw1") to help you search.
- **Don't share your solutions**: Everyone in the class can see public posts. So if you post elements of your solution in a public post, you've just spoiled everyone's learning. Not good. If your question involves specifics of your solution, make a private post ("visible to Instructors only") to keep your hard work hidden from prying eyes.
- **Post publicly by default**: In all other cases, public posts are preferable: your fellow students can answer (so you get an answer faster!) and other students can benefit from the answer as well.
- **Ask precise questions**: Posting a big chunk of code and then saying "help, I don't get it" is not an effective way to learn. Instead, narrow down your problem to the point where you can ask a specific, precise question, e.g., "on the 3rd line I get this error, what does it mean?"

For questions that require longer discussions/explanations, please attend office hours (see below) instead. If all else fails or for sensitive matters, send the instructor an email, either for discussion or to schedule an appointment.

5.3 Canvas

We will use Canvas to distribute materials, assignments, and feedback. All (non-exam) assignments will also be submitted via Canvas.

We expect you to check Canvas regularly.

Please do not use Canvas messages/comments to communicate with us; they are very easy to miss. Please use Piazza instead.

5.4 Office Hours

We will hold a combination of online and in-person office hours, circumstances permitting. We will maintain an up-to-date schedule (with locations) on the "Office Hours" page on Canvas. We will monitor attendance patterns and adjust the schedule to the best of our ability to limit crowding.

Online office hours will be held using the `gather.town` platform. We will use classroom C in: `https://app.gather.town/app/ttbZrso3rxoiYapc/NU-CS-Office-Hours`

See the "Office Hours" page on Canvas for the password to the room.

To ensure we can help students in a first-come first-served order, and to help us group up people with similar questions, we ask that you sign up on this form when you arrive to office hours, or when you have a follow-up question: `https://forms.gle/1pC2AXdkLeaDpkw99`

You can look at the state of the queue to give you an idea of how close you are to being next in line: `https://tinyurl.com/2f4jar72`

Expectations

Office hours are a great place to get advice and assistance on all aspects of the class (and even CS in general!) In particular, they are not limited to debugging assistance; conceptual questions are welcome too!

Keep in mind, however, that our course staff is here to **help you learn**, *not* to give you solutions or to solve problems for you. An important part of the learning process is learning to figure things out and solve problems on your own; having members of the course staff do that for you would be doing you a disservice!

This means you should expect our course staff to give you advice and strategies or point you to relevant resources (e.g., documentation, slides, videos), but not to hold your hand (so to speak) while you work through problems. This will be especially true later in the quarter as assignments get more challenging and office hours get busier; to make sure everyone can get help, they won't have time to do that.

To get the most out of office hours, come prepared. Please come with specific questions or issues: i.e., not "I don't know what to do." or "Where do I start?". And be ready to tell us what you have tried to resolve them: e.g., you wrote some test cases to narrow down where a bug could be, you've added some printing to understand what your code was doing. This will make it much easier for our course staff to figure out what's going on, and they'll be grateful. On the other hand, **if you are not prepared, our course staff will refuse to help you**.

Ultimately, though, none of us are psychic: it's absolutely possible there may be an issue with your code which we cannot figure out. The person who is in the best position to understand it is the person who wrote the code and has been staring at it for hours: i.e., *you*. Our job is to give you the tools you need to get to that understanding.

5.5 Additional Help

Asking on Piazza and office hours should be your first steps for getting help. If you require additional one-on-one help, email the instructor. Do not ask individual members of the course staff for one-on-one appointments directly. To ensure an equitable workload distribution across staff members, we will manage such requests centrally.

6 Resources

Textbook

For some of the topics we will discuss, we will follow the draft of a textbook I am writing for this class (available on Canvas). The tentative schedule later in this document details how lectures and textbook chapters line up.

For the most part lectures are intended to be standalone, with the textbook as a supplement for students who want to reinforce their learning or go deeper into a topic. Some topics, however, will not be covered directly in lecture; you will therefore need to read the relevant textbook chapter to be able to understand what follows in class. These topics will be announced in advance.

In addition, if you are looking for a reference, Cormen, Leiserson, Rivest, and Stein's *Introduction to Algorithms* is the standard one. It is *very* comprehensive, but not always easy to approach. Its emphasis is also more theoretical than ours. Nonetheless, it is a useful book to have on hand for computer scientists.

