CS 112 - Exam 2 Review Suggestions - Fall 2022

last modified: 2022-10-19

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.
  - Submit a photo or scan of it saved as a .pdf, .png, .jpg, .gif, or .tiff to Canvas **by 9:00 am on Friday, October 28** such that I can read at least some significant CS 112 Exam 2 material on it.
  - 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.
  - You are **encouraged** to have this **at hand** as you are taking Exam 2. (It is fine for you to have your Exam 1 study sheet on hand as well.)

Exam 2 Set-up

- You will take Exam 2 in Canvas while you are in BSS 317 on Friday, October 28.
  - You are **required** to be in BSS 317 while you are taking Exam 2.
    - If you attempt to take Exam 2 from anywhere else, you will receive a grade of **0** on the exam.
    - The instructor will take roll at the beginning of lab.
  - It will be set up with a **time limit** of **1 hour 50 minutes**, and will be available from **9:00 - 10:55 am** on Friday, October 28.
  - It will be set up such that **you may only attempt Exam 2 once**.
  - Let the instructor know of any technical difficulties, so they can make provisions as needed!
  - Exam 2 will be set up such that **you will be shown one question at a time**,
    - BUT there will be a list of question-links on the right-hand-side of the Canvas screen, and you can go back and forth between questions during that **one** exam attempt.
  - 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 look up information from your Exam 2 study sheet, your Exam 1 study sheet, personal notes, on-line, or from the course textbook during the exam, but note that **if you take too long looking up material, you may have trouble completing the exam during the time period**.
  - 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, statements, and fragments, including C++ function and class definitions.

    - You will be answering questions about concepts as well

- A link to a packet of references and additional instructions - intended for use with Exam 2 - will be linked from the Exam 2 Instructions.
  - So, you can have it open in another browser window while you are taking Exam 2.
  - This is intended both for reference and for use directly in some exam questions.
- Your studying should include careful study of posted examples and notes thus far.
- By necessity, we have been building on and making use of previous material in much of the new material since Exam 1.
  - But, **in general**, the **FOCUS** of most of the questions on this exam will be the material covered since Exam 1.
  - So, the **focus** of most of the questions will be on material covered:
    - from Week 4 lectures through **Week 9 lectures**;
    - from the Week 4 Lab Exercise through the **Week 9 Lab Exercise** (Friday, October 21)
    - from Homework 4 up through and including **Homework 7** (due 11:59 pm on Friday, October 21).
  - This review handout is intended to be a quick overview of especially important material.
  - The Savitch text is very comprehensive; references below to chapters in the text are there just to point out where in the text they are. You will NOT be responsible for all information in those chapters, just the parts we've covered in lectures, labs, and assignments.

    With that in mind:
    - Chapter 7 for arrays,
    - Chapter 10, Sections 10.2 and 10.3 - intro to writing classes
    - Chapter 5, Section 5.2 - pass-by-reference parameters
    - Chapter 9 - intro to pointers and dynamic memory management
    - Chapter 11, Section 11.4 - intro to the "big-3", destructor, copy constructor, and overloaded assignment operator
    - Chapter 8, Section 8.3 - intro to vectors

    ...can be useful for additional background reading.
  - TIP: It is **perfectly fine** to retake/read over the short-answer questions in Canvas from Homeworks 1 through 8 (particularly those for Homeworks 4 through 8) 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 112 course style guidelines in your answers (**including indentation**).
  - You should use the **Formatting Practice Question** linked from the course Canvas site's Home page to practice writing formatted C++ statements BEFORE Exam 2!
  - If you find you are having trouble with this, make sure to come by student hours or ASK ME before Exam 2, so you won't lose points for poorly-indented answers.

more on Arrays

- (Chapter 7 of the course Savitch text might be useful if you would like additional reading on this topic.)
- Be able to write a function with an array parameter; also know how to call such a function.
- As you know, when you pass an array parameter using pass-by-value, you can change the argument array that array parameter references.
  - If such a function is **not** intended to change its array parameter's corresponding array argument, what modifier should you put before its array parameter's declaration?

note: `void` functions and methods

- (Chapter 5, Section 5.1 of the course Savitch text might be useful if you would like additional reading on this topic.)
- Be able to read, write, and answer questions about non-`main` functions and methods that do NOT return a value.
  - What should be the return type for such functions and methods?
- Should be able to write a correct call to such a function or method given its header.

