CIS 130 - Exam 2 Study Suggestions

last modified: 03-29-10

- Anything covered in class or on a homework is fair game for the exam.
- You are responsible for the material covered through the end of Week 9 and through and including HW #6
- These are simply suggestions of some especially important concepts, to help you in your studying.
- 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 on one or both sides. 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.
- You will be writing and reading expressions and functions on this exam; you will be answering questions about concepts, expressions, and functions 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).
  - So, 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.
- The language of this exam will be C++ -- you will be tested on conditional/branching expressions, and conditional functions, but they will happen to be written in C++.

C++ basics

- There will be some questions that are extremely similar to those on Exam 1, but now you will be answering them using C++ instead of Scheme.
  - So -- make sure you are able to answer "C++ versions" of applicable (non-graphics-related) questions from Exam 1.
- what is a simple expression in C++? what is a compound expression in C++? Should be able to read these, write these, tell the type of a given expression, write an expression of a given type
  - be sure to use C++ data types -- `int`, `double`, `bool`, `string`, `char*`
  - literals are expected to be written using C++ syntax as well
- you should know the difference between an expression and a statement; you should know how a statement is terminated in C++.
  - note that, often/usually, C++ statements end in semicolons (unless you are talking about a block, { }). Expressions, which have a value, are usually within C++ statements, and so don't need semicolons.
- need to be able to write a C++ function using the design recipe! (need to be able to write the steps that ~st10/funct_play2 asks you for!)

  - how does a C++ contract differ from a Scheme contract? (note: even though funct_play2 fills it in for you, note that a C++ contract includes the function's name, as well as the types of the expressions it expects and the type of the result it produces)
  - need to be able to write a C++ function header, and a C++ function body. Need to be comfortable reading, designing, writing, testing, and calling/using C++ functions.
  - need to be able to write appropriate specific examples/tests for C++ functions
- need to follow the course style standards (for Scheme and C++)
- need to be comfortable with C++ identifiers, and C++ literals.
- how do you write a named constant declaration in C++?
- what types have we discussed so far in C++? how can you write literals of each? how would you declare variables of each?
  - you are expected to be comfortable with C++ string literals (anything written within double quotes); although these are really of type `char*`, note that they can be assigned to variables of type `string`
  - you should be able to declare new-style C++ `string` variables (for exam purposes, and really for this course's purposes, you are expected to use the `string` type for string parameters and named constants in C++)
- what are the basic arithmetic operators of C++? what do we mean by operator precedence? how do you write the relational operators in C++? the boolean operators? What happens when you divide two integers?

conditional/branching statements

- You should be very comfortable with reading and writing the C++ if statements, including the choose-exactly-one-of-more-than-two-choices style of if-statement
- You should be able to determine the appropriate number of and kinds of test cases/specific examples you need, based on the kinds/categories of data involved in a problem.
- You should be comfortable with interval notation for expressing ranges of numbers
- You should be comfortable with boolean values, boolean operators (and relational operators), boolean expressions, and boolean functions
  - keep in mind: what is the name of C++'s boolean type?
- How can you see if one expression has the same value as another expression? (be careful of the difference between = and ==) !!
- expect to have to read, write some if statements, and functions involving if statements
- 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?

the C++ bool type

- what are the only two `bool` literals?
- 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`.

numeric intervals

- should be comfortable reading and writing numeric intervals as discussed in lecture
- be able to write a function that determines whether a value is within a specified interval or not;
- be able to write the interval corresponding to a given C++ expression

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.)
- You should be able to determine if an identifier is a parameter, a named constant, a local variable, or a function name; you should be able to appropriately declare and use parameters, named constants, local variables, and functions.
- How can you set a local variable to a value? (You should know at least two ways, at this point.)
- Remember that a local variable is another C++ identifier --- so, what are the course style standards about how it should be named?

mutation - assignment statements, +=, ++, and more

- 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?
- what is the difference between = and ==? what does i = i + 1; do?
- 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 the value of some expression.

- What do +=, -= *=, /= mean? How are they used? What are their effects and semantics?
- What does ++ mean? How can it be used? What is its effects and semantics?
- IMPORTANT NOTE: 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 the `cin` and assignment statements, you now might be tempted to do this --- but this is considered to be poor style, and you should 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.)

interactive input/output

- you should be able to read, write fragments of code using `cin` to read in input from a user into a local variables interactively
- you should be able to read, write fragments of code using `cout` to output values of expressions to the screen
  - you should be able to do simple formatting (putting in spaces where desired, putting in newlines where desired)
  - 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;
- what library do you need to `#include` to use `cin` and `cout`?
- make sure you know the difference between a function expecting a parameter and a function performing interactive input; make sure you know the difference between a function returning/producing a value and printing a value ot the screen.

void functions and function with no parameters

- how do you write a contract for a function that doesn't expect any parameters? How do you write its header? How do you call such a 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?
  - if it doesn't return anything, what are examples of what such a function might do instead?
  - what statement does not need to be included within a void function?
  - how do you call a void function?

"complete" C++ program

- What is the definition of a C++ program? What function must be part of every C++ program?
  - There are several acceptable headers for this function; what is the one that we have been using?

  - Given course style standards, what is this function expected to return?

- Note that the examples handout will include the main template main_template.txt from the public course web page; you should be able to write a main function, given that; you should be able to read a main function; you should be able to tell, from a collection of functions making up a program, what that program would do when it is run
- you should be able to write a main function to test a void function (a function that doesn't return anything);
- How can you compile and link a collection of functions to create an executable version of a C++ program? Given all of the files involved in a C++ program, how can you link and load and create a C++ executable program for that program?