Supplementary Videos

A variety of supplementary videos about various 214-related topics are available on Canvas (on the page of that name). These have been produced by (current and former) members of our course staff, and provide additional information and advice that we don't have time to cover in lectures. We recommend watching them early on; they offer valuable advice that can be quite helpful when working on assignments.

7 Assessment

This class uses a form of *Specifications Grading* (Nilson 2015). In particular, it features mainly qualitative assessment, and does not rely on points, percentages, and weighted averages to determine final grades.

This is likely quite different from what you have seen in your other classes, and will take some getting used to on your part. This system, however, has a number of benefits for students—see section 7.7.

7.1 Programming Assignments

This class features five programming homeworks and a final project that has a programming component, which chiefly contribute to our *functional correctness* learning objective. All programming assignments and related sub-assignments (see below) must be done individually; see our Academic Integrity policy later.

When evaluating your programming submissions, we will give you feedback regarding where they fell short as well as assign the following possible outcomes:

- **Got it**: your submission implements the specification correctly and in full.
- **Almost there**: your submission implements the specification with minor mistakes or omissions.
- **On the way**: your submission has elements of a correct solution, but requires non-trivial further work to be correct and/or complete.
- **Not yet**: your submission requires significant further work.
- **Cannot assess**: we could not evaluate your submission as is; repairs are required before we can do so.

Specifics of what is required to achieve each outcome are included in the handout for each assignment; our expectations are stated up front, no surprises.

Resubmissions

Making mistakes and correcting them is a natural part of the learning process. An outcome of **On the way** or **Not yet** is *NOT* a failure: it just means you still have things to learn from this assignment. Our assignments are *intentionally* challenging and our grading is *intentionally* strict in order to maximize your learning; it's perfectly normal to not get them entirely correct on the first try.

To give you the opportunity to incorporate our feedback and get credit for your improved learning, we will allow you to resubmit the code portion of each homework one week after the initial deadline.[3] To determine final grades, we will use the best of the two outcomes your submissions have achieved.

Given the scale and importance of the project, we will give you *two* opportunities to resubmit it instead of just one, and consider the best of the three outcomes.

Resubmissions are optional: if you're happy with your first submission, you can skip the second.

Self-Evaluations

Each of the five programming homeworks will be followed by a self-evaluation, due one week after the initial deadline. Self-evaluations will consist of a few questions about the code you turned in.[4]

The purpose of these self-evaluations is to evaluate your submissions on *non-functional correctness*: desirable aspects of programs that go beyond strictly producing correct answers. Aspects like efficiency, testing, robustness, or factoring. In actual programming (i.e., not in classes), these are *just as important* (if not *more so*) than raw functional correctness. This class will give you opportunities to cultivate these skills and habits as you work on assignments.

Project Design Documents

Instead of a self-evaluation, your final project will be accompanied by three design documents to explain your design and provide rationales for your decisions. More details will be included in the final project handout.

The final project and its associated design documents contribute to our *evaluation* and *integration* learning objectives.

[3]See schedule at the end for exact dates.

[4]Specifically, the code you turned in for the *first* submission, *not* the resubmission, and *not* work in progress towards the resubmission. You can download your original submission from Canvas if you overwrote/misplaced it.

7.2 Worksheets

To evaluate specific aspects of our *theory* learning objective, we will have two worksheet-style assignments.

To ensure everyone has a solid theoretical foundation, these worksheets will be done as Canvas quizzes with retries; we strongly encourage you to re-do them if necessary to make sure you fully understand that material.

7.3 Exams

We will have two in-person, on-paper midterm exams which will chiefly assess our *theory* learning objective. No notes or electronics (laptops, calculators, tablets, phones, smart watches, tamagotchis, etc.) will be allowed during exams. Specifics and logistical details will be announced leading to each exam.

7.4 Final Grades

Final letter grades will be determined as follows.

