COMP_SCI 214 — Data Structures — Winter 2026

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, concretely, and in context.

2 Course Staff

Instructor	Prof. Sruti Bhagavatula (`srutib@northwestern.edu`)
Teaching Assistants	Tochukwu Eze
Peer mentors	Alison Bai, Amaia Garnett, Jonas Goldberg, Hayley McCormack, Omotayo Oludemi, Henron Ruan, Daniel Solomon, Burke Stanton, Maya Wassercug, Daisy Zhang, Richard Zhang, Sharon Zheng

2.1 Contacting Us

Contact members of course staff for non-content-related inquiries through Piazza (explained later in the syllabus) NOT via email so we don't miss your message! To do this, you can share a post privately with "Instructors" (includes me and TAs/PMs) or specific members of course staff. You may also share a post with just me if the content is of a sensitive nature or you just want to reach me. **My email address is provided here for use in emergencies, otherwise, expect delays in responses.** We will do our best to respond to direct messages or inquiries within two weekdays.

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

- **Implementation**: can you apply data structures and algorithms concepts to produce implementations that display correct behavior and produce correct results?

  Assessment: programming homeworks

- **Integration**: can you combine multiple data structures and algorithms to decompose and solve large and complex problems?

  Assessment: final project

- **Theory**: can you understand, apply, and contextualize definitions and techniques related to the theoretical aspects of data structures and algorithms: computational complexity, invariants, etc.

  Assessment: worksheets, exams

- **Design**: can you produce programs that are efficient, robust, and maintainable, and demonstrate good programming habits and hygiene? And can you make informed decisions when picking data structures and algorithms and evaluating alternatives?

  Assessment: self-evaluations, mutation testing, design documents

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. Students are also expected to arrive in time for the start of the quarter and remain until the end.[2]

This is consistent with my observations over a number of years: not coming to class and not engaging 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. Attendance and engagement will be recorded measured via in-class exercises, described below.

In-Class Exercises

We will have daily in-class exercises whose purpose to (1) record your attendance and (2) actively engage you with the material and support your learning process. You must receive credit for **at least 50% of classes to pass the class.** Anything less than this 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 and exercises cannot be made up: you should be nowhere near that threshold, so that a sudden mishap or illness would not push you over the edge.

Exercises will often contain open-ended questions that require active engagement with and attention to the content in class that day. Since the exercises will directly be related to content, they can happen at any time during class. You will complete these exercises using "Poll Everywhere"; you can find instructions for how to use this tool on the Canvas homepage.

Receiving credit: If your answer (1) demonstrates engagement and legitimate effort and (2) is correct to a degree that indicates satisfactory engagement and learning, and (3) shows that you are checked into the class's location, you will get a score of 1 out of 1. An insufficient answer based on the above criteria gets a score of 0, as do missed exercises/classes. You are encouraged to discuss these exercises with your neighbors and need to complete these activities synchronously during class time only.

I will aim to have in-class exercises for a given week graded by the start of the following week.

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

[2]`https://www.northwestern.edu/provost/policies-procedures/classwork-curricular-policies/in-person-arrival.html`

Recordings

We will make a best effort attempt at recording lectures, and the recordings will be available from Canvas under the "Panopto" tab (often a few hours after class ends). 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 and contagious. Please watch the recordings instead.

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 (or answer you directly on Piazza for certain types of questions). If I cannot get to a question during lecture, I will answer you on Piazza after class.

5.2 Piazza

We will use Piazza for discussion, Q&A, and announcements. You can access the class's Piazza board through the "Piazza" link at the top of the Canvas page and on the left-side menu in Canvas.

We expect you to monitor the class's Piazza board regularly.

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

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 (code, word descriptions, brainstorming approaches, etc.) 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 a private Piazza note (or email in emergencies), 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, office hours guidelines you must adhere to, and instructions for getting help.

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.

Before attending office hours, you must read and understand the office hours guidelines linked at the top of the Canvas homepage.

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, send the instructor a private Piazza note. 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 reference the draft of a textbook written by Prof. St-Amour (available on Canvas). The schedule on Canvas details how lectures and textbook chapters line up.

For the most part lectures are intended to be standalone, with the textbook as a supplement to reinforce their learning or for students who want to 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 Materials

A variety of supplementary videos and documents about various 214-related topics are available on Canvas on the **Supplementary Materials** page. 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) which ties final grades to concrete levels of achievement towards our learning objectives. In particular, it features mainly coarse-grained assessment, and does not rely on percentages and weighted averages to determine final grades. This is likely different from what you have seen in your other classes and will take some getting used to, but it comes with a number of benefits for students—see section 7.9.

