Lecture 38: Information on the Final

Contents

1 The Take-Home/Programming Portion 1
2 The Written Portion 1
3 FAQ 2

The Point of This Document

Here is some information on what to expect from each part of the final. This document does NOT contain the exam questions. Those will be posted separately.

1 The Take-Home/Programming Portion

Overview: For this portion, you will be asked to write code for a couple of problems, perhaps with constraints on which language features or data structures you can use (in particular, each problem will specify whether to use Java or Scala). This part of the exam checks whether you are able to make and execute decisions about implementing solutions to problems.

Time expectations: The exam will be designed to need roughly 4 hours (on average across students). You may work on it throughout the period when it is out (there are no restrictions on the amount of time you spend on this portion, you can come back to it multiple times, etc).

Grading: In general, we will be looking at whether you can produce working implementations (as graded through test suites), whether you make good decisions in structuring code (what you put in interfaces, traits, abstract classes, etc). You may be asked to write some tests, but thorough testing will not be a requirement for this part of this exam. More details will be on the exam handout, alongside the actual questions.

Notes and references: You are welcome to consult class materials, language documentation, and your own notes while working on this part. General web searches, Stack Overflow, etc. are not permitted. Collaboration with others is not permitted.

2 The Written Portion

Overview: This portion 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.

Time expectations: The exam is scheduled in a 3-hour slot. The exam will be designed to take roughly 1.5 hours (on average across students), so everyone should have plenty of time.
Notes and references: You may reference notes on paper, but you may not use any electronic resources (laptop, phone, etc) during the exam. You are welcome to write up separate notes to bring to the exam (i.e., you don’t need to use only notes that you took during the semester). You may not share notes with others during the exam.

For a sense of what notes might be most useful, see the next section on possible types of questions. The exam will be self-contained, in that you could come with 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 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?

3 FAQ

Do I need to pass the final to pass the course? We are giving a final exam to check whether you are able to do work related to the course entirely on your own, without assistance from TAs or others. Failing to pass the final is thus a serious warning sign that we will look at closely when determining final grades. So, there is no hard and fast rule, but failure to pass the final will invite greater scrutiny of the rest of your work as we decide final grades, and may lead to you NCing the course. We can’t state a numeric passing cutoff at this time because we haven’t yet written the exam, but the cutoff will be fair (rather than harsh).

How much will the final exam be worth in determining final grades? The course missive lists the final as being worth 20%, which is the default weight that we use with everyone. That
said, we do compute course grades manually, and we 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: [http://cs.brown.edu/courses/cs018/feedback](http://cs.brown.edu/courses/cs018/feedback)