Base Grade

Your *base grade* will be determined based on your performance on homeworks and the final project:

		A- Great	**B** Good	**C+** Fine	**D** Poor
Homeworks	# **Got it**	all 5	3	2	1
	# **Almost there** (or better)	—	1	1	1
	# **On the way** (or better)	—	1	1	1
	# not counted	—	—	1	2
Final project	Minimum outcome	**Got it**	**Almost there**	**On the way**	**On the way**

The criteria in each column are the minimum criteria for that base grade. Note that the counts within a column should all refer to distinct homework assignments; for example, the one **Almost there** or better homework for a **B** base grade is in addition to the three **Got it** ones. To determine your base grade, check your homework and project outcomes against the first column. If at least one criteria in the column does not apply, move on to the next column. Keep moving until you reach a column for which your outcomes apply: this is your base grade. Failing to meet the criteria for a base grade of **D** will result in automatically failing the class (i.e., an **F**).

For example, a student whose homework submissions earned 4 **Got it** but one **On the way** and whose project submissions earned an **Almost there** would earn a base grade of **B**. Modifiers (see below) may adjust this base grade further. As another example, a student whose homework submissions earned 2 **Got it**, 2 **On the way**, 1 **Not yet**, and whose project submissions earned an **Almost there** would earn a base grade of **D**.

Warning: these criteria put a lot of importance on the final project, which is intentional. We believe[5] that the learning experiences from the project are especially valuable. We therefore consider it very important, and we want you to do so as well.

Modifiers

This base grade will then get adjusted with *modifiers* based on your performance in other aspects of the class. Each of these can either increase your grade by some number of partial letter grades (*positive* modifier), leave your grade unchanged (*neutral* modifier), or decrease your grade by some number of partial letter grades (*negative* modifier). See the tables below for the possible modifiers in this class.

[5] And past students agree!

Expectations: good work which meets our expectations will be recognized with a neutral modifier. Positive modifiers are reserved for exceptional work; these are the exception, not the norm. Work that does not meet our expectations will receive negative modifiers; interpret those as an invitation to adjust your approach for this part of the class.

		+1 Outstanding	**0** Good	**-1** Poor
Worksheets		Both 100%	Both ⩾80%	Either <80%
Self-evaluations	# 5/5s	⩾4	—	—
	# ⩾ 3/5s	—	⩾3	⩽ 2
	# ⩽ 2/5s	⩽ 1	⩽ 2	⩾ 3

Both worksheets are considered together to determine a single modifier; both must meet the expected level. Similarly, all five self-evaluations are considered together to determine the self-evaluation modifier. Self-evaluations will consist of five questions, and each one will be counted towards the modifiers based on how many of the five questions you got correct (all 5, 3-4, 2 or fewer).

	+1 Outstanding	**0** Good	**-1** Poor	**-2** Warning	**-3** Danger!	**-4**	**-5**
Exam #1	18-20/20	13-17	10-12	8-9	6-7	4-5	0-3
Exam #2	18-20/20	13-17	10-12	8-9	6-7	4-5	0-3
Exam improvement	#2 mod. ⩾ #1 mod. + 2						

Each of our two exams will be out of 20 points. Depending on the score you earn on an exam, its modifier will range from **+1** (an increase of one partial letter grade) to **-5** (a decrease of one full letter grade plus two thirds), with the plurality of scores leading to a neutral modifier (no change).

In addition to each exam providing its own modifier, if your performance improves significantly between the two exams, you will earn a positive (**+1**) modifier. Here, "significantly" means the second exam is two modifier steps (or more) above the first one: e.g., going from a **-1** to a **+1**, or from a **-3** to a **0**.

	+2	**+1**	**0**	**-1**
Project program	2x Above expectations*	Above expectations**	Meets expectations	
Project documents		Excellent	Minor errors or omissions	Major errors or omissions

* Base grade requires **On the way**, earned **Got it**.

** Base grade requires **On the way**, earned **Almost there**. Or requires **Almost there**, earned **Got it**.

Finally, the final project contributes two modifiers: one for your design documents, and one for exceeding the expected outcome set by your base grade.

For example, suppose our first hypothetical student from before who earned a base grade of **B** got everything right on both worksheets (**+1**), submitted self-evaluations that earned three 5/5s and two 2/5s (**0**), did poorly on exam 1 (**-1**) but well on exam 2 (**0**), and wrote excellent project documents (**+1**). This would work out to an increase of one partial letter grade, for a final grade of **B+**.[6]

For another example, suppose our other hypothetical student who earned a base grade of **D** did poorly on exam 1 (**-1**) and great on exam 2 (**+1** on its own, and **+1** for improvement), earned an **Almost there** on the project (above expectations for a **D** base grade, so **+1**) while having **+1** for the worksheets, **+1** for the project design documents, and **0** for self-evaluations, this would result in a total of **+4**. Starting from a base grade of **D**, our hypothetical student would earn a final grade of **B-**.[7]

[6]Good job, hypothetical student!

