CSE 154 Final Exam “Tips and Tricks” \(^.^)/

This guide includes some carefully-curated tips from your TAs on how to study and take a CSE 154 final exam. We’ve include common “exam problem categories” to guide your studying, but keep in mind that anything we’ve emphasized in the course (lecture, section, lab, homework) may appear in some way or another on the exam (e.g. for Short Answer questions, similar to what you saw on the Midterm).

You may find certain tips more useful than others depending on your study styles, but we recommend you read through each category as you study each type of major topic (the exam is cumulative). We hope this helps, and good luck!

General Studying Tips

- Breathe. You're here to learn. You've successfully completed the homework assignments, section problems, and labs, and it's just a matter of applying what you've learned to different problems!
- Practice early so you don't cram the day before! A clear head is essential to doing well on a CSE exam, as there may be different types of problem-solving involved. With that, it's perhaps most important to get at least 7 hours of sleep the night before to be able to have a sharp mind when the time comes to solve new problems with familiar topics.
- Prioritize reviewing sections and labs - these are designed to supplement weekly material through practice problems (application).
- Then, when you feel more confident about the material, work through provided practice exams on paper, not your computer. This is the first time you will have an exam for web programming, and it's really easy to forget common syntax when you're used to typing it and using autocomplete features. You definitely want to get used to solving these without a screen.
- When working on practice exam problems, read the description at least once before attempting. Underline any small details (case-insensitivity, error handling, query parameters, etc.).
- Then read the problem once more carefully after you've finished your solution. It's easy to forget the small details, and it's worth the 1-2 minutes to double-check. Seriously, this is strongly recommended.
- Make sure to give yourself space between statements to account for any last-minute fixes/special cases in your problems.

- When studying, study what you don't want to study. It's a common habit of students to avoid what they're dreading, but this is unfortunately a recipe for not being prepared when the time comes to solve that type of problem on the exam. For example, if you still don't understand how to build a JSON response in PHP, watch the lecture video on those again and review the associated section/lab.
- Make sure to review your midterm exam - make sure you understand why any answers were incorrect and identify common mistakes (e.g. referencing undefined variables, not using setInterval appropriately, etc.)
- Practice using the cheat sheet and become familiar with it - make sure you understand all of the material covered. During an exam, students can easily spend a non-trivial amount of time looking things up.
- Look over your homework assignments. You've came long and far. However, the final exam will be cumulative and many parts may be answered by what you've done on your homework. You know your homework the best… use it!

HTML

For problems involving HTML, you should be familiar will working with HTML (including reading and finding validation errors), but **you will need to write HTML from scratch** in this exam.

HTML: Tips for Studying

- Review the different types of HTML tags and common validation errors. You may find the exam cheat sheet and Problem 1 from the Midterm helpful for this.
- Make sure you're familiar with the different type attributes of <input> tags (e.g., text, radio, checkbox, etc.) and their different attributes (e.g., checked, value, and name). Remember that <button> and <textarea> are tags on their own so they do not need to be set as the type attribute for an <input> tag.

HTML: Common Mistakes

- Mixing up attributes for different tags (e.g. src on the <a> tag instead of <href>).
- Be careful with tags that are self closing (e.g. img should not have a closing tag)

CSS

These types of problems usually ask you to answer questions about CSS selectors or styles, and/or write CSS to meet given criteria (sometimes given a screenshot) with provided HTML.

CSS: Tips for Studying

- Take out a piece of paper. Take 5 minutes and
    1. write out all of the *CSS properties* (e.g., "display", "border", "font-family", etc.) you can recall
    2. the different *values* you could give each (e.g., "flexbox" for the "display" property)
    3. the *different HTML tags* you might need to set the property on.

  At the end of the 5 minutes, review any properties you may have felt less confident about and make sure you know how to use them for different elements. You may find the exam cheat sheet helpful for this, as well as the related lecture/section/lab slides.
- Use codepen.io to play with different CSS properties.
- Review the differences between block (e.g., <p>) and inline (e.g.,) elements.
- Review the differences between ID's and classes.
- Review the box model and how to use it (e.g., flexbox display, padding vs. margin).
- Visit your favorite web page, copy/paste the HTML and try to duplicate the appearance using CSS. Depending on the complexity of the page, this may be more advanced than you're used to, but it is probably the best practice you could get. Avoid working with HTML pages that use <table> (this is common in older web pages where the <table> was used improperly for layout hacks).
- Suggested practice problems:
    - Midterm CSS problem
    - Flexbox Froggy
    - Review HW1/HW2
    - CSBS CSS writing problems
    - CSBS CSS query selector mysteries

CSS: Common Mistakes

- Forgetting to handle small details (italic or bold text on one span element, text alignment, overflow, clear, flexbox properties, etc.).
- Improper use of flexbox (e.g., "flex-direction: row" instead of "flex-direction: column").
- Treating a block element as an inline element and vice versa. These two types of elements behave differently for different CSS properties.
- Not taking inherited relationships between elements into consideration. For example, if a inside of a has a width set to 50%, it's 50% of the width relative to the parent .

