Laboratory 4 Activity Instructions

1. Download the "**`Lab04Activity_v01.cpp`** and "**`formatted_cmdline_io_v08.h`**" from the "Laboratory Activity" segment in the "Week 4" folder. Remember to save the formatted command line file as a header file in Dev C++. Also note that this is a new formatted command line header; do not use your older one.

2. Build and run the program. As long as you saved the two files correctly, this should work. The program multiplies two numbers and provides a product.

3. Now you will be making this program more modular by using functions.

4. First, go to the header of the **`printTitle`** stub function in the "Supporting Function Implementation" area of the program.

5. Under the following lines:

```
// display first line of title
   // function: printString, printEndLines
```

   Type the following code, then compile:

```
printString( "Multiplication Program", NO_BLOCK_SIZE, "LEFT" );
printEndLines( ONE_LINE );
```

6. Under the following lines:

```
// display second line of title, with extra vertical space
   // function: printString, printEndLines
```

   Type the following code, then compile:

```
printString( "======================", NO_BLOCK_SIZE, "LEFT" );
printEndLines( TWO_LINES );
```

7. Now go back to the main function, and replace the following lines:

```
// function: iostream <<
cout << "Multiplication Program" << endl;
cout << "======================" << endl << endl;
```

   with the following code, then compile:

```
// function: printTitle
printTitle();
```

8. As long as there are no warnings or errors, you should see that the program does exactly the same thing it did before. But now you are calling a function to do the work, and making the main function code more readable.

9. Next, go to the header of the **`getInput`** stub function in the "Supporting Function Implementation" area of the program.

10. Under the following lines:

```
// print "Enter " pretext
   // function: printString
```

    Type the following code, and compile:

```
printString( "Enter ", NO_BLOCK_SIZE, "LEFT" );
```

11. Under the following lines:

```
// print input name ("first" or "second")
   // function: printString
```

    Type the following code, and compile:

```
printString( inputName, NO_BLOCK_SIZE, "LEFT" );
```

12. Under the following lines:

```
// prompt user for input with last part of string ("number: ")
   // function: promptForInt
```

    Type the following code, and compile:

```
response = promptForInt( " number: " );
```

13. Under the following line:

```
// return input value
```

    Replace the following code:

```
return 0; // temporary stub return
```

    with this code, and compile:

```
return response;
```

14. Now go back to the main function, and replace the following lines:

```
cout << "Enter first number: ";
cin >> firstNum;
```

    with the following code, and compile:

```
// function: getInput
firstNum = getInput( "first" );
```

15. Now while still in the main function, replace the following lines:

```
cout << "Enter second number: ";
cin >> secondNum;
```

    with the following code, and compile:

```
// function: getInput
firstNum = getInput( "second" );
```

16. Now run the program. Since you have been compiling all along, there should be no problem with building and running it.

17. Next, go to the header of the **`calcProduct`** stub function in the "Supporting Function Implementation" area of the program.

18. Under the following line:

```
// initialize function/variables
```

    Type the following code, and compile:

```
int product;
```

19. Under the following lines:

```
// calculate product
   // operation: math
```

    Type the following code, and compile:

```
product = valOne * valTwo;
```

20. Under the following line:

```
// return product of math
```

    Replace the following code:

```
return 0; // temporary stub return
```

    with the following code, and compile:

```
return product;
```

21. Now go back to the main function, and replace the following lines:

```
// operation: math
product = firstNum * secondNum;
```

    with the following code, and compile:

```
// function: calcProduct
product = calcProduct( firstNum, secondNum );
```

22. Again, go ahead and run the program.

23. Finally, see if you can finish out the **`displayResult`** function implementation as you have the other functions, and then replace the display code in the main function. If you do not have time to complete this, the code will be uncovered after 5:00 pm this afternoon, so you can work on your own (do this first) and then check the results.

24. As you can see from your work today, the program remains the same whether you break it up into modules or not. However, if you look at the main function when you are finished, it is eminently more readable. It also takes advantage of abstraction of operations and the ability to diagnose problems in a much easier way. Take some time to review this, and refer back to this code as needed when you are developing your own programs.