[7]Great recovery, other hypothetical student!

For reference, the possible letter grades at Northwestern are: **F**, **D**, **C-**, **C**, **C+**, **B-**, **B**, **B+**, **A-**, and **A**. Going up one partial letter grade means going forward one step in this sequence, and going down one letter grade means going backwards one step.

The letter grade sequence "saturates" on either end: applying **+4** to a base of **B** results in an **A** (there is no **A+**), and applying **-2** to a base of **D** results in an **F** (there are no **D-**, **F+**, or **F-**). And as a reminder, failing to meet a base grade of **D** results in a final grade of **F**, regardless of modifiers.

The attendance requirement described in section 5.1 takes precedence over everything: failure to meet the attendance requirement results in a final grade of **F**, regardless of base grade or modifiers.

7.5 Late Policy

Unless otherwise indicated assignments are due by 11:59pm on their due date.

To accommodate everyday slippage and minor life hiccups, each student starts the class with three *late tokens*. Each of these late tokens can be exchanged for a two-day *no questions asked* extension to any (non-exam) deadline, covering all assignments, resubmissions, etc. due that day. Only one token can be used per deadline; i.e., two days is the maximum extension possible. Late tokens will be used automatically when you submit late; you don't need to reach out to us and ask for permission.

In the event a student runs out of tokens, I'll be in touch. Being occasionally late is perfectly fine. Being consistently late, on the other hand, may be a sign of deeper underlying issues: let's talk so we can figure out the best way to help you.

To make deadlines and accommodations equitable for all students, we will not grant further ad-hoc extensions or accomodations, barring extreme circumstances. If you do end up in a situation that would warrant additional flexibility, you must contact your dean of students[8] and have them contact me. They will help you coordinate extensions and accomodations across all your classes, which will ensure you get the support you need across the board, not just in this class.

If you require accommodations for medical reasons, you should follow the process described here: `https://tinyurl.com/d84bvvzj`

7.6 Redo and Regrade Policy

Mistakes can happen (on your part) when submitting or (on our part) when grading assignments. To accomodate for that, each student starts the class with three *redo tokens* which can be exchanged either for a *redo* or a *regrade*. Details follow.

Redos

In the event a student misunderstands instructions or makes a minor mistake, they can spend a redo token to submit a corrected version of either a homework self-evaluation or a project design document.[9] Other kinds of assignments (e.g., homeworks, worksheets, or exams) cannot be redone.

Redos for an assignment are due *one week after the initial deadline* or two days after feedback is released, whichever is later. We will aim to grade redos within a week after that.

Self-evaluation redos must, just like initial submissions, be about the first submission of the relevant homework; the redo is solely about the answer to the self-evaluation questions themselves.

Because manual action on our part is required to process these redos, students wishing to spend a redo token must fill out this form: `https://forms.gle/SS8kunXNm8GRKMu8A`

Out-of-band requests coming in via email, piazza, or in person will be redirected to the form and/or ignored. Once a student runs out of tokens, no further redos will be accepted from them. Grading of redos are final; there will be no additional opportunities to resubmit.

[8]Dean Joe Holtgreive (`jjh@northwestern.edu`) for McCormick students, Dean Christine McCary (`christine.mccary@northwestern.edu`) for Weinberg students, and others for students in other schools.

[9]Except the third and final design document; its deadline is already as close to the end of the quarter as it can be.

Regrades

If a student believes we have made a mistake when grading either (1) one of their self-evaluations, (2) one of their design documents, or (3) one question on one of their exams, they can spend one of their redo tokens to request we take another look.

Bear in mind, grading mistakes are limited to (1) clerical errors on our part, or (2) failing to grant credit for an answer that *we* consider correct. Disagreeing with a decision we made, or having a different opinion regarding what is correct does not constitute a grading mistake.

Like redos, regrade requests are due *one week after the initial deadline* or two days after feedback is release, whichever is later. Regrade requests must also be submitted through the form linked above; out-of-band requests will similarly be redirected/ignored.

