Second Midterm Exam Logistics

Midterm II: Quick Info	
Exam Goes Out	March 12th, 12:30PM Pacific.
Exam Comes Due	March 14th, 12:30PM Pacific.
Topic Coverage	Cumulative; mostly Lectures 10 – 18 and Assignments 4 – 7. Topics from Assignment 8 and Lectures 19 – 25 are fair game but deemphasized. Topics purely in section or the textbook aren't tested.
Exam Format	47-hour take-home exam, written to take three hours, structured similarly to a standard programming assignment.
How It's Graded	Similar to programming assignments: functionality score is based on correctness; style score is based on coding style. Functionality is weighted at 90%; style at 10%. Style is graded as in the assignments.
Honor Code Policies	See handout for full details. Open-book, open-note, open-Internet. You are not allowed to communicate with other humans about the exam.
How to Prepare	Do lots of practice problems, and keep the course staff in the loop.

Exam Timing and Format

The second CS106B midterm exam is coming up soon. It's a take-home coding exam. The exam will go out on Friday, March 12th at 12:30PM, and it will come due on Sunday, March 14th at 12:30PM (Pacific time). We've designed it to take three hours, though you'll have 47 hours to complete it. The exam focus is

- Lecture 10 – Lecture 18 (recursive backtracking up through and including hashing)
- Assignments 4 – 7.

The topics from after this point (Lectures 19 – 25 and Assignment 8) are also fair game for the exam, though they will not be as emphasized. In particular, we'll assume that you're comfortable with linked lists, binary search trees, Huffman coding, and basic graphs, and that you've completed Assignment 8. Topics from earlier in the quarter may also be tested directly or indirectly.

You are responsible for the material from lectures and from the assignments, so you should be prepared to answer questions about topics that came up purely on the coding assignments (binary heaps, enumerated types, etc.). We will not test you on anything that appears purely in the section handouts or purely in the textbook, though those are excellent resources when studying.

The exam format is very similar to that of the assignments. We'll release a set of starter files for Qt Creator and expect you'll be doing your coding there. We'll provide the problem descriptions themselves through the CS106B website, the same way we've handled normal assignment handouts. And you'll submit everything through Paperless, just as usual. (We do not have the same 48-hour grace period for late submissions; ***please leave adequate buffer time to submit!***)

Our Exam Philosophy

We want you to do well on this exam. To be very explicit about our exam philosophy:

- We're ***not*** trying to weed out weak students.
- We're ***not*** trying to enforce a curve where there isn't one.
- We're ***not*** trying to measure "coding potential" or "innate programming ability."

Rather, we want you to show what you've learned up to this point so that you get a sense for where you stand and where you can improve. The exam format is basically the same as the coding assignments, with the exception that you'll be doing everything individually and without help from the course staff or other students.

Honor Code Policies

You are required to abide by the Honor Code policies outlined in the Honor Code Policies handout available on the course website. We'd like to call particular attention to the following rules.

This midterm exam must be completed individually. It is a violation of the Stanford Honor Code to communicate with any other humans about the exam, to solicit solutions to this exam, or to share your solutions with others.

This exam is open-book, so you are free to make use of all course materials on the course website and in the textbook. You are also permitted to search online for conceptual information (for example, by visiting Wikipedia). However, ***you are not permitted to communicate with other humans about the exam or to solicit help from others***. For example, you ***must not*** communicate with other students in the course about the exam, and you ***must not*** ask questions on sites like Chegg or Stack Overflow. (You may ask questions to the course staff on Ed; if you do, you must post your questions privately. See below for details.)

All work done with the assistance of any material in any way (other than provided CS106B course materials) must include a detailed citation (e.g., "I visited the Wikipedia page for *X* on Problem 1 and made use of insights *A*, *B*, and *C*"). ***Copying solutions is never acceptable***, even with a citation, and is always a violation of the Honor Code. If by chance you encounter solutions to a problem, navigate away from that page before you feel tempted to copy.

If you become aware of any Honor Code violations by any student in the class, your commitments under the Stanford Honor Code obligate you to inform course staff.

Getting Help

Because this is an exam, we will not be able to help out with coding or debugging questions the way that we normally do on the assignments. For example, we will not be able to help interpret compiler error messages, assist with C++ syntax, investigate crashes in your code, assist with debugging, etc. After all, part of the purpose of the exam is for you to demonstrate your ability to work through coding problems independently.

We can, however, answer a small set of questions about the exam problems, mostly clarifying questions about what the instructions mean or help getting the project files set up on your machine. To ask these sorts of questions, make a private post on EdStem. ***All questions about the exam must be posted privately***, and ***you must not post any code on EdStem***.

