

# CSE 332: Data Structures and Parallelism

---

## Course Manifesto, Autumn 2018

### What Is This Document?

This document is a supplement to the course syllabus which explains the distinctive features of this course and how they work.

### Course Pre-Requisites

There are two pre-requisites for this course: a programming course (CSE 143) and a discrete math and structures course (CSE 311). These two courses are *equally important*. In CSE 332, you will be writing code and proofs. You will be solving discrete math problems and algorithmic problems. You will be implementing data structures and analyzing them. Characterizing this course in your mind as a “programming” course wouldn’t be quite right—but characterizing it as a “discrete math” course would be equally not quite right.

### Course Goals

CSE 332 is a course about *data structures, abstraction, analysis, algorithms, and parallelism*. We take these course goals very seriously and every exercise, lecture, section, and project will hit on at least one of them. Here’s a quick run-down:

#### Data Structures

You’ve seen some data structures in previous courses, but we will focus on understanding the *ADTs* and the *implementation* of them now. During this course, you will implement a Stack, a Queue, a Heap, a trie, a balanced binary search tree, and a hash table. We will also study graphs which can be seen as a generalization of many of these.

#### Abstraction

Every time you write a piece of code, you are making an abstraction. CSE is based around the various abstractions that we make (when we use Java, we usually don’t have to worry about where in memory the Strings are stored or how the operating system decides when your program runs).

As we discuss CSE 332 topics, we will explore various abstractions that make our lives easier as we design various data structures and algorithms. Because of this, the abstractions that you use when designing the project will factor significantly into your grades on them. Additionally, this means that the projects will *not always be explicit about what you should do*. Design decisions and implementation details will be left to you.

#### Analysis

We will use the mathematical background you gained in CSE 311 to analyze algorithms and data structures throughout the quarter.

#### Algorithms and Parallelism

During the second half of the quarter, we will focus on parallel algorithms and graph algorithms. This (especially the parallelism part) will diverge significantly from the mental model you’ve used to program thus far.

## Course Meta-Goals

This course has several “hidden” meta-goals that you should be on the lookout for.

### Real-World Applications

All of the projects that you complete in this course will have a “real-world” deliverable. The goal is for you to be proud of what you’ve built. Your responsibility for the projects will mostly be in the back-end (the data structures and algorithms that make these applications work).

### Larger Code Bases

All of the projects come with code bases that are significantly larger than what you likely saw in previous courses. You can ignore much of the supporting code, but you *must* read and understand the interfaces you’re given. Each project will have a package called `cse332.interfaces.*`, and it is to your advantage to at least read the comments in the classes in this package.

### Group Work

See the handout on choosing a partner for more detail on this.

## Course NON-Goals: Testing and Debugging

This is **not** a course about *testing code*, and this is **NOT** a course about *debugging code*. You will, however, likely spend a significant amount of time on the projects doing debugging. There will be two sets of tests for every project: `gitlab-ci` tests and *private tests*. To avoid wasting your time, we will release the `gitlab-ci` tests as early as possible (as well as their source code). You will not be able to see the results of the private tests until after your project has been graded. The `gitlab-ci` tests will run every time you push your code to gitlab; so, we recommend that you push early and often.

## Projects and Exercises

In this course we will have two streams of work that will be going on in parallel (Parallelism is in the title of the course after all :-). We will try not to make things from the project stream and the exercise stream due on the same day, but ultimately you are responsible for managing your time. In general, once a project has been submitted, the next one will be released (if not before). Similarly with weekly exercises. It is not intended that you wait to start on exercises until you have finished the most recent deadline for a project. The idea of the parallel streams is this allows us to give you the largest amount of time for both the projects and the exercises. It leaves time management up to you.

- We will have three projects, each weighted approximately equally. Due dates for these will be listed on each project specification. Each project will have between one and two checkpoints (described below). Checkpoints will be held on Tuesdays. Projects will be done with partners and may be submitted at most 2 days late (see policy below).
- Most weeks, we will have (2-4) exercises due, for a total of 20 or so exercises. These will typically be due on Friday. Each exercise will have its point total listed, but the goal is for exercises to be of approximately equal weight. Exercises will be completed individually and will NOT be accepted late.

## Tokens (“late days”)

You begin the quarter with *four* “token”s. The CSE 332 “late days policy” works as follows:

- *Projects and Tokens for late days:* During the quarter, each token may be used to gain 24 extra hours for a *programming project* (this is a standard late day). You may only use a token on a programming project if both partners have a remaining token (both must use a token to get the late day), and you may not use more than two on any individual project. That is, projects cannot be turned in any later than two days past the original deadline. A form will be posted with each project where you will need to request to use a token on that project (This form will only become available after the deadline has passed).
- *Exercises:* Exercises will NOT be accepted late. If you have a serious illness or have some other legitimate reason to turn in an exercise late, contact the instructor via e-mail. We make no promises, but we will be as lenient as we can within reason. In all other circumstances (e.g., taking a trip, missing the deadline, oversleeping, etc.), exercises will not be accepted late.
- *Re-doing Exercises with Tokens:* In the last week of classes, you may use any remaining tokens you have left to *redo* that number of exercises. That is, you can exchange each token for a chance to re-submit an exercise and incorporate the feedback you were given. After using a token in this way, your new grade on the exercise will be the grade your re-submission gets.

## Checkpoints

Every project will have between one and two “checkpoint”s. All checkpoints will be held on Tuesdays.

### What Is A Checkpoint?

CSE 332 is likely one of the first courses that you are taking where the project durations are closer to a month than a week. A “checkpoint” is a short meeting between your project group and a member of the CSE 332 course staff to ensure that you are making progress and are on track to finish the project.

Each project indicates which pieces are “due” at the checkpoint. To “pass” a checkpoint, you will need to demonstrate that your code is passing the tests for these pieces. You will need to bring a phone or laptop to each checkpoint meeting.

### Failing A Checkpoint?

If you do not put forth a “good-faith-effort” to meeting the checkpoint, we will note this down. If you consistently get noted as failing a checkpoint, it will affect your grade. (In other words, an isolated incident is okay, but more than once and we will have serious concerns.)

It is important to note that not passing the tests does *NOT* mean that you fail the checkpoint. If you have been working and haven’t quite been able to debug all of your code, that is okay. That said, we recommend you do your best to actually pass all the tests, because getting behind early is an easy way to never catch up.

*written* feedback on projects tends to be minimal—the real feedback is verbal, and we *expect you to come and talk to us about it*.

### But I Don’t Want To Show Up!

Unless you have substantial questions, checkpoints really should only take a few minutes, but in the case where your group is passing the relevant `gitlab-ci` tests, we offer the following alternative:

You and your partner may both fill out an “opt-out” form indicating that you have passed the `gitlab-ci` tests and do not wish to meet with a member of course staff.