

## Programming

### Exercise

#### 8.1

### Parameter Lists

Purpose. The purpose of this lab is to make you feel more comfortable with parameter lists by having

you make some simple changes to an existing subprogram's parameter list.

Requirements. Rewrite the paramList.cpp program from chapter 8, and name it myParamList.cpp. Modify the program so that it averages 3 values, instead of 2. Also, modify it so

that it prompts the user for the three values to be averaged, instead of setting their values in main.

The prompts should appear in main, and not in the function.

Further modify the program so that the output result is labeled, such as "The average is...", and formatted to have 2 decimal digits.

Program I/O. Input: 3 whole numbers from the console keyboard. Output: the average of the 3 inputs, with a label, and rounded to the nearest hundredth..

Examples (3). Here's what the output should look like, with user input in blue:

Enter the first whole number: 2

Enter the second whole number: 3

Enter the third whole number: 5

The average is 3.33

## Programming

### Exercise

#### 8.2

### More Parameter Lists

Purpose. The purpose of this lab is to make you feel more comfortable with using parameter lists

when calling a function from main, by having you make changes to an existing program's function

calls. It also gives you more practice with the "random number generator".

Requirements. Rewrite the addition.cpp program from chapter 8, and name it myAddition.cpp. Modify the program by using the C++ random number generator to generate the

two input numbers for each question. Limit the randomly-selected input numbers to between 0 and

10, inclusive.

To avoid having identical code appearing in main 5 times, because specific numbers no longer appear in the parameter lists of the calls, modify main to use a count-controlled loop that calls the

function.

NOTE: You need this as the first statement in "main": srand(time(0)); Never put

srand(time(0)); in a loop or function (other than main) -- it should be executed only once.  
Program I/O. Input: 5 whole numbers, each in response to a math problem. Output: 5 correct/incorrect responses to a user input.

Examples. Here's what the output should look like, with user input in blue:

6 + 4 = 10

Correct!

6 + 6 = 12

Correct!

3 + 7 = 11

Very good, but a better answer is 10

5 + 9 = 14

Correct!

10 + 3 = 13

Correct!