Preparing for the Exam

There are a number of ways that you can prepare for this upcoming exam. Here is our recommendation of what you should do to get into the best shape that you can.

1. ***Redo the assignments.*** When you're first working on the assignments, you're simultaneously trying to figure out what the assignment is asking you to do, solidifying your understanding of the content from lecture, tinkering around to see what happens, and figuring out the necessary problem-solving techniques. That's fine, and that's normal. What matters, though, is that, in that process, you internalized the appropriate techniques and developed your coding and problem-solving skills.

   If you have the time to do so, pick one or two of the most challenging programming questions we asked you to solve, download a fresh copy of the starter files, and *solve the same problem again without referencing your initial solution*. If you're able to do so with a little trial and error, great! It means that you've gotten out of that assignment what we expected you to get out of that assignment. On the other hand, if you're struggling on the assignment a second time, there's a good chance that some key skill or technique hasn't yet clicked for you, and it's worth talking to your SL or stopping by the LaIR to talk through the ideas.

   If you worked with a partner, it's doubly valuable to attempt the problems a second time on your own, just to make sure that you personally are comfortable taking on the questions and that you weren't leaning too much on your partner for insights.

2. ***Work through practice problems***. We've posted a collection of practice problems on the course website. It's a compilation of old exam problems, which might give you a better sense of what sorts of questions we might ask.

   When you're working through those problems, *don't just hand-write things and call it a day*. Make sure that you actually got things working. So download a blank set of starter files from the course website, type up your solution, and try running it on some sample test cases. If things work, great! If not, try debugging your code and see if you can fix it. If you're still stuck, no worries! That's a great indicator that you should ping your SL or drop by the LaIR to get some help.

3. ***Work through the section problems.*** The problems we give out in the section handouts each week are a great way to practice specific skills.

4. ***Review IG feedback and make sure you understand it completely and unambiguously.*** We hold interactive grading sessions on the assignments for a good reason – it's a chance for someone with more coding experience than you (your section leader) to offer their advice and insights about how you can do a better job in the future. If you haven't yet done so, take some time to review the feedback you received. If there are suggestions of the form "try doing it this way next time," take a few minutes and actually go do it the other way. If there are stylistic points that they've pointed out to you, great! Go patch up your code to meet those style guidelines.

   And hey, what should you do if you find something that you don't understand? Ping your SL and ask for clarification!

5. ***Keep the SLs in the loop.*** As you're studying, please take the initiative to ask us questions when you have them. If you're not sure about how or why a certain piece of code works, or why a certain piece of code *doesn't* work, or why a certain concept works a certain way, etc., go on EdStem or stop by the LaIR and ask us a question. If you worked through any practice problems (section handouts or practice exams), ask your section leader to review your answers and offer polite but honest feedback on how you did and what you need to work on. You can also visit office hours if you'd like!

How We Grade

We'll be grading the exam similarly to how we grade programming assignments. We'll run your code on a variety of test cases (many of which we'll have shared with you in the starter files) to determine what aspects of the code are working properly and which aspects need work, and we'll review how you've structured your code to assign a style grade.

We do not assign points by counting how many tests you pass. Instead, we review your code, identify what works and what doesn't, and assign a score based on your overall correctness and strategy. This means that if your code is mostly on the wrong track but coincidentally happens to pass most of the tests, we may assign a lower score than if your code is basically correct but has a tiny error that causes it to fail most of the tests. In other words, it's always best to try to get the code working according to the best of your ability, rather than submitting something that you know is mostly incorrect but happens to pass a lot of the tests.

As with the previous midterm, the point balance will be ***90% functionality***, ***10% style***. Unlike the first midterm, ***our style expectations for this midterm are the same as in the assignments***. Please refer to our style guide for more details.

We want to be transparent about our grading philosophy for exams and our exam policies. Here's a quick rundown of some of the frequently asked questions about CS106B exams.

- ***Do you give partial credit?*** Yes, absolutely! Because the exam is open-resource, we generally do *not* award partial credit for answers that just consist of a lot of code copied from lecture, section handouts, etc. The best way to earn partial credit on the problem is to make a good effort to solve it, and to do so in a way that shows you understand the underlying methodology.
- ***Can I write more than one answer to a problem?*** No, please do not do this. If you put down multiple answers to a question, we will grade whichever answer gives you the *fewest* number of points. This policy is in place to prevent "shotgunning" down multiple answers with the hope that one of them will work.
- ***Is pseudocode okay?*** We generally discourage people from writing pseudocode on an exam. It is better to just write real C++ code, or to at least outline in C++ what you would be doing. You should not expect to receive much, if any, partial credit for writing pseudocode.

You can do this. Best of luck on the exam!