Decisions we make regarding regrades (or lack thereof) are final.

7.7 Why this system?

This assessment system carries many benefits to you as a student.

- **Learning and grades align.** Learning is our true goal here; grades are (at best) a distraction. But given the impact grades can (unfortunately) have beyond the classroom, ignoring them altogether may not be an option for everyone. As a compromise, this system ties grades directly to concrete achievements and milestones in your learning, rather than on an amorphous pool of fungible points of assorted provenances.

- **You are in control.** You decide what grade you want to aim for, and the expectations you'll need to meet for that goal are stated up front and transparently. Grades are determined solely by your achievements, not by those of your classmates; no percentage cutoffs, curves, or other such nonsense. No more "hoping" for a grade, and no more surprises when letter grades come: you can always tell exactly how things are going, and know what you need to do if you want to change it.

- **High standards, low stakes.** You're all super smart. Yes, that means you too. You all have the potential to learn a *lot* in this class and grow as computer scientists. To inspire you to learn as much as you can, we will hold you to high standards; one doesn't learn much by consistently producing mediocre work. At the same time, we recognize that high standards can carry a lot of pressure; which is why we're also including mechanisms to lower stakes whenever possible (see next item).

- **There is room for mistakes.** Making mistakes is part of the learning process, and you should get credit for learning from them. Resubmissions give you an opportunity to do so. And for places where resubmissions don't make sense (e.g., exams), some slack is built-in to the grading standards to allow for small slips, and the consequences of even larger slips are bounded.

- **Flexibility built-in.** Life happens; we recognize that. But not all students feel comfortable asking for flexibility when necessary, either as a matter of personality or of upbringing. For this reason, "hiding" flexibility behind explicit requests is not equitable. Instead, this system comes with built-in flexibility in a number of places—resubmissions, late tokens, many possible modifiers, etc.—all of them with *no questions asked* and *no need to ask for it*. That way everyone benefits, regardless of comfort level.

Caveats

That being said, this is a large class–which introduces issues of scale and equity—and we're limited by the length of the term—which introduces issues of scheduling. This means we've unfortunately had to make some compromises when designing our assessment strategy.

In an ideal world, we would assess your learning directly using mechanisms like oral exams and code walkthroughs, with infinite retries and no hard deadlines. But we are not in an ideal world, so we're forced to assess your learning indirectly through your work using automated testing, written exams, and other mechanisms geared towards efficiency. These mechanisms, while sadly necessary in our context, may not reflect your learning 100% accurately.

Therefore, it is your responsibility to make sure the work you submit is as close a reflection of your actual learning as possible, so we can get the best picture we can of where you're at as a student. And in turn, so you can get as much recognition as possible for your learning. Concretely, this means:

- For programming assignments, being thorough in your testing to avoid silly mistakes, making sure you read our instructions closely, and taking full advantage of our feedback when working on resubmissions. You have ample time for each assignment; you can afford to be careful.
- For "one-shot" assessments like exams, be very careful to read instructions completely and thoroughly, and ask for clarifications if need be. We can't assess your learning if you answer the wrong question.

8 Software

Code examples and programming assignments will use the DSSL2 (Data Structures Student Language version 2) language. It runs on top of the Racket environment. You will need to install version 8.14 from:

`https://download.racket-lang.org`

If you have an old version from a previous course, some things will not work.

Then, to install DSSL2 proper, from DrRacket open the *File* menu and select *Install Package...*, then type `dssl2` as the source. Then click *Install*. When it's done, the *Install* button will change to *Update*, indicating that the package is installed. To familiarize yourself with the language, see the DSSL2 reference:

`https://docs.racket-lang.org/dssl2/`

8.1 Why DSSL2?

We designed DSSL2 as a language specifically for learning data structures. It features everything we need to build and reason about data structures and related concepts, while at the same time avoiding distractions and pitfalls from other languages that, while very useful in practice, are not relevant to our goals.

DSSL2 is a close cousin of Python, so if you know Python, you will be able to pick up DSSL2 easily and vice versa. We specifically do not use Python (or other production-ready languages) because not only do they have the aforementioned distractions, they also include a whole cornucopia of useful data structures already! This makes perfect sense for large-scale programming in practice, but is not appropriate for learning data structures; we want to be building those ourselves!

