CS 328 - Exam 1 Review Suggestions - Spring 2025

last modified: 2025-02-26

Exam 1 BONUS Opportunity

- You can receive (a maximum) ***5 POINTS BONUS*** on Exam 1 if you do the following:
  - Make a **hand-written** Exam 1 study sheet (a single sheet of paper, no larger than 8.5" by 11", on which you have hand-written as much as you would like on one or both sides).
  - Submit a photo or scan of it saved as a `.pdf`, `.png`, `.jpg`, `.gif`, or `.tiff` to Canvas **by 3:00 pm on Wednesday, March 5** such that I can read at least some significant CS 328 Exam 1 material on it.
  - You are **encouraged** to have this **with you** as you are taking Exam 1.
  - **NOTE**: if this is typed rather than handwritten, you will **not** receive bonus credit, and you will **not** be allowed to use it during Exam 1.
  - Please let me know if you have any questions about this, and I hope it helps you in reviewing course concepts more effectively before Exam 1.

Exam 1 Set-up

- You will take Exam 1 in SH 108 on Wednesday, March 5.
  - You are expected to work **individually** on the exam -- it is not acceptable during the exam to discuss anything on the exam with anyone else.
  - You may have your Exam 1 study sheet with you during the exam. Otherwise, the exam is closed-note, closed-book, and closed-computer/closed-electronic-devices.
- I expect there will be a few multiple-choice questions, and the rest will be short- to medium-answer questions.
- Your studying should include careful study of posted examples, notes, and assigned zyBooks chapters thus far.
- You are responsible for material covered in class sessions, lab exercises, and homeworks through and including the Week 5 Lab Exercise (Thursday/Friday, February 20/21) and Homework 5 (due 11:59 pm on Friday, February 28).
  - This review handout is intended to be a quick overview of especially important material.
  - You will start being responsible for PL/SQL triggers and CSS-related material for Exam 2.
  - TIP: It is **perfectly fine** to retake/read over the short-answer questions in Canvas from Homeworks 4 and 5 as you are studying for Exam 1!

    These are set up for unlimited retakes, and only keep the highest score, so you will not hurt your grade by doing so!
  - This is likewise true for the Chapters 1 and 2 zyBooks course text activities.

- While PL/SQL and SQL are not case-sensitive (except within quotes!), strict-style HTML *is*. You are expected to use the correct case (when applicable) in your answers.
- You are expected to follow CS 328 course style guidelines and coding standards in your answers.
- A packet of example code will be given out along with the exam, both for reference and for use directly in some exam questions. Because of the nature of this code (some being used directly in exam questions, for example), it cannot be made available in advance -- however, it will happen to include at least the following:
  - an *uncommented* version of `html-template.html`
  - example HTML document that happens to include at least one of each of the following elements: anchor (hyperlink), form, submit button, label, fieldset, textfield, radio button, checkbox, drop-down box
  - an example of a PL/SQL stored procedure and an example of a PL/SQL stored function
  - an example of PL/SQL exception handling
- The ability to read and make use of existing code is an important skill.
  - It is possible that you may have to diagnose what is wrong with provided buggy code, and how it might be fixed, and/or perhaps you could be asked to modify code.
  - You might be asked to complete incomplete code (you could be given partial code, and asked to complete or modify or debug it in some way).

n-Tier Architecture

- For CS 328 purposes, be comfortable with the concept of an n-tier architecture that includes a **client tier**, one or more **application tiers**, and a **data tier**.
- You should know on which of these tiers different parts of an application are **executed** (which may differ from where, say, they are requested from or generated).
- You should be starting to be able to consider on which tier different parts of an application should be handled, and starting to be able to consider pros and cons for application parts where there is more than one potential choice of tier for handling them.

Intro to HTML

- What does HTML stand for? What is the intent of HTML?
- Consider an n-tier architecture. On which "tier" is HTML *executed*?
- Note that you are expected to write "strict"-style HTML (for exam answers as well as in homework/lab submissions).
- What is an element? What is an attribute?
- What is a block element? What is an inline element? What are examples of each? What must an inline element be contained within?
- You are responsible for those HTML features that have been discussed in class and zyBooks chapters

1 and 2, as well as those HTML features that have been used in posted course examples and in homeworks.

- That includes (but is not limited to) basic HTML document structure, as well as HTML elements such as those for titles, paragraphs, headings, numbered and unnumbered lists, images, hypertext links, forms, fieldsets, labels, textfields, password fields, submit buttons, radio buttons, checkboxes, textareas, and drop-down boxes.
- What is the **`id`** attribute? What is the particular rule with regard to its value?
- You should also be familiar with the basic rules of "strict"-style HTML syntax (for example: how all tags for elements with content must be closed, the way that void (no-content) elements must be written, how attribute values must be written, the case rules, how elements must be properly nested, etc.)
- You should be familiar with HTML terminology (for example, element, attribute, tag, content, etc.) You should be comfortable with the differences between an HTML element, an attribute of an element, the value of an attribute within an element's start tag, and the content of an element.
- an *uncommented* version of the posted example `html-template.html` will be provided in a references packet along with the exam.

