CS 112 - Final Exam Review Suggestions - Fall 2022

last modified: 2022-12-07

Final Exam BONUS Opportunity

- You can receive (a maximum) ***5 POINTS BONUS*** on the Final Exam if you do the following:
  - Make a **hand-written** Final Exam study sheet.
  - Submit a photo or scan of it saved as a .pdf, .png, .jpg, .gif, or .tiff to Canvas **by 12:40 pm on Thursday, December 15** such that I can read at least some significant CS 112 **post-Exam-2** material on it.

    (for example: inheritance, calling a base class' version of a redefined method, exception handling, multiple inheritance, polymorphism)
  - Please let me know if you have any questions about this, and I hope it helps you in reviewing course concepts more effectively before the Final Exam.
  - You are **encouraged** to have this **at hand** as you are taking the Final Exam, along with your Exam 1 and Exam 2 study sheets.

Final Exam Set-up

- You will take the Final Exam in Canvas while you are in BSS 317 on Thursday, December 15.
  - You are **required** to be in BSS 317 while you are taking the Final Exam.
    - If you attempt to take the Final Exam from anywhere else, you will receive a grade of **0** on the exam.
    - The instructor will take roll at the beginning of the Final Exam period.
  - It will be set up with a **time limit** of **1 hour 50 minutes**, and will be available from **12:40 - 2:40 pm** on Thursday, December 15.
  - It will be set up such that **you may only attempt the Final Exam once**.
  - Let the instructor know of any technical difficulties, so they can make provisions as needed!
  - The Final Exam 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 Final Exam study sheet, 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 definitions.
    - You will be answering questions about concepts as well

- A link to a packet of references and additional instructions - intended for use with the Final Exam - will be linked from the Final Exam instructions.
  - So, you can have it open in another browser window while you are taking the Final Exam.
  - 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.
- The Final Exam is CUMULATIVE!
  - if it was fair game for Exam 1 or Exam 2, it is fair game for the Final Exam;
  - Thus, using the posted review suggestions for Exam 1 and Exam 2 in your studying for the Final Exam would be a **GOOD** idea.
    - Note that these are still available on the public course website's "Homeworks and Handouts" page, http://nrs-projects.humboldt.edu/~st10/f22cs112/112handouts.php
- You are responsible for material covered in class sessions and homeworks.
  - This review handout is intended to be a quick overview of especially important material since Exam 2.
  - Again, 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.
  - TIP: It is **perfectly fine** to retake/read over the short-answer questions in Canvas from course Homeworks as you are studying for the Final Exam!

    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 the Final Exam!
  - If you find you are having trouble with this, make sure to come by student hours or ASK ME before the Final Exam, so you won't lose points for poorly-indented answers.

Intro to linked lists

- (Chapter 13 of the course Savitch text might be useful if you would like additional reading on this topic.)
- What is the basic structure of a node in a classic, singly-linked C++ linked list?
  - Be comfortable with how to write, use a node class for a typical singly-linked list.
- What data field is typically included in a node class to allow its instances to be used in a linked list?
  - What other data field(s) are also typically included?
- What is a linked list frequently defined using? (A linked list is frequently defined as a pointer to its first element, often called a head pointer.)
  - (But there are other approaches, also -- for example, defining a linked list as a pointer to a dummy head node, and the actual data nodes followed that dummy head node. You will NOT be tested on this approach, though.)
- Not all linked lists define one -- but, if you had one, what would a tail pointer typically be considered as pointing to?
- You should be comfortable with the typical conceptual drawing/picture of a linked list (with boxes and arrows).
- How do you access data field(s) in a linked list node?
- If you have a pointer to a node, how to you access the public parts of the node pointed to by that pointer?
  - What happens if you try to do the above for a pointer whose value is `NULL`? How should you avoid doing this?
- You should know how to make a pointer pointing to one node in a linked list be changed to point to the next node in that linked list.
- You should be comfortable with typical actions on singly-linked lists that we have discussed -- for example, "walking through" a linked list, adding something to the beginning of a linked list, adding something after a given node, searching for something in a linked list, printing the contents of a list, etc.
- You should be able to reason, compare/contrast about linked lists versus arrays.