Note also that this class is *not* a programming class. It is a *computer science* class, where learning a specific language is not one of our learning objectives. The concepts and programs you see in this class are fundamental and universal: they apply to whatever language you choose, now or later in your career.

8.2 Development Environment

The DrRacket IDE (which comes with Racket, and which you may have used in 111) has the most complete DSSL2 integration of any environment; that's the one I encourage you to use. If you're already familiar with other environments, varying levels of DSSL2 support are also available for:

- **VSCode**: Joshua Irvin wrote a DSSL2 plugin for VSCode.[10] I have not tried it, though, so *caveat emptor*.
- **Vim**: Vim users can try Marko Vejnovich's DSSL2 plugin.[11] I have not used it myself either.
- **Emacs**: Emacs's Python mode offers adequate, but far from ideal DSSL2 support.

You're welcome to use these alternative environments if you wish, but neither I nor the course staff will be able to offer you support. You'll be on your own.

[10]`https://github.com/Gamefreak130/dssl2_vscode_extension`
[11]`https://github.com/markovejnovic/vim-dssl2/`

8.3 Hardware

By default, we assume that you will be using your personal computer to work on assignments. If need be, you should also have physical access to the Wilkinson Lab (Tech M338), as well as login access to the computers there.

If you cannot access these computers, please reset your password at:

`https://selfserv.eecs.northwestern.edu/temp_password/`

If that does not work, please contact `root@eecs.northwestern.edu`. Similarly, contact root if you do not have physical access to the labs.

9 Academic Integrity

The trust employers, colleagues, and the public put in a Northwestern degree is built on the premise that our grades are an accurate reflection of student learning and effort. Any attempt at subverting the relationship between learning and grades is a threat to this hard-earned trust and, left unaddressed, hurts all Northwestern students and alumni. As such, we take any attempts at violating academic integrity very seriously.

In this class, we expect all students to know, understand, and abide by the McCormick Academic Integrity policy,[12] as well as the course-specific policies below. Failure to abide by these policies is a serious offense which carries serious consequences.

Fundamentally, anything you turn in for this class must be your own individual work, and be the product of your own thinking and reflection. There are two main ways in which your work may fall short of this expectation: accessing unauthorized sources and engaging in excessive collaboration.

9.1 Unauthorized Sources

Sources which you are *allowed* to use while working on assignments:

- Materials posted to *this course's* Canvas or Piazza *this quarter*.
- Interactions with members of the course staff, be they in office hours, on Piazza, etc.
- Feedback *you* receive from us on *your* work, for example in homework grading reports.
- That's it.

In contrast, sources which you are *not allowed* to use or even consult while taking this class:

- Submissions, feedback, or work in progress from other students, past or present, in full or in part. This includes test cases; those are part of submissions.
- Solutions or solution fragments you may find online or elsewhere.
- Course materials from previous quarters, regardless of how you obtained them.
- Outside tutors or "work-for-hire" services. This includes AI-based "tutor" systems.
- Code/text/etc. generation tools (AI-based or otherwise), such as Github Copilot, ChatGPT, etc. This applies to *any* use, not just generation of artifacts you turn in.
- This list is not exhaustive; ask the instructor before using any source not explicitly listed above.

[12] `www.mccormick.northwestern.edu/students/undergraduate/academic-integrity.html`

Using, or even having access to, unauthorized sources constitutes a violation of this class's academic integrity policy, regardless of the extent to which the source was actually used, or even if it was used at all.

Similarly sharing, posting, uploading, or otherwise making available your own solutions (in full or in part) or any course materials (homeworks, exams, solutions, test cases, etc.) is also a violation of our academic integrity policy. This extends even after the quarter ends; course material remains private information which you may not share or reproduce.

A note for students retaking the class: resubmitting or otherwise reusing your work from earlier attempts at the class is **also** considered plagiarism. You should delete any old work you may have, and you must do all work *from scratch*; otherwise you will not learn as much as you should.

9.2 Excessive Collaboration

Collaboration is a good thing, and something you'll be expected to do in the workplace. However, because this is an introductory class, we also need to ensure that *every* student has a solid foundation, which will allow them to be effective and productive collaborators throughout their career. For this reason, students must tread very carefully when collaborating in this class, to ensure they do not (accidentally or willfully) cross the line into excessive collaboration, with all the negative effects on learning this implies.