Final grades will be determined by combining grades from four separate *tracks*, each one corresponding to one of the class's learning objectives: *implementation*, *integration*, *theory*, and *design*. Each track has its own independent letter grade (**A**, **B**, **C**, **D**, **F**; no plusses or minuses) determined using the rules in the following sections. The rules for combining track grades into final grades can be found in section 7.5.

7.1 Implementation Track

We will assess the *implementation* learning objective through **five** programming homework assignments.

For each assignment, the functional correctness of submissions will be checked with **four** separate *test suites*, each one covering part of the homework's specification. Each test suite includes a number of individual test cases, and a test suite passes when *all* of its test cases pass.

- Each passing test suite awards one *implementation point*.
- Test suites with even a single test failure are worth **no** implementation points.
- Submissions which we cannot successfully run or which are otherwise unacceptable will earn **no** implementation points. The full details are in the assignment handouts.

To determine grades for the implementation track, add up implementation points across all five homeworks and consult the following table:

Points	20	16–19	12–15	10–11	0–9
Grade	**A**	**B**	**C**	**D**	**F**

For example, a student whose submissions passed all four test suites on all five homeworks—and thus ends the class with 20 implementation points—would earn an *implementation grade* of **A**; good job!

A student whose submissions passed a total of 14 test suites would earn an implementation grade of **C**: 14 is enough to reach the threshold for **C**, but not the one for **B**.

Students whose submissions pass fewer than 10 test suites total will earn an implementation grade of **F**.

Learning from your Mistakes

Making mistakes and correcting them is a natural part of the learning process. Not passing a test suite is *NOT* a defeat: 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.

When evaluating your programming submissions, we will give you feedback regarding where they fell short; precisely so you know what you still need to work on. To give you the opportunity to incorporate our feedback and get credit for your improved learning, you will have the option to resubmit a corrected version of each homework one week after its initial deadline.[3]

When counting implementation points, we will count the *highest* number of test suites either submission has passed.[4] Resubmissions are optional: if you're happy with your first submission, no need to resubmit.

7.2 Integration Track

We will assess the *integration* learning objective with the programming portion of the final project; other parts of the project will count towards the *design* track.

This track functions in much the same way as the *implementation* track does, with one difference: the project, being significantly more complex than earlier assignments, will be checked using **six** test suites, each worth one *integration point*.

To determine grades for the integration track, consult the following table:

Points	6	4–5	3	2	0–1
Grade	**A**	**B**	**C**	**D**	**F**

Learning from your Mistakes

The resubmission mechanism described above for homework assignments applies to the final project as well, with one difference. Given the scale and importance of the project, we will give you *two* opportunities to resubmit it instead of just one, for a total of three submissions.

7.3 Theory Track

We will assess the *theory* learning objective with **four** worksheets and **two** midterm exams.

Worksheets

The purpose of the worksheets is to build a solid foundation on (a subset of) the theoretical topics in the class ahead of exams. Worksheets are set up as Canvas quizzes with retries; we strongly encourage you to re-do them if necessary to make sure you fully understand that material.

Each *fully correct* worksheet is worth one *theory point*. This is a high bar, but with retries we expect every student to get all of them entirely correct. The theory track is calibrated accordingly, so don't neglect them!

Beware, though: worksheets alone are not preparation enough for the exams. There is a number of topics they do not cover which may be on the exam, and their coverage of the topics they do cover is not exhaustive either.

[3]See schedule on Canvas for exact dates.

[4]Do note, this is about the *number* of test suites only. If a first submission passed three test suites and failed one, and the second also passes three and fails a different one instead, this counts as three.

Exams

We will have two in-person, on-paper exams. No notes or electronics (laptops, calculators, tablets, phones, smart watches, headphones, etc.) will be allowed during exams. Specifics and logistical details will be announced leading to each exam.

Each exam awards a number of *theory points* depending on score:

Exam Score (x)	$85\% \leqslant x$	$70\% \leqslant x < 85\%$	$55\% \leqslant x < 70\%$	$35\% \leqslant x < 55\%$	$x < 35\%$
Theory Points	4	3	2	1	0

Due to the nature of exams, exam grades are inevitably "noisy". The use of ranges here "smoothes out" that noise while retaining significant distinctions: the difference between a 66% and a 68% is not meaningful, so it's not worth stressing over. The difference between a 66% and a 96%, however, almost certainly reflects a significant difference in understanding; it makes sense to recognize those differently.

Theory Grade

To determine grades for the theory track, add up theory points across worksheets and exams (for a maximum of 13) and consult the following table:

Points	12	10–11	8–9	7	0–6
Grade	**A**	**B**	**C**	**D**	**F**

For example, a student who earned theory points for all four worsheets and whose two exams earned 86% and 90% would have earned 4+4+4=12 theory points. This meets the bar for an **A** on the theory track.

A student who earned theory points for three of the four worksheets only, and whose exams earned 72% and 85% would have earned 3+3+4=10 theory points. This student would received a **B** on the theory track.