JavaScript (DOM/Events/Timers)

For problems involving JavaScript DOM manipulation and event handling, you should be comfortable implementing functionality of a page given criteria for the page's event-driven behavior. You should also be able to manage timers with setInterval/setTimeout, and be able to start/stop animations where appropriate. Note that some problems also require you to manage more than one timers. Remember to review the last two Midterm problems, and note the “Midterm Clobber Policy” this problem will apply to.

JavaScript (DOM/Events/Timers): Tips for Studying

- Review JS functions covered in class. You may also find the exam cheat sheet and lecture slides helpful for reference.
- Review different event types we’ve covered as well as how to use timers (setInterval and setTimeout).
- Review how to add/remove event listeners with addEventListener and removeEventListener, and what happens when you add multiple event listeners for the same element.
- Review how to navigate the DOM with selector functions (e.g. document.getElementById and document.querySelectorAll) and parent/children DOM navigation.
- Review how and when to append/modify/remove new elements to the page using JS.
- Suggested practice problems:
    - Midterm Problems 4 and 5
    - Lab 3 (Encrypt-It) and Lab 4 (Nonograms)
    - Week 4 Section (Problems 1 and 2)
    - CSBS event mystery problems

JavaScript (DOM/Events/Timers): Common Mistakes

- Incorrectly accessing values of different input elements (how do drop-down/select, vs. radio button vs. checkboxes vs. text input values differ?)
- Not handling "case-sensitivity" for text if relevant/specified.
- Forgetting to use parseInt on strings with integer values.
- Using one timer to when multiple are needed (recall midterm timer problem for example where multiple timers are needed) or using multiple timers when only one is needed.
- Mixing use of setInterval instead of setTimeout
- What were your mistakes on Problems 4/5 on the Midterm?

JavaScript (AJAX with JSON/Text)

This type of problem may involve using a PHP web service (e.g., "quotes.php") by:

1. Setting up the page and/or getting some value(s) from user input on an HTML page.
2. Using these values to request data from the web service using AJAX and GET and/or POST.
3. Outputting the results in HTML **using JavaScript** and DOM methods.

JavaScript (AJAX with JSON/Text): Tips for Studying

- Review and practice processing data from PHP web services which return JSON output.
- Review making AJAX GET and POST requests using fetch introduced in class (refer to lecture/section/lab slides related to AJAX for review).
- Creating and appending HTML elements into the page using JS.
- Review HW4 (BestReads) assignment.
- Suggested practice problems:
    - Practice Exam 1 (Problem 4)
    - Practice Exam 2 (Problem 5)

JavaScript (AJAX with JSON/Text): Common Mistakes

- Incorrect syntax for fetch requests (not setting body for POST requests, not processing fetch response before using it, etc.).
- Not calling your AJAX request method if needed when the page loads vs. only when responding to user input.
- Forgetting to pass parameters to the URL when building a fetch request (if applicable).
- Incorrectly processing JSON objects (e.g. forgetting JSON.parse) or using JSON.parse for plain text response types
- Forgetting to split plain text response types with “\n” where appropriate

PHP (Web Service and/or File I/O)

Many PHP exam problems ask you to implement as small web service, possibly processing directories or files. Such problems may involve working with or without query parameters to perform some function (using math, arrays, and/or files). Some ask you to handle GET and/or POST requests and output a specific response type (JSON or plain text). Some exams also have had a PHP web service problem complement a JS AJAX problem, where you write the web service your JS program fetches from.

PHP: Tips for Studying

- Review common PHP functions (refer to exam cheat sheet).
- Review array creation, processing, and modification in PHP. Arrays work pretty differently in PHP vs. JavaScript. Knowing the right methods can help save time on the exam.
- Review HW4 (BestReads) assignment.
- Review how and when to use GET vs. POST parameters (most of the time these will be specified in the problem, but it's important to understand the difference between their use cases).
- Review reading/writing files, getting file names, and checking whether files exist.
- Review building and outputting data in JSON and plain text formats.
- Review headers, content-types, and setting errors with HTTP error codes,
- Suggested practice problems:
  - Practice Exam 1 (Problem 5)
  - Practice Exam 2 (Problem 6)
  - CSBS PHP exercises (especially file processing)

PHP: Common Mistakes

- Improper use of GET and POST query parameters.
- Forgetting to use "$" for PHP variables.
- Using incorrect PHP array syntax (e.g., arr.length or arr.size() instead of count(arr))
- **Using "+" to concatenate strings instead of "."**
- Improper use of PHP glob and/or scandir and/or file functions (refer to this table of different file i/o functions).
- Forgetting to check if query parameters are set (if asked to do so).
- Forgetting to set content type with header() and printing results after processing your data.
- If asked to handle errors, missing header("HTTP/1.1 400 Invalid Request").

