CISC105 Spring 2007 Lab03

- Review the code examples from class.
- Your textbook is a valuable resource. When something in your code is causing trouble, find an example in your book and compare thw two. Alos, your book can go into details that I don't have time to cover in class, so be sure to read the text version of each topic we cover - it will help to cement the knowledge in your brain, and may raise or answer questions.
- Some programs below are associated with a question. **Answer the questions** using C comments below your code in the program file.
- Remember that comments are an important part of your program. Every program must have your **name, TA, section,** and a **description** of what the program does, and every function must have its own comments. Programs that have complex innards need comments sprinkled throughout.
- Any time you expect user input, you must print a **prompt** telling the user what to enter.

Programs

For each numbered problem below you will write a small program (except the last one). Name each program lab03.n.c, where n is the number in the list below. For example, the name of the file for the first will be lab03.1.c

1. In the last lab we looked at type **void** functions that print values, but do not return values. Math functions typically **do not** print anything - they return a value. The function call is used as an expression that evaluates to a value of the same type that the function prototype states.

   Write a function of type double that takes a single double parameter and returns the square of that parameter. Have your main() call the function three times on different values and nicely print the returned values.

2. Copy program lab02.8.c. into a file for this lab. Add a second function that finds the larger integer, but **doesn't print anything**. Instead, this function will be of type **integer**, and so the call to the function will evaluate to an integer in main(). Think: what integer should the function return?

   Have the main() function print the value returned by your new function. When you're done, your main() should have two calls inside, one to the old function and one to the new one.

3. Copy lab03.1.c. Take the previous calls out of main().

   Add a new function that returns the sum of three doubles. Call the function on 3.4, 5.67, and 1.21. (Should you print the result in main()?)

   Now in main() call the sum function again, but this time use your squaring function too, so that you pass the squares of the numbers into the sum function. Think: how many times will your squaring function be used?

4. Copy 3. Make a third function that takes three double parameters and returns a double. The function body will consist of calls to the other functions you already have (i.e. there will be no multiplication or addition happening in the body, just calls), and will return the sum of the squares of the three parameters.

   After your function is working, add code to allow the user to input the three numbers to main().

5. Write a program with a while loop that prints the integers from 0 to 10 inclusive on one line.

6. Write a program with a while loop that prints the integers from 0 to some integer entered by the user (inclusive) on one line.

7. Write a function that prints a row of n asterisks, where n is a parameter. Call the function from main() with user input.

8. Copy lab03.7.c. Put the user input and the call to the function inside a while loop, and continue asking for length and printing asterisk rows until the user enters a negative number. (Hint: prompt and scan input *before* the loop starts; call the function at the top of the loop, and then prompt and scan again before the end of the loop).

9. Write a program using the && operator in the condition for an `if` statement. Take a real number as user input (use a double variable) and print "34.5 is in range" **if** the number is between 20 and 40, inclusive; print "34.5 is out of range" otherwise (where 34.5 is just an example number - you print the input). Test your program with the three possible cases of numeric input.

10. In a single program, use the logical operators && and || to write an expression that correctly captures each of the following the ideas, and prints the evaluation:

    (a) Three is less than five and five is greater than zero

    (b) Three is greater than one or two is less than four

    (c) Six is less than three or greater than one

    (d) Ten is greater than twenty, or five is less than both six and seven

11. The boolean[1] C expression `(5 < x < 10)` is valid in the language, but doesn't mean what you might think. Use the expression as the condition for an if statement that prints "x is between 5 and 10", take user input for x, and show that the user can enter a value for x that makes an apparent contradiction. Write on your script or explain in your comments what is really being calculated.

You should have a total of 11 programs named lab03.1.c to lab03.11.c. Make a single script file (see lab00 for the scripting instructions) where you cat, compile, and run each one in its final form (if it didn't compile, don't run it in the script - mark the place in the printed script file with a colored marker so it stands out). After all files have been run in the script, use ls and cd to show your new directories and their files in the script.

On the first page of every printed copy for this course, your name, section, and TA's name must appear.

Submit all program and script files on MyCourses before midnight Thursday of next week, and give the paper version to your TA at the beginning of your Friday lab (or in lecture Friday if you have a Wednesday lab). Note: cat, compile, and run each program in order! Do *not* cat all programs, then compile, etc.

[1] "Boolean" means it evaluates to true or false.