A third student who earned only two theory points from worksheets, and whose exams earned 55% and 70% would have earned 2+2+3=7 theory points. This would be enough for a **D** on the theory track.

7.4 Design Track

We will assess the *design* learning objective with self-evaluations and mutation testing connected to the homework assignments (see *implementation*) as well as with design documents connected to the final project.

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 as your **first** submission.

The purpose of these self-evaluations is to evaluate your submissions on *non-functional correctness*, i.e., desirable aspects of programs that go beyond strictly producing correct answers: e.g., efficiency, testing, robustness, or factoring. In software engineering practice, 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.

Each self-evaluation which earns credit for *at least half (50%)* of the questions earns one *design point*. This is a very modest bar, but be sure to learn from your mistakes to avoid repeating them in later assignments!

Mutation Testing

Starting with homework 2, we will assess the thoroughness of the test suites you write for your **first** homework submissions using a cutting edge software engineering technique called *mutation testing*.

Mutation testing consists of running a test suite on a series of intentionally-buggy implementations—referred to as *mutants*—and seeing if the tests can catch the bug in each one—referred to as *killing* a mutant. A good test suite kills all the mutants (i.e., catches all the bugs) whereas a poor one leaves some of them to survive (i.e., some of the bugs went undetected!)

We have created a number of intentionally-broken implementations for each homework, and will be running the test suites in your submissions on them. The more of these mutants your test suite kills, the more thorough it is! To help you refine your approach to testing (and not repeat the same blind spots in future assignments), you will receive a report with feedback from the mutation testing process.

Each homework where your tests kill *at least half (50%)* of our mutants earns one *design point*. Again, this is a very modest bar, but be sure to learn from your mistakes to avoid repeating them in later assignments!

Design Documents

The final project will be accompanied by three design documents whose goal will be to explain your design, explore alternative scenarios, and provide rationales for your design choices. Each design document which meets the criteria outlined in the project handout to a *satisfactory* level is worth one *design point*. More details will be included in the final project handout.

Learning from your Mistakes

While we expect you to follow good design practice as you work on assignments, we understand that slips and misconceptions happens. What matters, in these situations, is that you correct course and learn from the mistake.

Our design assignments feature opportunities for *redos*, which you can use to get credit for learning from your mistakes.[5] Redos are due the week after the original assignment. Like resubmissions, redos are optional. And like resubmissions, we will consider the higher score across both attempts when computing final grades.

Because design assignments require manual grading, redos involve additional labor for our course staff. To keep our staff's workload manageable, each student will be limited to three (3) redos over the course of the quarter, tracked using *redo tokens* on Canvas. Use those cautiously, and be careful to avoid silly mistakes in your initial submissions.

Details for redos for each kind of assignment are below.

Self-evaluations Self-evaluation redos must include (1) answers to all the questions of the relevant self-evaluation reflecting your *second* homework submission, and (2) a short (but complete) explanation of the changes you made to your initial submission following our feedback that led to these updated answers.

Mutation testing Mutation testing redos must include, for each surviving mutant, the line number of the specific test case[6] you added to your *second* homework submission which would kill that mutant. To get credit for killing a mutant in this second round of mutation testing, the specific test you pointed to must suffice to kill it; pointing to an unrelated test won't cut it.

Design document Design document redos must include both (1) the updated document itself, as well as (2) a short (but complete) explanation of the changes you made to your initial submission to address our feedback.

[5]With the exception of the third design document, which is due right at the end of the quarter.

[6]For large test blocks, the line number of the specific `assert` statement.

7.4.1 Design Grade

To determine grades for the design track, add up design points across self-evaluations (up to 5), mutation testing (up to 4), and design documents (up to 3, for a maximum total of 12) and consult the following table:

Points	11–12	9–10	8	7	0–6
Grade	**A**	**B**	**C**	**D**	**F**

For example, a student who missed one self-evaluation (4 design points out of 5), whose test suites killed at least 50% of mutants on all homeworks (4) and whose design documents were all satisfactory (3) would earn 11 design points total; enough for an **A** on the design track.

For another example, say a student wasn't careful when writing their assignments and earned design points for only two of their self-evaluations and two rounds of mutation testing, but did a better job on their three design documents which were all satisfactory. This adds up to 7 design points, which is enough for a **D** in design.

7.5 Final Grades

Final grades are computed from track grades as follows:

- If any of the track grades is **F**, the final letter grade is automatically **F**. All four learning objectives are important; students must be competent at *all* of them to pass the class and be ready to move on.
- Otherwise, take the *highest* grade which has been reached by **all four tracks**. This is the *base grade*, a student's baseline level of mastery of the material in this class.
- Then, count how many of the other tracks (0–3) are higher than the base grade. That's how many of them are *above expectations* relative to the base grade.
- For each track above expectations, increase the base grade by one step in the following sequence of possible grades: **D**, **D+**, **C-**, **C**, **C+**, **B-**, **B**, **B+**, **A-**, **A**. The sequence "saturates" at the top: there is no grade above **A**.
- Finally, if the resulting grade is **D+** (which is not a valid grade at Northwestern) round it to a **D**.

