CSE 332: Data Abstractions

Course Manifesto, Winter 2016

What Is This Document?

Very few people read course syllabi, and we understand this. This document is a supplement to the course syllabus which explains the distinctive features of this course and how they work. In particular, many of the policies and decisions made around this course are based on pedagogical literature. If a policy doesn't make sense, or you have a good reason for an exception to be made, don't hesitate to contact the instructor. We make no promises that we'll be able to accommodate all requests, but we will make an effort to do so.

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.

This course will involve programming and theory. It's a feature, not a bug!

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 *ADT*s and *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 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.

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.

Tokens ("late days")

Many CSE courses have a quota of "late days" that you are allowed to use over the course of the quarter. Your instructor likes the idea of leniency but hates how these late days end up working out. Some students try to save their late days "in case they need them" and end up not using late days when using one earlier on would have helped significantly. Other students use too many late days early on and then never get back on track.

In CSE 332, there are two distinct streams of assignments: "exercises" and "projects". This introduces an additional wrinkle: using a late day on an exercise seems wasteful as compared to using one on a project.

The CSE 332 "late days policy" works as follows:

- If you are sick 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. You may or may not lose a token depending on the situation.
- You begin the quarter with *four* "token"s. During the quarter, each token may be used to gain 24 extra hours for a project (this is a standard late day). You may only use a token on a partner's 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.
- 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.

Ulterior Motive?

Nearly every checkpoint will have an "ulterior motive". We will use these opportunities to discuss project feedback, exam feedback, etc. with you. As a result, the *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*. The checkpoints are also the *only* opportunity to initiate a re-grade on a project or exam.

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.

Please note that we will often point out concerns about your code at checkpoints, which, left unfixed will turn into points lost, and that if you take the opt-out option, you are forfeiting this free advice. If you take the opt-out option, you are also **forfeiting the right to a regrade reqest** on any project/exam returned and discussed at that checkpoint. You may not opt-out and then e-mail a regrade request. We find that discussing regrades in person leads to better resolutions in almost every situation.