CS 111 - Exam 2 Review Suggestions - Fall 2024

last modified: 2024-11-06

Exam 2 BONUS Opportunity

- You can receive (a maximum) ***5 POINTS BONUS*** on Exam 2 if you do the following:
  - Make a **hand-written** Exam 2 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`, or `.tiff` to Canvas **by 9:00 am on Thursday, November 14** such that I can read at least some significant CS 111 ***specifically-for-Exam-2*** material on it.
  - You are **encouraged** to have this **with you at** as you are taking Exam 2.
  - **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 2.
  - 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 2.

Exam 2 Set-up

- You will take Exam 2 in GH 218 on Thursday, November 14.
  - 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 2 study sheet on hand during the exam. Otherwise, the exam is closed-note, closed-book, and closed-computer/closed-electronic-devices.
- I expect there will be some multiple-choice questions, and the rest will be short- to medium-answer questions.
  - You will be reading and writing C++ expressions, including C++ function and named constant definitions.
  - You will be answering questions about concepts as well.
  - The questions **will all involve C++**, although many of the concepts **overlap** with those from the first exam (since we covered many of the same concepts in Racket and in C++).
  - Indeed, to emphasize the common concepts, note that *some* Exam 2 questions *may* be simply C++ versions of questions from Exam 1.
- Your studying should include careful study of posted examples and notes thus far.
- You are responsible for material covered in class sessions, lab exercises, and homeworks through and including Week 10, Lecture 2 (Thursday, October 31), the Week 10 Lab Exercise (Friday, November 1), and Homework 9 (due 11:59 pm on Friday, November 8).

  - This review handout is intended to be a quick overview of especially important material.
  - TIP: It is **perfectly fine** to retake/read over the short-answer questions in Canvas from Homeworks 7 through 9 as you are studying for Exam 2!

    These are set up for unlimited retakes, and only keep the highest score, so you will not hurt your grade by doing so!
- Remember that C++ **is case sensitive** - for example, `String` is not the same data type as `string`. You are expected to use the correct case in your answers.
- You are also expected to follow **CS 111 course style guidelines and coding standards** in your answers (**including indentation**).
- A reference page will be given out with Exam 2 (*in addition to* your optional handwritten page of notes); this is intended both for reference and for use directly in some exam questions.
  - This will include a copy of the posted `111template.cpp`.
    - I believe that the ability to use such a reference effectively is an important skill.

C++ basics

- What is a simple expression in C++? What is a compound expression in C++? Should be able to read these, write these, tell the type of a given expression, write an expression of a given type.
- (You are expected to be familiar with the C++ types discussed so far, including knowing their C++ type names.)
- How can you write a comment in C++?
- You should know the difference between an expression and a statement; you should know how a statement is ended in C++.
  - Note that, often/usually, a C++ **statement** is ended with a semicolon (unless you are talking about a block, `{ }`).
  - An **expression**, which has a value of some data type, is typically within a C++ statement, and so should **not** end with a semicolon.
- Need to be able to write a C++ function using the design recipe! Need to be able to write the design recipe steps, in the right order!
  - How does a C++ signature differ from a Racket signature?
  - Need to be able to write a C++ function header.
  - Need to be able to write appropriate specific example/test `bool` expressions for C++ functions.
    - You should write these as `bool` expressions within a comment after the purpose statement for a function.
  - Need to be able to write a C++ function body.
  - Need to know how to place each example/test `bool` expression into a `cout` statement within a `main` function, to actually execute that example/test and print whether it passed or not to the

screen.

  - Need to be comfortable reading, designing, writing, testing, and calling/using C++ functions.
- Need to follow the course style standards.
- Need to be comfortable with C++ identifiers and C++ literals.
- Should be able to write a named constant declaration in C++.
- What types have we discussed so far in C++? How can you write literals of (most of) these? How would you declare variables of each?
- You should be comfortable with the C++ types `string` and `char*`.
  - You are expected to be comfortable with C++ `char*` literals (anything written within double quotes); note that they CAN be assigned to variables of type `string`.
  - You should be able to declare new-style C++ `string` variables (`string`, made available by using `#include <string>`)
  - (And note that it is a course style standard that, when you want a function to have a string parameter or return a string value, you are expected to use type `string` rather than `char*`.)
  - You should be comfortable reading and writing C++ statements and expressions using the `string` methods `length`, `at`, and `substr`, and concatenating `string` instances using `+`; given a description of a "new" `string` method, you would be able to write C++ statements and expressions using that method.
- What are the basic arithmetic operators of C++? What do we mean by operator precedence? How do you write the relational operators in C++? ...the boolean operators? What happens when you divide two `int`s?
- Given a function header, you should know how to then write a syntactically-correct compound expression calling that function.

C++ `if` statement

- Need to be comfortable reading and writing C++ `if` statements
- You should be able to write these using the course-required indentation.
- (And you still need to be able to write an appropriate set of examples for a function involving multiple categories of data -- need an example for each category, and for the boundaries between those categories!)
  - So, to make this point, I also could ask at least how many tests should be written for a particular function.

example of a side-effect: screen output (`cout`)

- Know that a `cout` statement's printing to the screen is *not* the same as a function's `return` statement returning the value for a function call.

- Should be able to write a `cout` statement that could appear in a testing `main` function to print s message that you are testing a particular function.
- Should be able to write a `cout` statement that could appear in a testing `main` function to print the value of one of a function's test expressions to the screen.
- Should be able to write a `cout` statement that could appear in a `main` function to print to the screen the value of compound expression calling a function.

"complete" C++ programs

- what is a C++ program? what function must be included in a C++ program? There are several acceptable headers for this function; what is the one that we have been using? (`int main()`)
  - Given the CS 111 course style standards, what is this function expected to return?

    ```
    return EXIT_SUCCESS;
    ```
- Note that the Exam 2 References will include the posted `111template.cpp` from the course Canvas "home" page.
  - Using this template, you should be able to write a "testing" `main` function that meets the class style standards for testing a non-`main` function.
- Given the name of a C++ source code `.cpp` file, you should be able to write the command you would use in a CS50 IDE Terminal to **compile** and create an executable program from that,
  - and, be able to write the command you would use in a CS50 IDE Terminal to **run** the resulting executable program.