For example, suppose a student earns a **B** on the implementation track, a **B** on the integration track, a **C** on the theory track, and an **A** on the design track. This student's base grade would be **C**: the highest grade all four tracks have reached. All three other tracks are higher, which means their final grade will be three steps above **C**; i.e., they will finish the class with a **B**.

For a second example, say a student earns grades of **A**, **A**, **B**, and **B** on each of the four tracks. Their base grade would be **B**, and two other tracks would be above expectations, for a final grade of **A-**.

For a third example, another hypothetical student earns grades of **A**, **B**, **D**, and **A**. Their base grade would be **D**, and three tracks would be above expectations: they would end the class with a **C**.

For a final example, if a student has track grades of **B**, **B**, **F**, and **A**, their base grade would be **F**: any track grade of **F** results in a final grade of **F**.

Advice: because all four tracks are important, it's a good idea to focus on portions of the class where one is weaker, rather than doubling down on the portions where one is already strong. Which makes sense; this is how one learns the most! Please reach out if you want to discuss the most effective way to do that.

7.6 Late Policy

Unless otherwise indicated assignments are due by 11:59pm on their due date. Each Canvas assignment allows unlimited submission attempts, however, we will only consider and grade your **final** submission made to Canvas.

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. If more than one assignment (e.g., a self-evaluation, a resubmission, and a first submission) have the same deadline, one late token can be applied to all those assignments. Late tokens will be deducted automatically when you submit late; you don't need to reach out to us and ask for permission or even tell us about it. Note that an assignment that is received one minute later than the deadline will still count as using a late token.

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[7] 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.

7.7 Medical Accommodations

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

7.8 Religious Accommodations

Accommodation requests for religious reasons must be made by January 16th, 2026. You can find more details here: `https://www.northwestern.edu/accessibility/accommodations/religious.html`.

[7]Dean Joe Holtgreive (`jjh@northwestern.edu`) for McCormick students, Dean Rosemary Bush (`rosemary.bush@northwestern.edu`) for Weinberg students, and others for students in other schools.

7.9 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. If your learning is solid, grades will follow.
- **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 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 where you need to focus your efforts if you want to correct course.
- **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 challenge 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 and redos give you an opportunity to do so. And for places where those don't make sense, some slack is built-in to the grading standards to allow for small slips and mishaps.
- **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, redo tokens, late tokens, etc.—all of them with *no questions asked* and *no need to ask*. 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 assignments, be thorough in your testing and checking to avoid silly mistakes, make sure you read our instructions closely, and take 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 and allowed. If you answer the wrong question, we cannot tell if that came from a lack of understanding of the material or from momentary confusion; and we will have to assume the former.

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 9.0 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 use DSSL2, make sure DrRacket is set to "Determine language from source" in the bottom left corner[8] and start your program with the line: `#lang dssl2`

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 algorithms 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.[9] I have not tried it, though, so *caveat emptor*.
- **Vim**: Vim users can try Marko Vejnovich's DSSL2 plugin.[10] I have not used it myself either.

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.

You can also compile programs with DSSL2 using VSCode without a plugin using the following approach: Open your file in VSCode, then open up the "Terminal" within VSCode. You can find the terminal at the bottom of the screen (the icon with the "X" and the triange with an exclamation mark) or look for "Terminal" in Help. As long as you opened the file directly in VSCode, the terminal should be operating within the directory of your file (change the directory if not). In that terminal, run your program with the command "racket file.rkt". The output of this command will be the same as what you would see in DrRacket.

[8]If you see something else, like "Advanced Student Language" click the bottom left and choose "The Racket Language".

[9]`https://github.com/Gamefreak130/dssl2_vscode_extension`

[10]`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 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)

10 Northwestern University Syllabus Standards

This course follows the Northwestern University Syllabus Standards.[11] Students are responsible for familiarizing themselves with this information.

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

11 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.

11.1 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.

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 regardless of instructor: resubmitting or otherwise reusing your work from earlier attempts at the class is **also** considered plagiarism. You must 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.

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

11.2 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.

11.3 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.

11.4 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 will make a bad situation much worse.

11.5 Agreement

Please confirm that you have read and understood the class's academic integrity policy, and that you will abide by it this quarter. Print the pages from the syllabus relevant to the academic integrity policy and bring a signed paper copy to class **by Class #2**. We will not accept work from you until you do so.

Printed name: ______________________________ **Net ID:** __________

Signature: ______________________________ **Date:** __________