- What happens when an HTML form is submitted? If an HTML page has multiple forms, you should be able to tell what happens if specified actions are taken on any of the individual forms.
  - What is the difference between a form submitted using the (default) **`get`** method and one submitted using the **`post`** method?
  - Given a form whose method is **`get`**, you should be able to give the URL that would result given what has been done to the form at the time that it is submitted. (That is, what will be the name=value pairs within that URL?)

    [BUT! I do **NOT** expect you to know the codes that blanks and other special characters are replaced with in those pairs within the URL!]
  - If the form's method is **`post`**, will you still see those name=value pairs within the URL that would result when the form is submitted?
- Where would you normally place an HTML page on nrs-projects? What permissions does the HTML file need to have there? What permissions do all of the directories in that file's path need to have? What URL would you then use to access that page?
  - You are expected to be able to write Linux commands, such as you would type at an nrs-projects shell prompt, for setting up and maintaining HTML files, directories, and their needed permissions.
  - How could you write a hypertext link (**`a`** or anchor element) to another HTML page in the **same** directory?
  - How could you write an **`img`** element for an an image stored in the **same** directory?
  - How could you write an **`img`** element for an image stored in an **`images`** subdirectory of the directory this page is within?
  - Given a description of an HTML file and its location on nrs-projects, you should be able to write

an absolute URL that could be pasted into a browser to successfully display it or could be the value of an **`a`** element's **`href`** attribute to successfully link to it.

SQL

- Consider an n-tier architecture. On which "tier" is SQL *executed*?
- Note that you may be asked to write SQL `select`, `insert`, `update`, or `delete` statements on this exam, as they can be very important in database applications. Your skills (and comfort) in writing them should be increasing during the course of this semester.
- You should be able to read a SQL `select`, `insert`, `update`, or `delete` statement and, given example tables, determine what it would do; you should be able to modify and/or debug such a SQL statement.

PL/SQL

- You should be able to read, write, and execute PL/SQL stored procedures and stored functions.
- You are expected to be able to distinguish between SQL statements, PL/SQL statements that are not also SQL statements, and SQL*Plus statements.
  - Note that PL/SQL subroutines may contain both PL/SQL and SQL statements, but **not** SQL*Plus statements.
- What SQL*Plus command should you enter to be able to see PL/SQL **`dbms_output.put_line`** output?
- Consider an n-tier architecture. On which "tier" is PL/SQL *executed*?
- What does PL/SQL need to "add" to SQL, so that procedural programming will be possible?
- You are expected to be comfortable reading and writing PL/SQL stored procedures and stored functions.
  - What are the differences between a PL/SQL stored procedure and a PL/SQL stored function?
- What must you put after the end of a PL/SQL subroutine (after its body's `END;`) so that it will be created and compiled?
  - What SQL*Plus command should you enter to see the compilation errors for the latest attempt at compiling a PL/SQL subroutine?
- Executing PL/SQL subroutines:
  - How do you call a PL/SQL stored procedure from within SQL*Plus? How do you call a PL/SQL stored procedure from within another PL/SQL subroutine?
  - How do you call a PL/SQL stored function from within SQL*Plus? How do you call a PL/SQL stored function from within another PL/SQL subroutine?
- You are responsible for those PL/SQL features that have been discussed in class, as well as for those PL/SQL features that have been used in posted course examples and in homeworks.
- Given PL/SQL code, example tables, and example calls, you should be able to tell what would

happen. Given error messages or errant behavior, you should be able to debug PL/SQL and SQL.

- How can you call a PL/SQL stored procedure or stored function that expects parameters? How can you call one that does not expect any parameters?
- Make sure you know how to do the following:
  - create and compile a PL/SQL subroutine
  - find the compilation errors in a subroutine
  - execute a stored procedure or stored function (at both the SQL*Plus level and within a PL/SQL subroutine)
- Make sure you know:
  - the basic skeleton of a PL/SQL subroutine
  - how to specify a parameter being passed to a stored procedure or stored function
  - where you need to declare local variables within a PL/SQL subroutine, and how to declare them
  - how to write a PL/SQL assignment statement
  - how to write a **`select`** statement that puts values into local variables
  - how to use **`dbms_output.put_line`** (and what SQL*Plus statement is needed to be able to see its output)
  - how to use the concatenation operator for strings
  - how to write an **`IF`** statement
  - how to write a **`WHILE`** loop, "**classic**" **`FOR`** loop, and **cursor-controlled** **`FOR`** loop

PL/SQL Exception Handling

- How can you handle exceptions within a PL/SQL subroutine?
- You should be comfortable with the syntax and semantics of PL/SQL exception handling; you should be able to read and write PL/SQL code that includes exception-handling.
- What is a pre-defined exception? What are some of the common Oracle pre-defined exceptions, and when are they raised?
  - (For Exam 1, it will suffice to be familiar with the pre-defined exceptions **`TOO_MANY_ROWS`** and **`NO_DATA_FOUND`**; if others are used, they will be described accordingly.)
- You should be able to write PL/SQL code that performs desired actions when a particular exception occurs.