Intro to writing your own C++ classes

- (Chapter 10, Sections 10.2 and 10.3 of the course Savitch text might be useful if you would like additional reading on this topic.)
- An object is a variable that has operations as well as data associated with it -- in C++, when you create a class type, an instance of that class is an object.
  - These actions associated with an object are called methods (sometimes also called member functions).
  - Be able to call a method of an object.
- Need to be comfortable defining and implementing a C++ class; need to be comfortable reading and using C++ classes.
  - You should be able to write a class definition and implement that class, given specifications of what is desired;

- Given a C++ class, you should know how to declare instances/objects of that class. You should be able to call methods of the class using either object identifiers or pointers to an object.
  - How do you declare an object using a zero-argument constructor? ...using a constructor that has arguments?
  - How do you write a function that expects an object as a parameter? How do you write an object argument for such a function when calling that function?
- What is the purpose behind having a public part and a private part of a class definition? What is normally placed in the public part of the the class definition? In the private part of a class definition?
- What is a class data field (also called a class member variable?)
- What is a class method (also called a class member function)?
- With what visibility is it considered good style to declare class data fields?
  - How can you refer to such a data field within a method of that class?
  - How might another function (NOT a method) access a data field, if the class permits?
- What is a class constructor?
  - What is the name of a class constructor?
  - What is distinctive/different about a constructor's header?
  - It is considered good style for each constructor to do... what?
- Need to know what is meant by overloading a method within a class, and how to do it;
  - Why is it considered common practice to overload constructors in a class?
  - What kind of constructor is it considered good style to regularly explicitly include in a class, along with whatever other constructor(s) would be useful for the users of that class?
- What is an accessor/getter method? What is such a method's purpose? What is our class style for naming these? What is the typical visibility for these methods?
- What is a mutator/setter method? What is such a method's purpose? What is our class style for naming these? What is the typical visibility for these methods?
- And there can be methods that are neither constructors, accessors, or mutators -- and these can have visibility of `public` or `private`.
  - When might you want to define a private method?
  - Note: You will **not** be asked any questions about `protected` visibility.
- Need to be able to write a class definition; need to be able to write implementations for a class' methods.
  - What is the scope resolution operator `::` used for when implementing a class' methods?
- Given a class definition, you should be able to use that class: declare instances of it (using both zero-argument and non-zero-argument constructors), call its selector methods, mutator methods, and "other" methods.

Pass-by-value and pass-by-reference parameter passing

- (Chapter 5, Section 5.2 of the course Savitch text might be useful if you would like additional reading on this topic.)
- What is meant by a function argument being **passed or called by value**? What is passed to the parameter in that case?
  - How do you write a pass-by-value parameter within a function header?
  - How do you write an argument for a pass-by-value parameter when calling that function?
  - If you happen to change a (non-pointer) pass-by-value parameter within its function body, what happens to its corresponding argument when that function is called?
- What is meant by a function argument being **passed or called by reference**? What is passed to the parameter in that case?
  - How do you write a pass-by-reference parameter within a function header?
  - How do you write an argument for a pass-by-reference parameter when calling that function?
  - If you happen to change a pass-by-reference parameter within its function body, what happens to its corresponding argument when that function is called?

Intro to pointers, dynamic memory management, and dynamic arrays

- (Chapter 9 of the course Savitch text might be useful if you would like additional reading on this topic.)
- What is a pointer? What does it do? How can it be used?
- How do you declare a pointer variable in C++? What are some of the ways you can initialize it? Whats are some of the ways you can set it?
- According to CS 112 course style standards, if a pointer is currently not "pointing" anywhere, then to what special value should it be set?
  - [NOTE: CS 112 course style standard: we use `NULL` for this! 8 -)]
- How can you make a pointer point to something?
- How can you change or use what a pointer points to?
- How can you access the contents of what a pointer points to?
- I could ask a question involving the boxes-and-arrows notation used in lecture;
  - for example, given a code fragment and corresponding boxes-and-arrows, you might have to say what would be in a given box afterwards, or to what box a given arrow would point afterwards.
- If a pointer is pointing to an object, how can -> be used to call the methods associated with the object that the pointer is pointing to?
- How how can you dynamically allocate memory from the heap/freestore and have a pointer point to

that memory?

  - You should be able to do this for scalar values (single values like `int`, `bool`, `double`, etc.), arrays, and objects;

- How do you free (or deallocate) this memory when you are done using it? What is the special syntax needed if you are doing this for a dynamically-allocated array?
  - What might happen if you do not do this?

Intro to the "big-3", destructor, copy constructor, and overloaded assignment operator

- (Chapter 11, Section 11.4 of the course Savitch text might be useful if you would like additional reading on this topic.)
- Know that these so-called "Big-3" methods -- destructor, copy constructor, overloaded assignment operator -- are needed when a class' data fields involve dynamic memory.
- What is a destructor? What is the method name for a destructor method? What is its header? When is it called? What should it do?
- What is a copy constructor? What is the method name for a copy constructor method? What is its header? When is it called? What should it do?
- What is an overloaded assignment operator? What is the method name for using for implementing an overloaded assignment operator for a class? What is its header? When is it called? What should it do?

Intro to C++ `vector` class

- (Chapter 8, Section 8.3 of the course Savitch text might be useful if you would like additional reading on this topic.)
- What `#include` directive is needed to use vectors?
- What is a vector in C++?
- How can you declare a vector in C++? How can you add something to a vector instance? How can you access something in a vector? How can you remove an element from a vector?
  - How can you find the size (number of elements in) a vector?
  - How can you find the current capacity of a vector? What is the difference between a vector's size and a vector's capacity?
- In terms of their size while a program is running, what is an important difference between arrays and vectors?