CIS 130 Exam #2 Review Suggestions

* last modified: 11-11-05, 12:32 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 test particularly focuses on the HtDP reading packet sections 4-5, and intro to while loops (Ch. 12, 2nd text). (The Exam #1 material still applies --- we have built atop it, so it cannot be avoided! --- but the focus of most questions will be the "new" material.)

* 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.

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

  * (note, too, 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)

  * 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.)

* the **bool** type
    * what are the only two **bool** literals?
    * how do you declare a variable of type bool?
    * what kinds of expressions return values of type **bool**?
    * you should be able to write a function that takes a **bool** parameter; you should be able to write a function that returns a value of type **bool**. You should be able to declare a variable of type **bool**, and should be able to set a **bool** variable's value to be the result of a conditional/Boolean expression.
* **conditional** expressions: **relational** expressions, **boolean/logical** expressions
    * how do you write conditional/Boolean expressions in C++?
        * what are the relational operators provided by C++?
        * what are the Boolean operators provided by C++?
        * an expression that involves relational and/or Boolean operators is a conditional/Boolean expression; (note that such expressions may involve arithmetic sub-expressions, too --- e.g., seeing if the sum of two values is less than another...)
            * what is the type of such an expression? (not **int**, but...)
            * given a description of such a desired operation, you should be able to write an appropriate C++ expression implementing that operation.
            * Need to be comfortable reading, writing, debugging conditional/Boolean expressions.
    * What are the two possible literals of C++'s **bool** type?
    * (be careful of the *difference* between = and ==) !!
    * Beware of common pitfalls (e.g., know how to properly see if a value is between two other values)
* **numeric intervals**
    * should be comfortable reading and writing numeric intervals as discussed in lecture and in the assigned reading;

  * given a marked number line, write the equivalent interval; given an interval, be able to mark a number line so that it corresponds to that interval.

  * be able to write a C++ function that determines whether an argument is within a specified interval or not;
  * need to be able to write the interval corresponding to a C++ expression.

* **function that test conditions**

  * Should be comfortable reading, writing, and calling boolean functions (functions whose return type is **bool**.)

* **branch structure / C++ if-statement**

  * what are the 4 basic structures of programming? (sequence, branch, procedure/function, and repetition)

  * what is one of the statements that C++ provides for implementing the branch structure?

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

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

  * be comfortable with the if-else if-else "style"/pattern of using particularly-nested C++ if-statements to express a situation where you wish to make exactly one choice from a collection of three or more choices; should be able to indent it following the course style standards.

  * be comfortable designing, reading, and writing if-statements; you should be able to read an if-statement, and tell what it is doing; you should be able to given its output.

  * be comfortable designing, reading, and writing **conditional functions**; given a conditional function, you should be able to read, write calls to that function; you should be able to indicate what such a function call would return as its value.

  * what (according to our class style standards) are the minimum examples/tests that should be included for a **conditional function**?

  * should be able to read a flow chart depicting a branching situation;

* **local variables**

    * how is a local variable declared? Where is a local variable declared? Once declared, in what part of a program is that local variable "visible"? (This is called its **scope** --- the part(s) of a program where a particular name has meaning.)

    * one can initialize the value for a local variable when it is declared --- how?

    * if a local variable has been declared but not yet set to anything, what should you assume about what is stored in that local variable? Be comfortable with the course style standard related to this!

    * how can you set a local variable to a value? (You should know at least **three** ways...)

    * remember that a local variable is another C++ identifier --- so, what are the course style standards about how it should be named?

* **assignment statements**

    * what is an **assignment statement**? What does it do? How do you write one? When should you use one?

    * what do we mean by the LHS and RHS of an assignment statement? what can be on the LHS of an assignment? what can be on the RHS of one?

    * be comfortable reading and writing assignment statements; I am quite likely to see if, given a fragment including assignment statements, whether you can say what the values of various variables would be at the end of that fragment.

    * be able to write an assignment statement that **increments** a particular local variable; be able to write an assignment statements that sets a particular local variable to some expression.

