Lecture 38: Information on the Final

Contents

1 Exam Details 1

2 FAQ 3

The Point of This Document

Here is some information on what to expect from the final. As a general rule, the exam will not be designed to trick you. It will simply check how well you understand core concepts from the course.

For sample questions, see the exam from 2019, which is available in Canvas, under the Files section. Keep in mind that last year’s exam was designed to be given on paper, while we have to make adjustments to question formats this year to accommodate online. Also, last year’s exam was designed to take roughly 90 minutes within a 3-hour slot.

1 Exam Details

Overview: The exam will focus on conceptual questions, design choices, analysis, and other questions that are better answered open-response 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.

Time expectations: The exam will be designed to take 60 minutes (you’ll have at least 90 minutes to work on it).

Notes and references: The exam is open notes. You may use your own notes, and any notes that we have posted. You may not share notes with other students (you are welcome to prepare notes together, but no using google docs, etc, to communicate info once someone starts taking the exam).

The exam will be self-contained, in that you could have no notes and still answer all the questions if you knew the properties of the various data structures and algorithms we covered, as well as the general mechanisms for structuring programs (traits, classes, etc). The exam will not expect you to recall details of specific lab or homework questions.

What do we need to know?

Data Structures We expect that you know the various data structures we have studied this semester (Linked Lists, Arrays, Dynamic Arrays, Binary Search Trees, Balanced Binary Search Trees, Heaps, Hashmaps, Priority Queues, Graphs, and Disjoint Sets). 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 operations.
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).

**Algorithms** We’ve looked at searching and sorting algorithms, depth-first and breadth-first search, shortest-path algorithms, minimum spanning tree algorithms, as well as dynamic programming. You should understand what problems each of these algorithms help solve, and roughly how the algorithms work. You should be able to design a recurrence for a dynamic programming problem.

**Object-Oriented Design and General Programming** You should understand what classes and interfaces/traits are and when they get used. You should understand programming concepts like public and private vars, val and var modifiers, mutable and immutable data structures, and how constructs like loops, exceptions, and assignment operations work.

The exam will be designed to allow answers in either Java or Scala, as you prefer.

**What do we 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 bits of syntax won’t matter

- Specific examples from class, homeworks, or lab— the exam will be self-contained, rather than say things like “remember the armadillos? ...”

- Details of how page rank or decision trees work.

**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 or algorithms (that we’ve covered) for use within the problem. For this, you would want to know the running time (and perhaps space performance) of standard operations of various data structures, as well as the purpose of various algorithms we’ve covered (e.g., depth-first search vs Dijkstra in graphs, etc).

- Given code that implements some (new) algorithm, explain the time- or space-performance of the algorithm in terms of big-O analysis. You will not be asked to formally prove big-O, but you would be expected to justify your answer (with statements like “we loop over each edge and perform operation X which has worst-case time O(log e), where e is the number of edges”)

- Given a hierarchy of classes, traits, and/or interfaces, discuss whether the various methods and variables are in the right places, or whether they should be organized differently.

- Given code that claims to solve a particular problem, discuss whether proposed changes to the code could lead to a more efficient solution (such as when we changed the strategy for updating parents in the disjoint-set data structure).

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

Do I need to pass the final to pass the course? You don’t need to pass the final on its own, but you need a passing average on the solo assignments of the course (graded quizzes, programming exam, and final exam), weighted according to the weights in the course missive. That said, failing to pass the final will invite greater scrutiny of the rest of your work as we decide final grades. I can’t state a numeric passing bar at this time because I haven’t yet written the exam, but it should come in around 60%.

How much will the final exam be worth in determining final grades? The course missive lists the quizzes+final as being worth 15%, which is the default weight that we’ll use with everyone. That said, I do compute course grades manually, and I will look at the broad picture of your work when deciding course grades for those of you who end up on the border between two grades.

Please let us know if you find any mistakes, inconsistencies, or confusing language in this or any other CS18 document by filling out the anonymous feedback form: [https://cs.brown.edu/courses/cs018/feedback](https://cs.brown.edu/courses/cs018/feedback)