CIS 130 Final Exam Review Suggestions

* last modified: 5-02-07, 10:30 am

* remember: YOU ARE RESPONSIBLE for course reading, lectures/labs, and especially anything that's been on a homework, in-lecture exercise, or lab exercise; BUT, here's a quick overview of especially important material.

* The final exam is **cumulative** in CONCEPTS,

  but the LANGUAGE of the exam will be C++, to reduce cross-syntax confusion.

  You will **not** be writing any Python for the final exam.

  * so, you should still use the review suggestions for Exam #1 and Exam #2 for studying for the final exam, but substitute "C++" for "Python". (Note that they are still available from the public course web page, under "Homeworks and Handouts".)

  * you should still use your Exam #1 and Exam #2 for studying for the final exam, but imagine how those answers would look written in C++. Some questions may be similar in style to those asked on the first two exams, except you would answer them using C++.

    (ASK ME if there are test questions for which you are NOT SURE how they would look in C++!)

* you are permitted to bring into the exam a single piece of paper (8.5" by 11") on which you have **handwritten** whatever you wish. This paper must include your name, it must be handwritten by you, and it will **not** be returned.

  Other than this piece of paper, the exam is closed-note, closed-book, and closed-computer, and you are expected to work individually.

  (Studying beforehand in groups is an excellent idea, however.)

* Note that final exams are *not* returned. I will retain them for at least 2 years, and you may come by my office to see them if you wish after grades are posted into Moodle.

* You are expected to follow course style standards in your exam answers, and there may be exam questions about course style standards as well.

* You should still be comfortable with the design recipe for functions, and should be able to appropriately fill in the opening comment block "templates" we've been using

(and the main and .h file templates as well).

Note: the final exam handout will include the main() function template and .h file template posted on the public course web page.

* note that answers may lose points if they show a lack of precision in terminology; for example, if I ask for a literal or an expression and you give an entire statement, instead; or, if a statement is requested that requires a semicolon, and it is not ended with one.

* this will be a pencil-and-paper exam, but you will be reading and writing C++ code, statements, and expressions in this format.

    * note that you could be asked to write C++ expressions and functions;

      (note, too, that answers may lose points if they show a lack of precision in terminology; for example, if I ask for a literal and you give a whole expression, instead)

    * note that I could ask you what given C++ code does or means; I could give you one or more expressions, a function, etc., and ask you what it does or what would then be shown if it were executed (or how you could write a call to use it, etc.)

    * you could be asked to modify a piece of code or function or module, or to correct a segment of code or a function or a module, as well;

* note: you are still expected to be comfortable with and use the design recipe; you are still expected to be familiar with and use the course style standards (both those you needed to know for the first exam, and those that have been added during this time period).

* (note that there could be questions about the class style standards, and your grade for test questions may be affected by whether you follow those class style standards in your answers.)

* (note that there could be questions where you are given code and/or design recipe elements, and asked if they are "legal", if they meet class coding standards, if they are syntactically or logically correct; you might have to correct code or design recipe elements that are "incorrect" or "illegal" in some sense.)

* what are the four basic structures of programming?
* know what an **algorithm** is; know what **pseudocode** is.
    * algorithm: a finite sequence of steps for solving a problem;
    * pseudocode: an algorithm expressed in natural-language-ish steps

*** high points from the material SINCE Exam #2:**

* **separate compilation of C++ functions**

* Given a .h file template (such as that on the public course web page) or an example of a .h file, you should be able to write a .h file for another auxiliary function.

* at this point, you have written "pure" functions that accept parameters and return a result;

you have also written C++ main functions, as well as auxiliary functions that are not so "pure" (they may have side-effects, they may not return anything, they may require no parameters, they may change the values of pass-by-reference parameters, etc.!)

- * you should know the difference between a function **returning** something and function **printing** something to the screen; you should be able to write functions that can do either, depending on what is specified.

- * you should know the difference between a function **accepting** something as a parameter and a function **prompting** a user for input (and then **reading** in such input); you should be able to write functions that can do either, depending on what's specified.

- * you should know the difference between an **expression** and a **statement**; you should know how a **statement** is terminated in C++.

- * Given a function header, you should know how to then write a "legal" call to that function;
  - * If a function is a void function, how is it called?
  - * If a function accepts no parameters, how is it called?
  - * If a function returns a value, how is it (typically) called?
  - * If a function expects one or more parameters, how is it called?

* how can you just compile a C++ function at the cs-server command line? Given all of the files involved in a C++ program, how can you link and load and create a C++ executable for that program?