Intro to Inheritance

- (Chapter 15 of the course Savitch text might be useful if you would like additional reading on this topic.)
- What is a base class? What is a derived class?
- What are some of the benefits of derived classes? Why might you want to derive a class from another class?
- When is it appropriate to derive one class from another? [when specialized instances of a kind of

thing are related to instances of another kind of thing via an IS-A relationship]

- Given code for a derived class and for a base class:
  - You should be able to say which is the base class, and which is the derived class.
  - You should be able to say what is inherited by the derived class.
  - You should be able to write declarations of objects of both base and derived classes.
  - You should be able to write appropriate method calls for objects of both base and derived classes.
- Given a base class, and a description of a desired class that should be derived from that class, you should be able to write the declaration of that derived class (its `.h` file); you should be able to write its implementation (its `.cpp` file).
  - You should be able to declare appropriate constructor methods for the derived class.
  - You should be able to implement appropriate constructor methods for the derived class, making proper calls of the base class' constructor so that the inherited private data fields are appropriately initialized.
  - When writing implementations for a derived class' methods, can those methods access inherited private data fields directly? [No, they can't.] So how can they access those private data fields?
- What does it mean to redefine an inherited method in a derived class?
  - You should know how to redefine an inherited method in a derived class' declaration (in its `.h` file), and in its implementation (in its `.cpp` file).
- In the derived class' methods, you should know how to call the base class' version of a redefined method if that is desired.
  - When using an object of a derived class, you should know how to call the base class' version of a redefined method if that is desired.

Difference between redefined and overloaded

- What do we mean when we say that a method can be overloaded?
- What do we mean when we say that a derived class can redefine a method from its base class?

Exception handling in C++

- Should understand the basics of exception handling in C++.
  - Remember, there are some homework short-answer questions related to this.
- If a statement within a logical section of code might throw an exception, and you would like to catch and handle that exception, within an instance of what thing should you place that logical section of code?
- Within an instance of what thing would you place code handling a particular type of exception?
- What statement can you use to throw an exception in C++?

- In C++ what data type(s) can be thrown as an exception?
- What are the options for how a C++ exception may be caught? Unless there is a good reason not to, which of these is recommended by `isocpp.org`'s FAQ on exceptions?
- If a `try` block is followed by multiple `catch` blocks, and an exception is thrown, which `catch` block's actions will be done?
- Be aware of the C++ standard library `exception` class, and what you need to `#include` to use it or its also-available derived classes.

Inheritance and Data Types

- (Chapter 15 of the course Savitch text might be useful if you would like additional reading on this topic.)
- You have an object of a derived class. What is/are the type(s) of that object?

  You have an object of a base class. What is/are the type(s) of that object?
- Can you assign a derived class object to a variable declared to be an instance of its base class?

  Can you assign a base class object to a variable declared to be an instance of a class derived from it?
- A function expects a parameter whose type is of a base class.
  - Can you call that function with an argument that is an instance of a class derived from that base class?
- A function expects a parameter whose type is of a derived class.
  - Can you call that function with an argument that is an instance of the base class that derived class is derived from?

Polymorphism, dynamic/late binding, `virtual` methods, multiple inheritance, abstract classes

- What is **polymorphism**? ["...the ability to associate MULTIPLE meanings to ONE [method] name by means of a special mechanism called **LATE BINDING** [dynamic binding]"
  - How can `virtual` be used to make late binding/dynamic binding possible for a method?
  - How can you write an expression that can use late binding/dynamic binding?
- What is multiple inheritance?
  - Make sure you can tell the difference between a class with multiple parents and a class that is part of a "chain" of single inheritance.
  - How can you define a derived class that has multiple parent classes, multiple base classes?

    `[ class NewClass: public Parent1, public Parent2 ]`
  - What are the types of an object of a derived class with multiple parents?

  - If a derived class has multiple parents and those parent have a common public method, how can an object of the derived class call that method?
  - What is the diamond problem? What is a recommended solution for the diamond problem?
- What is an **abstract** class? What is a pure virtual method? What is the pure specifier?
  - Can you create instances of an abstract class?
  - If a derived class does not implement all of the pure virtual methods of its parent, what is also true, then, of that derived class?
  - What is a pure abstract class?