Lab #3 – Practice some more C

Let's practice with some C programs. We need to become more comfortable with "strings" in C, passing parameters and return values, including the use of pointers, I/O and file I/O. Please refer to the C overview handout for guidance. As always, please feel free to ask questions if there is something you don't understand.

Let's create a new folder that will contain today's C work. Hmmm, let's call the directory C. ☺

You will need the following files to complete this lab: prime.c, phone.c, struct.c, struct.in, mystery.c, and words2.txt. From Linux you can use the scp command to copy these files from another machine that you have access to.

A Linux tip: If you are running a command that will take a long time, it is probably a good idea to run it "in the background." To do this, type & at the end of your command. This way, you can still type other commands and continue to multitask while the command is running.

Prime Numbers

Let's start with a simple program. Compile and run prime.c. Look at the source code, and be sure you understand all of the C features used in it. How could this program be enhanced to make it more useful?

Phone number words

1. The purpose of the program phone.c is to ask the user for a 7 letter word, and then turn it into the corresponding phone number. You will finish this program. Everything is done except the body of the verify function. Read the comments to see what needs to be done. Compile, and then run your program a few times to make sure it works.

2. Modify this program so that instead of asking the user for input, it takes its input directly from the command line. In other words, typing "phone liberty" means that the word "liberty" should be used as the input to the phone program.

   You should also add code that warns the user if no command line input was specified.

Structures

Perhaps the most conspicuous difference between C and Java is that there is no object-oriented programming in C. C was developed in 1972, years before OO was invented. The C++ language came out in 1986 and supported OO as well as the traditional procedural programming paradigm. And Java, announced in 1993, is fully OO.

In C you cannot declare a class, so instead we use a `struct` – which stands for structure (which some people also call a “record”). A struct in C is like a class in Java or C++, but there are only data members, no constructors or member functions of any kind. Yes, you could write functions that manipulate structs, but they would act like “static” functions in Java. Also, we don’t have the encapsulation that you see in OO – everything is public: you can’t declare anything to specifically have public, protected or private access.

Examine carefully the program struct.c. At the beginning of the program we declare a struct that represents a new data structure to model information about a student. The program’s main() function delegates the work to 3 functions: get_input(), compute_averages(), and output(). I have written the first and last of these, and you are to write the compute_averages() function.

Compile and run the program. I have provided an example input file called struct.in. Use the redirection operator < so you can run the “struct” program on this input. You will notice that the students’ averages need to be calculated. Feel free to experiment further by adding more names and scores to the input file.

Once you have implemented the compute_averages() function, modify the format string in the output() function so that the names, scores and the average are in neat columns. Also, you should change the way the average is printed so that it only prints to 1 decimal place instead of 7 by default. √

Before moving on to our last exercise, I want to show you something. Take a look at mystery.c. Compile and run this program. Isn’t this incredible?! Basically, C lets you do anything, even make your code absolutely unreadable.

Reverse Dictionary

Now, let’s try something a little more challenging. ☺ The file words2.txt is a pretty complete listing of words in the English language. Write a C program to convert this into a REVERSE dictionary. In other words, alphabetize based on the right end of the word rather than the left. Write your revised list of words to a new file.

As always, try to accomplish this program in stages. First, make sure you can open a file, and read all of its contents. Determine a good way to reverse a string. Finally, determine how to sort your array.

```
main() {
   if (you_love(C))
      honk();
}
```