* what function must be included in a C++ program? what is one of the accepted headers for this function? Given course style standards, what is this function expected to return?

* What statement is used to perform interactive input into a C++ program? What statement is used to perform interactive or screen output from a C++ program?

* Be prepared to give the precise output of fragments of C++ code; you should be comfortable knowing how cout will "behave" with endl's, literals, and other expressions.

    * I could give you a "grid" of squares, and ask you to write out precisely what would be displayed, 1 character per square, to see if you know;

* You may also be expected to modify existing code, add to existing code, and/or correct erroneous code.

* you are expected to be comfortable with C++ string literals (anything written within double quotes);

    * you should be able to declare both new-style C++ string variables (**string**, and #include <iostream>) and old-style C-strings (**char[size]**, as parameter **char***)

    * note that C++ string literals are old-style C strings, but can be assigned to **string** variables (C++ will convert them as part of the assignment). However, passing them as parameters is another matter...! (sigh!)

* the **char** type

    * how do you write a **char** literal?

    * how do you declare a variable of type char?

    * you should be able to write a function that takes a **char** parameter; you should be able to write a function that returns a value of type **char**. You should be able to declare a variable of type **char**, and should be able to set a **char** variable's value appropriately.

* **switch statement**

    * another C++ statement that implements a kind of **branch** structure!

    * what is the syntax of the C++ switch statement? What are its semantics?

    * should be very comfortable with the course-expected indentation for switch statements.

    * what is the purpose of **break** within a switch statement? What happens when it

is omitted?

- what are legal types for the control expression and the case expressions within a switch statement?

- be comfortable reading switch statements; you should be able to read a switch statement, and tell what it is doing; you should be able to give its output.

* **for statement**

  - another C++ statement that implements a kind of **repetition** structure!

  - what is the syntax of the C++ for statement/for loop? What are its semantics?

  - should be very comfortable with the course-expected indentation for **for** statements/**for** loops;

  - when are **for** loops appropriate? Should be able to decide when a **while** loop is more appropriate, and when a **while** loop is more appropriate;

    - you should be able to convert from a count-controlled while loop to the equivalent for-loop, and vice versa;

  - be comfortable designing, reading, and writing for loops; you should be able to read a for loop, and tell what it is doing; you should be able to give its output.

* **arrays**

  - what is an array in C++? (be careful -- remember, it is more restricted that Python arrays/lists)

  - expect to have to read, write, and use arrays; you should be comfortable with array-related syntax and semantics, and with common "patterns" for using arrays.

  - what 3 pieces of information does the compiler need to know to declare an array? how do you declare an array in C++?

  - how can you initialize an array? (there is more than one way...)

  - what is an array index? what is the size of an array? if you know an array's size, what are the indices of that array? what is the type of an array?

  - how do you write an expression representing a single element in an array? how do you write an expression representing the entire array?

- how do you pass an array as an argument? how do you declare an array as a parameter?

- how can you do something to every element within an array? how can you use every element within an array? what particular statement is especially useful in doing such actions?

* **file input/putput**

  - why might you want a program to be able to read from a file? why might you want a program to write to a file?

  - what C++ standard library is used for the file input/output that we used?

  - how do you set up and open a file for reading in C++? how do you set up and open a file for writing in C++? how do you set up and open a file for appending in C++?

    - ...and how do you close such a file stream when you are done?

  - once you have opened an input filestream, how can you read something from it?

  - once you have opened an output filestream, how can you write something to it?

* **prefix increment operator**, **postfix increment operator**

  - What are the prefix and postfix increment operators (++)? How are they written? Where are they written? What are their effects and their semantics?

  - Be able to accurately read, write code fragments containing them;

  - (You should be able to handle the prefix and postfix decrement operators (--) also.)

* **+=, -=, *=, /=**

  - What do +=, -= *=, /=, %= mean? How are they used? What are their effects and semantics?

  - Be able to accurately read, write code fragments containing them, too;

* **pass-by-value** and **pass-by-reference**

    * What is a pass-by-value parameter? What is a pass-by-reference parameter? How can you tell them apart? What is the difference between them? What is the difference in the possible *arguments* between pass-by-value and pass-by-reference parameters?

    * EXPECT at least one question involving pass-by-value and pass-by-reference parameters.

    * input parameters, output parameters, input/output parameters --- what are they? for each, what, in general, type of parameter passing is most appropriate?

    * in terms of class style standards, when is it more appropriate to use pass-by-reference? when is more appropriate to use pass-by-value?

    * by default, how are arrays passed? what is the other way that arrays can be passed? How do these differ?