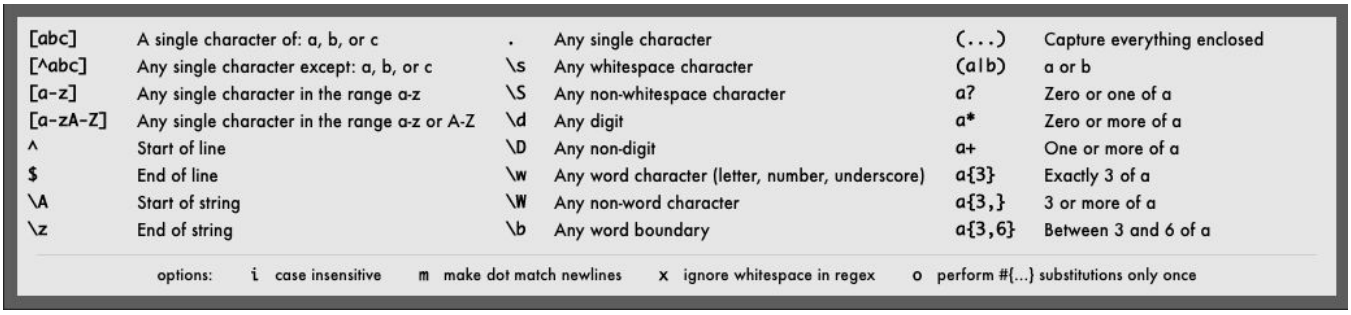
Regular Expressions

For Regex problems, expect to see a short answer question that will ask to either examine or write regular expressions (in formats similar to those in Lab 8 and provided practice exams)

Regular Expressions: Tips for Studying

- Practice! Rubular is a great resource to help test different regular expressions.
- Suggested practice problems:

- Review the practice problems given in Lab 8
- The two practice finals have great examples!
- You will have access to this regex cheatsheet in the exam (we aren't looking that you memorize these, but that you can know how a regex match works for a given pattern and test input strings)



[abc]	A single character of: a, b, or c	.	Any single character	(...)	Capture everything enclosed
[^abc]	Any single character except: a, b, or c	\s	Any whitespace character	(a|b)	a or b
[a-z]	Any single character in the range a-z	\S	Any non-whitespace character	a?	Zero or one of a
[a-zA-Z]	Any single character in the range a-z or A-Z	\d	Any digit	a*	Zero or more of a
^	Start of line	\D	Any non-digit	a+	One or more of a
$	End of line	\w	Any word character (letter, number, underscore)	a{3}	Exactly 3 of a
\A	Start of string	\W	Any non-word character	a{3,}	3 or more of a
\z	End of string	\b	Any word boundary	a{3,6}	Between 3 and 6 of a

options: i case insensitive m make dot match newlines x ignore whitespace in regex o perform #{...} substitutions only once

Regular Expressions: Common Mistakes

- Forgetting to handle "empty cases" if applicable (e.g., "" may or may not be a match for a regex).
- Forgetting what anchors ("^" and "$") mean (^ means the beginning of a string, $ is the end, so /^foo.*$ matches any string that starts with "foo" followed by 0 or more characters
- Not handling case-[in]sensitivity.

SQL

This type of problem will likely ask you to write 2-3 SQL queries on databases we've introduced in lecture, section, and/or lab (worlddb, imdb, or gamesdb) - we will provide any necessary schemas for reference on the exam. You should be prepared to write multi-table queries in this problem (e.g. with JOIN).

SQL: Tips for Studying

- Review the SQL lecture slides specific to writing queries
- Review Section 11 problems that join columns from multiple tables - these are the hardest SQL queries to write on the exam. We won't give you super super complicated join problems, but you should still be ready to write queries that join multiple tables.
- Solve at least 6 practice problems (from sections, labs, and/or practice exams) that require multi-table queries– do these on paper, and without looking at the solution.

When you're certain you've answered it correctly, check your answer on the SQL query tester.

- Create a few small SQL tables of your choice (e.g., FoodData, Pets, AnimalData, Games, etc.). Review the different types of data that may be appropriate for different attributes (e.g., VARCHAR for strings, INTEGER for integers, FLOAT for decimals, etc.).
- Review queries that that work with two *different* tables in the same query, and problems that work with two instances of the *same* table in the same query
- Suggested practice problems:
  - Section 11 Problems
  - Practice Exam 1 (Problem 7)
  - Practice Exam 2 (Problem 3)

SQL: Common Mistakes

- Forgetting to order the query result as specified.
- Referring to a column in one table that is instead used in a different table.
- Missing columns in SELECT or criteria in WHERE.

Short Answer

Don't forget to key concepts emphasized over the quarter! These often are found in the "Short Answer" section of the exam. Otherwise, we won't provide any specific tips here that we haven't already mentioned in this document, but make sure to review the midterm short answer section and you can find exams with a short answer section here (see the two provided Practice Finals, the 18au Midterm, and the three Practice Midterms).

Good Luck on the Exam!