Midterm Prep Guide (Fall 2024)

Here is some information on what to expect from the midterm. See the table of contents for high-level sections.

We will hold at least one review session in the week before the midterm–details TBD after we finalize room bookings.

Logistics

When: Thursday, October 17 from 7-9pm EDT
Where: **Barus & Holley 166**. This is an **in-person**, paper exam.
Instructions: You can view the exam instructions page here.

You must bring your ID – proctors will check it as you turn in your exam

Special notes

- *If you have a letter through SAS/SEAS for time-accommodations, you will receive an email over the weekend regarding logistics. You will get whatever extra time is due to you according to your letter. If you have SAS/SEAS accommodations and have not yet sent your letter to Nick, please do so ASAP.*
- *If you have a conflict with the exam due to another course, please email cs200-profs@brown.edu ASAP if you have not done so already. If you have already emailed Nick, thanks! You'll get a response over the weekend.*
- *Otherwise, you must take the exam in-person on the scheduled day and time.* ***If there are extenuating circumstances that would prevent you from taking the exam, contact Nick ASAP at cs200-profs@brown.edu***

Overview

The exam will focus on conceptual questions, design choices, analysis, and other questions that are better answered on paper rather than in code. While you might be asked to read some code or identify problematic parts of code that has been given to you, you will not be asked to write more than a line or two of code on the written exam. The questions will be a mix of multiple choice, fill in the blank, draw a diagram of something, and free response.

The questions are not designed to trick you, nor will they introduce new content.

Notes and references: You may bring one 8.5x11in sheet of paper with notes (written or typed). You may write on both sides.

The exam will be self-contained, in that it won't expect that you remember any details of specific examples from class or lab. In other words, a notes page will not be necessary.

What if I need a clarification during the exam? Raise your hand and a proctor will come over to you.

Practice Exams

We have several old exams and solutions available that you can use to practice. Some notes on these:

- Some of these exams were given online (some via Gradescope, some not)--**your exam will be in-person and on paper**.
- **Some of these exams reference "Project 1", which we haven't encountered yet.** We will not ask you about Project 1, but you may see questions in a similar format to some of these (ie, determining runtime from a piece of code). The last column of the table notes which exams have Project 1-specific problems, and how you should interpret these for practicing: Nick recommends skipping some problems (because they rely on too much context we haven't seen), and others are okay (because they're sufficiently generic). You will *not* be asked about decision trees (which is the topic of Project 1).

Here's a guide to what exams are available and where you can find them:

If the Canvas link doesn't work: just log into Canvas and go to the **Files** section for our course–you need to be logged in to Canvas with your Brown account to view the files.

Exam	Format	Where to find	Solutions	Comments
CS200 Spring 2024 #1	Paper	Canvas	Canvas	• This midterm had two versions, this is version 1. • **Nick is looking into Question 2 (details here).** Don't worry too much about Q2 right now. • **Skip questions 8-10.** These rely on Project 1, so they are out of scope.
CS200 Spring 2024 #2	Paper	Canvas	Canvas	• This midterm had two versions, this is version 2. • **Skip question 11.** This question relies on Project 1, so it's out of scope.
CS200 Fall 2023	Paper	Canvas	Canvas	• This midterm was a bit too short. Yours will probably have one more problem than this.
CS200 Spring 2023 #1	Paper	Canvas	Canvas	• This midterm had two versions, this is version 1. • **Questions 9-12 reference project 1, but don't require project-specific knowledge**. Nick thinks these are okay as practice problems.

CS200 Spring 2023 #2	Paper	Canvas	Canvas	• This midterm had two versions, this is version 2. • **You should skip questions 7-9.** Nick thinks these rely on a bit too much context from Project 1 to be fair as practice problems. (You may find it useful to look at the solutions, though.)
CS200 Spring 2022	Online	Gradescope	Canvas	• This was a 90 minute exam. You will have two hours, so your exam could be longer. • Gradescope is really picky about answer formats–see this note for details, but don't worry too much since your exam will be on paper

There are also some study questions in this Recap on Lists.

