SYSC 2006 Foundations of Imperative Programming

Lab 3

Learning Strategies

The demand for TAs is always highest in the last portion of the lab; if you need specific help, take the initiative and ask for help at the start of the lab.

***Always track your time*. If you cannot finish this lab in the allotted time, you need to study more before the next lab or assignment.**

Objective

- Refresh of prerequisite material: functions (pass-by-value and pass-by-reference) and arrays.
- Pass-by-reference is different in C and in C++.
- Demonstrating Test Driven Development

Agenda

1. **Lab Introduction by the TA**: Unless your past performance is A's, you are expected to gather round the TA for a brief introduction of the lab. The TA will identify the key points to learn, probable error spots, and suggested work strategies. TAs are not necessarily experienced lecturers so please help the process by gathering quickly and being quiet. Be at your best behaviour for these 15 minutes.
2. **Activity A**
3. **Activity B**
4. **[Optional] Demo**: Ask a TA to give you feedback on your lab. There is no mark for this demo. The purpose is for your learning benefit. Anyone with a grade below B should always ask for feedback.
5. **Submission:** Submit your **source** files on the course webpage.

Activity A (1 hour)

You will again "program to a function prototype". In this lab, however, you are provided with the test driver (the calling code) in lab3a.c. You only have to provide the (correct) implementation of all the functions. The functions are simple array exercises and are mostly self-explanatory except for that last two which require you to explore C's math library. They are described below.

- TIP: The test driver is written in a format that you will have to re-produce in later labs. Pay attention to its structure as you work.
- TIP: This exercise is listed as taking only 1 hour. There is actually a lot of code to write so you must be efficient. Hopefully you will see that the functions all follow the same pattern, so copy-paste is a very practical approach.

avg_magnitude()

A sound is recorded by using a microphone to convert the acoustical signal into an electrical signal. The electrical signal can be converted into a list of numbers that represent the *amplitudes* of samples of the electrical signal measured at equal time intervals. If we have n samples, we refer to the samples as $x_0, x_1, x_2, \cdots, x_{n-1}$.

The *average magnitude*, or average **absolute** value, of a signal is given by the formula:

average magnitude = $(|x_0| + |x_1| + |x_2| + \cdots + |x_{n-1}|) / n = \Sigma |x_k| / n$; k = 0, 1, 2, $\cdots$, $n - 1$

Explore C`s math library (math.h) for the function that calculates the absolute value of real numbers.

avg_power()

The *average power* of a signal is the average squared value, which is given by the formula:

average power = $(x_0^2 + x_1^2 + x_2^2 + \cdots + x_{n-1}^2) / n = \Sigma x_k^2 / n$; k = 0, 1, 2, $\cdots$, $n - 1$

Mandatory Instruction: Unless it is the express purpose, ***a function must not print anything***. As a rule-of-thumb, printf() inside of a function is prohibited.

Activity B (1 hour)

You are provided with another test driver (lab3b.c), this time calling functions that use strings. You are to repeat the same procedure but now you will be testing your knowledge of strings. There is an extensive string library but in this lab you are not permitted to use any of them. The reason is to provide you with more experience using loops and character arrays.

The final function max_occurrence() is a tricky function. First, it uses reference arguments. Second, it requires nested loops. Ask the TA for a tutorial if needed.

Self-Test: Did you learn from this lab?

1. How many syntaxes do you know for initializing a string? Prove your versatility by initializing both an empty string and a single-character string using the 'character' notation.
2. Compare and contrast the length and capacity of a string.
3. How do the function prototypes functions with array parameters typically differ from the prototypes for functions with string parameters? Explain.
4. How do the function implementations for functions with array parameters typically differ from the implementation for functions with string parameters? Explain.
5.