CIS 130 - Final Exam Study Suggestions

last modified: 05-04-10

* Remember: anything covered in lecture, in lab, or on a homework is FAIR GAME.

* 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.

* 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 Scheme for the final.

  * so, you should still use the review suggestions for Exam #1 and Exam #2 for studying for the final exam, but substitute "C++" for "Scheme" in the Exam #1-related material. (Note that these review suggestions 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 Exam #1 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++.

* 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 posted on the public course web page.

* note that answers may lose points if they show a lack of precision in terminology or syntax; 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.

repetition

* need to be comfortable with the basics of the C++ while loop and for loop statements; need to be comfortable with their syntax and semantics, need to understand how they use mutation of a local variable to control repetition;

* should be able to read, write a count-controlled loop (using both a while-loop and a for-loop), a loop that does something a certain number of times;

* when is a for loop appropriate?

* you should know that while loops are better for **sentinel-controlled loops**, and for "other" kinds of loops.

* you should be comfortable with **sentinel-controlled loops**

    * you are expected to know, and use, the **"classic" structure for a sentinel-controlled loop**, when that logic is what is desired;

* should be very comfortable with the course-expected indentation for while loops and for loops;

* you should be able to design, read, and write while loops and for loops; you should be able to read a while loop or a for loop, and tell what it is doing; you should be able to answer questions about what a while loop or for loop does when it executes

arrays

* need to be comfortable with the basics of C++ 1-dimensional arrays

* how do you declare an array? how can you initialize it? what are its indices? How do you access an individual element?

* 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.

* how do you pass an array as an argument? how do you declare an array as a parameter? In C++, when an array is a parameter for a function, what should also be a parameter of that function?

* 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?

* expect it: you will have to write at least one loop that does something involving every element in an array!

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?

* should understand how pass-by-value and pass-by-reference work;

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

* if you change a pass-by-value scalar parameter, is the corresponding argument changed? why or why not?
* if you change a pass-by-reference scalar parameter, is the corresponding argument changed? why or why not?
    * arrays are passed by value -- yet if you change the contents of a parameter array, the argument array is changed. Why?

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

also note...

* **expect** at least one question in which you are given function headers and asked to write appropriate calls of those functions;