CISC105 Spring 2007 Lab08

- Code incrementally, save and test often! I will save many versions of a long program on the way to its completion, so that if I make a big mistake or lose a file I can retreat to a previous, working version. **Programs that do not compile lose 50 percent** even if they "look" like they should work, so compile and test, compile and test, compile and test.
- **Finish project 2 early!** It is going to take longer than the first one.
- "String" is not a data type in C. What we call a string in C is a **null-terminated char array**. This kind of string is also used extensively in C++.
- Some programs below are associated with a question. **Answer the questions** using C comments below your code in the program file.

For each numbered problem below you will write a small program. Name each program lab08.n.c, where n is the number in the list below. For example, the name of the file for the first will be lab08.1.c

Programs

1. A character is represented by putting a symbol inside single quotes, so `'a'` is of type char. Use characters in quotes the same way you would use an integer value, i.e. you can say

   ```
   char c = 's';
   ```

   Write a program with a char variable. Ask the user to type a char, read it into your variable (how?) as a single char, and print either "Not a G" or "That's a G" depending on the input.

2. Copy the previous program, and add a loop so that the program repeatedly asks for a **single char**, reads a **single char**, and reports whether it is a G or not. The loop should stop when the user enters the char X.

   Observe your program in operation. Does it behave as you expect? Explain the behavior in your comments.

3. Declare an array of type char. Use *constant initialization* (with curlies) to initialize the array to contain 8 characters as follows: `'p','a','t','\n','\0','b','a','t'`. After initializing the array to contain these characters, use a loop to print the whole array out, one element at a time, using the format specifier `%c`.

4. Once you have done 8.3 and it is working correctly, write a new program and print the entire array at once without a loop, using **only** the name of the array and the format specifier `%s`. Explain what is printed.

5. Copy the previous program. Use assignment to change the third element of the array , `'t'`, to match the fifth element without using a quoted char constant (how?). Describe in your comments what happens when you print now, and explain it.

6. Declare three integer variables, and use assignment to put char constants in each one. Also declare three char variables, and put integer constants into each one. Print all six as both type int and as type char, with appropriate message (that's twelve things printed).

7. Declare a file pointer named "input" for reading a file. Use fscanf to read every *character* in the file until the function call **feof(input)** returns true (feof looks for a special sentinel Unix stores at the end of a file). As you read each character, print it to the screen. What Unix utility function does this program emulate? Call it on the C file for this program for your script.

   Hint: Just as we ask a user for input before we test to see if it is a sentinel value, we do the same here. Have one call to fscanf before your first test.

8. Declare a char array with unspecified size and initialize it to the string "toad". **Show** that the char after 'd' is the null-terminating char by comparing it to the constant char `'\0'` and printing a nice message.

9. Declare three character arrays of size 8. Read the three words from user iput "two wise rats" and put them in the three arrays using scanf and `'%s'`. Print the words from the arrays.

10. Use the previous program again, this time reading the words "fourteen artificial arthropods". What does the program print? Is it what you expected? Explain and include a drawing of the three arrays and their contents with your script.

You should have a total of 9 programs named lab08.01.c to lab08.9.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).

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.