    * course style standard to keep in mind: given the kind of parameters that we are using currently, note that it is poor style to *change* a parameter within a function. (Now that we have **cin** statements and **assignment** statements, you now might be tempted to do this --- but remember that it is poor style, and AVOID doing so.)

      (if you want to change a variable within a function, change a local variable instead --- for example, assign a parameter value to a local variable, and then change that local variable as desired instead of the parameter itself.)

* **main() functions**

  * in C++, what function is expected to be part of every C++ program? When a C++ program is run, where does execution begin (at the beginning of what function)?

  * what is the expected header for a C++ main() function (in CIS 130)?

  * according to course style standards, what is expected to be returned by every main() function?

  * what standard library ends up being #include'd in most main() functions? Why?

  * what do we mean by a "testing" main() function? Should be comfortable reading/writing/debugging a main() whose purpose is to test a non-main() function's examples.

  * how can you write a main() that implements an interactive interface to a non-main() function (particularly to a "pure" non-main() function)? Should be able to read/write/debug such a main() function.

  * (should also be aware that there are many other styles of main() functions, too; all end up serving as the "dispatcher" for all of the functions involved in a particular C++ program...)

* **#include**

  * should know when you need to #include the standard libraries **cmath**, **iostream**;

  * given the name of a standard library, you should be comfortable writing an appropriate #include statement to make that libraries' functions available within a function you are writing.

  * should know when you need #include for a function you have written;

  * given the name of a function that you have written, you should be comfortable writing an appropriate #include statement to make that function available within a function you are writing.

* **simple use of cin, cout; literal strings**

  * you should be comfortable reading/writing string literals in C++ (any characters surrounded by double-quotes).

  * what C++ library should you **#include** to use simple interactive stream input/output?

    * should be quite comfortable with how **cin** and **cout** work;

    * what does **endl** do? how can you get spaces in your output if you want them?

    * you should be able to write well-designed cout and cin statements to prompt a user for interactive input;

* **compiling your own functions and programs**

    * how do you just compile a function (whether main or auxiliary)? Given the name of a function to compile (or the name of a file containing a main() function), you should be able to write the command that you would type into cs-server to just compile that function (or read/debug such a command if given).

        * what kind of a file results from this command? what is its name? Can this resulting file be directly executed?

    * how do you compile, link, load, and create an executable for a C++ program, given the name of the file containing a main() function, and all of the functions involved in that program (called by that main(), called by a function called by that main(), or called by any function called by some function in that program)? You should be able to write the command that you would type into cs-server to do so (or read/debug such a command if given).

        * what kind of a file results from this command? what is its name? Can this resulting file be directly executed? How can it be used?

        * what part of this is specifying the name desired for the executable result?

        * what files do you need to include as part of this statement?

* **void functions** and **function with no parameters**

    * what is the return type for a function that doesn't return anything? If it doesn't do anything, what are some examples of what such functions might do instead? What does a void function's contract look like?

    * what statement does not need to be included within a void function?

    * how do you call a void function?

    * how do you write a contract for a function that doesn't return anything? How do you write its header? How do you call such a function?

* **loops/repetition**
    * remember, repetition is one of the four basic structures of programming (sequence, branch, procedure/function, and repetition)

    * know that a **while statement** is one of several statements that C++ provides for implementing the repetition structure;

    * what is the syntax of the C++ **while** statement? what is its semantics?

    * what do we mean by a loop condition? by a loop body?

    * you need to be familiar with the two classic "patterns" we discussed in connection with loops: count-controlled, and sentinel-controlled.
        * you should be able to read, write, and correct loops using the classic versions of each of these;

        * (you should know that these are simply two very common patterns --- they are not your ONLY possibilities! 8-) You should be able to read/write more general **event-controlled** loops, too, as well as other loop variants.)

        * yes, you do have to know how to write/read/correct a classic count-controlled loop that happens to use a while loop;

    * be comfortable reading and writing while loops; you should be able to read a while loop, and tell what it is doing; you should be able to give its output; you should be able to tell how many times its body is executed.

    * what is an infinite loop? how can it be avoided?

    * what is some other classic loop concerns? (does it go one too many, or one too few, times? is it ever entered at all?)

    * should be able to read a flow chart depicting a looping situation;