Note about the practice exams on Gradescope: If you try the practice exams in Gradescope, be aware that Gradescope is very picky when auto-grading fill-in-the-blank and multiple-choice answers–**we will not be picky on your exam, so long as we can understand what you've written**. For example, if we asked you which kind of list to use and you typed "array list" whereas our default answer for Gradescope to check was "arraylist", Gradescope would mark it wrong even though it is fine. When these exams were given, we audited all the grades and manually gave full credit for such situations–so pay attention to the *substance* of the answers in Gradescope, not the *points* that Gradescope's limited power suggests on the practice exams.

What do you need to know?

The exam will cover material up through the Friday, October 11th lecture material on Exceptions, and any clarifications posted after the lecture.

Data Structures: We expect that you know the data structures we have studied so far (Linked Lists, Arrays, Dynamic Arrays/ArrayLists, Doubly-Linked Lists, Trees, and essentials of HashMaps). You should be able to choose from or argue for or against these data structures for a given problem. You should be able to talk about the running times for operations that we have discussed on these data structures (at the level of big-O, not line-by-line time annotations).

For those data structures that you implemented on homework or projects, we expect you to be able to discuss roughly how those implementations work (but you won't be asked to reproduce or remember the corresponding code in detail).

The exam will **not** include details of how the runtime on Hashmaps works out to be constant, or why the runtime to add to an ArrayList works out to be "amortized" constant.

Object-Oriented Design and General Programming: You should understand

- what classes and interfaces are and when they get used.
- what it means to extend a class or implement an interface
- which class a computation should be in
- how constructs like loops, assignment operations, and exceptions work
- how recursion works across related classes
- how and when to throw (or catch) an exception

How Memory (Heap) and the Environment Work: You should understand how different constructs impact each of the environment and the heap. You should be able to draw the heap contents that result after a sequence of lines of code and talk about how objects in the heap are accessed through different names and expressions via the environment. You may be asked to draw a memory diagram, similar to those that you see in the practice exams.

Programming with Mutable vs Immutable Lists: You should understand the difference between using an immutable (functional/our LinkList) list and a mutable (Java built-in lists) from a programmer's perspective. For example, you should be able to discuss the difference in how a piece of code would run depending on which kind of list was being used.

The key ideas from assignments: We expect that you understand how your solutions to hwk 2 and hwk 3 work. You won't be expected to reproduce code, but we could show you part of a homework/project solution and ask what that part does or why it was important to solving the problem.

What do you NOT need to know?

- Syntax details -- you won't be asked to write more than a line or two of code, and even then, missing or incorrect bits of syntax won't matter
- Specific examples or code from class, homeworks, or lab -- the exam will be self-contained, rather than say things like “remember the Dillos? ..."
- The exact names of built-in methods on Java data structures -- as long as we understand what operation you are trying to do, you'll get credit for your answer.
- How to define exception classes
- How to annotate code line by line for runtime
- Specifics from the separate tracks the first two weeks.
- **How** the runtime of accessing a hashmap works out to be constant (but you should remember that it *is* constant)

Basically, this isn't a memorization exercise, or paper-based versions of questions that would be more easily answered in a programming environment. This is about the “bigger picture” rather than details of code.

What sort of questions might be asked?

Here are examples of what you might be asked to do (this list is not exhaustive):

- Given a problem scenario, describe the tradeoffs among various data structures (that we've covered) for use within the problem. For this, you would want to know the running time of standard operations of various data structures.
- Given a collection of classes and interfaces, discuss whether the various methods and variables are in the right places, or whether they should be organized differently.
- Given a partially-filled-in heap, show how it would change after running a given couple of lines of code.
- Given a piece of code, answer questions about what it would do.
- Explain what aspects of a data structure implementation or problem statement achieve certain goals (e.g., what part of an immutable list implementation allows adding an element to be constant time).

As you can see from these examples, the focus here is on concepts -- do you understand the material we covered so far this semester in a way that lets you make good design decisions?

How will this be graded?

There will be partial credit on all questions.

As a general rule, we are looking for you to demonstrate what you understand about the material so far. If you are asked to "justify your answer", say on a question about which data structure you would choose, don't just say "it's faster" – give a specific runtime instead. Don't just say "it's easier" – tell us what operation is easier and for who (the programming who is using the data structure, the person building the data structure, etc). Details are where you earn your points.