Lab 5 - Functions

1. Output vs Return

1. Create a new project for this lab, and create a new program called `functionFun.cpp`.

2. Add the usual `main()` function.

3. Create a function with the following header:
   ```
   void displayWelcome() {
   }
   ```
   ◦ Make this function output a welcome message to the screen.
   ◦ Have `main()` call this function and test your program.
   ◦ Have `main()` call this function a second time and test your program.
   ◦ Run the program. Output should look like (OK to have a different message!):
     Hello and welcome to the lab on functions!
     Hello and welcome to the lab on functions!

4. Create a function with the following header:
   ```
   void displaySumOfSquaresUptoN(int n) {
   }
   ```
   ◦ It should add up the square of each integer from 1 to `n`. (i.e. display $\sum_{i=1}^{n} i^2$)
   For example, if `n=4`, it adds $1^2+2^2+3^2+4^2 = 30$, and outputs the sum to the screen.

5. In `main()`, prompt the user for their favourite number (an integer), and pass it as an argument to the `displaySumOfSquaresUptoN()` function.
   ◦ Run the program. Your output should look like:
     Hello and welcome to the lab on functions!
     Hello and welcome to the lab on functions!
     Enter your favourite number: 5
     Sum of squares 1 .. 5 = 55

6. Add a new function with the following header:
   ```
   int getSumOfSquaresUptoN(int n)
   ```
   ◦ Make this function do the same thing as `displaySumOfSquaresUptoN()`, except instead of printing to the screen, have it return the sum.
     Hint: Copy and paste part of the `displaySumOfSquaresUptoN()` function.
   ◦ Have `main()` call this function, passing in the user's favourite number and have main print its return value to the screen using the following code:
     ```
     int sum = getSumOfSquaresUptoN(favNum);
     cout << "Returned sum of squares is: " << sum << endl;
     ```
Your output should now be:

Hello, and welcome to the lab on functions!
Hello, and welcome to the lab on functions!
Enter your favourite number: 5
Sum of squares 1 .. 5 = 55
Returned sum of squares is: 55

7. In `main()`, write code to outputs a number of stars ('*') equal to the sum of squares from 1 to n.
   - For example, if n is 3, the sum of squares from 1 to 3 is 14, so output 14 *'s:
     ********************
   - Use one of the functions you wrote in this section to help accomplish this.
     Hint: Does displaying the sum from 1 to n on the screen help with this? Does getting (returning) the sum from 1 to n help with this?
   - Output:

     Hello, and welcome to the lab on functions!
     Hello, and welcome to the lab on functions!
Enter your favourite number: 5
Sum of squares 1 .. 5 = 55
Returned sum of squares is: 55

8. Move the code that generates the row of '*'s into its own function. Pass in the number of stars to draw as an argument.
   - What should this function be named? Pick a name which best describes it.
   - What should the return type be for this function?
   - What arguments are needed for this function? What type and name should they have?

9. Try simplifying your `displaySumOfSquaresUpToN()` function. Think about the rest of the code you have written for this lab and figure out how to reduce the duplication of code.
   - Hint: You can call one function from inside another function.
   - Hint: You may need to reorder your functions so that the function being called appears the function calling it.
10. Understanding:

◦ What is the *fundamental* difference between a function returning a value, and a function displaying a value?

◦ For outputting the stars to the screen, will either one of the functions `displaySumOfSquaresUpToN()` or `getSumOfSquaresUpToN()` help? Why or why not?

◦ What is the difference between passing in an argument and using `cin` in the function?

◦ Can one function call another function?

2. Making Functions

Copy the `makeFunctions.cpp` file into your project from the course web site. Run the program to see what it does.

Use functions to break-down program into smaller pieces. Do not change what the program does (the output should be the same); just changing the program's structure.

♦ Which parts should be their own function?

_Hint: create one function for each calculation:_

*Each of the calculations for n², sum 1..n, and 2^n_  
*Think about the arguments these functions will need, and their return types._

*For example, create a function: `calcNSquared()` which accepts the integer n and returns the value of n squared._

♦ Note that for this exercises you will be making very short functions. Sometimes functions can be long (20-30 lines); sometimes they can be very short (1 line). Generally, shorter (1-10 lines) is best.

Optional (possibly not yet covered in lecture):
Move all of your new functions below your `main()` function. You will need to add prototypes for each of your functions.

Understanding:

♦ How to create a function which accepts parameters, and returns a value.

♦ The difference between having a function return a value, and output a value.

3. Skills and Understanding

You should now be able to answer all the "understanding" questions in the previous sections. If you want to leave early, then show the TA the following:

♦ Your operational programs which complete all of the above tasks.

♦ The TA may ask you to explain any section of the lab, or answer any of the "Understanding" questions.

♦ **Nothing** is to be submitted electronically or in hard-copy for this lab.