When working with current or former students in this class, you are limited to *arm's-length collaboration*. When collaborating at arm's length:

- You **may not** read, write, look at, record, or in any way transcribe solution elements from someone else. In turn, your collaborator **may not** read, dictate, or otherwise share elements from their own (or anyone else's) solution with you, nor can they directly read, write, or look at your own solution either.
- You **may not** have your own solution on your screen or otherwise in your field of vision while collaborating, and neither must your collaborator.
- You **must** cite any collaborators whose ideas affected your work, and explain the effect the collaboration had on your solution. Bear in mind, however, that citing a collaboration does not absolve you from the aforementioned limits on arm's-length collaboration.

To avoid getting close to the line, we *strongly* recommend you keep any discussions with current or former students either at a conceptual level, or about non-graded aspects of the class (e.g., examples from course materials).

You are of course always welcome to seek help from any member of the course staff on assignments; the above limitations do not apply to such interactions.

As always when in doubt, ask the instructor before doing something you're not sure about.

9.3 Policy Circumvention

This class has a number of policies in place—detailed in this document—to both assist your learning and ensure the integrity of our learning environment. Attempting to circumvent course policy, or assisting someone else in doing so, compromises these goals. It is therefore also considered an academic integrity violation, and will be reported and penalized accordingly.

9.4 Suspected Violations

Any suspected violation will be reported to the McCormick Office of the Dean. The Dean will conduct an investigation to determine whether a violation did take place or not, which will involve meeting with you. Should you find yourself in this situation, *be forthright and honest*; lying or withholding information from the Dean is a further offense, and can make a bad situation much worse.

10 Welcoming Environment

I consider this classroom to be a place where you will be treated with respect. I welcome ***all*** students, and expect all of you to do the same. Together, we can create an environment where everyone feels welcome and can engage fully in our community.

Each student has something of value to contribute, especially in engineering disciplines where empathy, communication, and teamwork elevate our contributions to society; and lack thereof can lead to disaster. Individual differences can deepen our understanding of one another, the world around us, and our lifelong role as engineers.

(Credit: Adapted from statements by the ASEE and Prof. Emma DeCosta)

11 Northwestern University Syllabus Standards

This course follows the Northwestern University Syllabus Standards.[13] Students are responsible for familiarizing themselves with this information.

[13] https://www.registrar.northwestern.edu/registration-graduation/northwestern-university-syllabus-standards.html

12 Tentative Schedule

Dates and content are subject to change. Any changes will be announced on Piazza.

Date	Homeworks	Self-Evals and Resubmissions	Lecture Topic (and Reading)
Tue 9/24	1 out		Intro, DSSL2 Basics
			(Chapter 2)
Thu 9/26			DSSL2 Q&A, Linked Lists
Tue 10/1	1 due, 2 out		Abstract Data Types, Stacks, Queues
Thu 10/3	worksheets out		Asymptotic Complexity
Fri 10/4		1 out	
Tue 10/8	2 due	1 due	Sorting
Thu 10/10			Dictionary ADT
			(Chapters 6 and 7)
Fri 10/11		2 out	
Tue 10/15	worksheets due, 3 out	2 due	Hash Tables
			(Chapters 8 and 9)
Thu 10/17			Graph ADTs and Representations
			(Chapters 10, 11, and 12)
Tue 10/22			**First Midterm Exam**
Thu 10/24	4 out		Graph Search
			(Chapter 13)
Tue 10/29	3 due, 5 out		Priority Queue ADT
			(Chapter 15.2-15.5)
Thu 10/31			Single-Source Shortest Paths
			(Chapters 14 and 15.1)
Fri 11/1		3 out	
Tue 11/5	4 due	3 due	Data Design
Thu 11/7	project out		Data Design, continued
Fri 11/8		4 out	
Tue 11/12	5 due	4 due	Minimum Spanning Trees, Disjoint Sets ADT
Thu 11/14			Self-Balancing Trees
Fri 11/15		5 out	
Tue 11/19	1st project deadline	5 due	Amortized Analysis
Thu 11/21			Relational Model
Tue 11/26	2nd project deadline		Probabilistic Data Structures
Thu 11/28			*Thanksgiving: no class*
Tue 12/3			Persistent Data Structures
Thu 12/5			**Second Midterm Exam**
Tue 12/10	3rd project deadline		