LAB #5 – Functions and Errors

Finish Lab 4:
Due to a large portion of the class not finishing lab 4, take time in this lab to get caught up and finish the lab. In this lab, you are required to do the extra part of the summation task. You need to define a function, \texttt{int f(int x)}, which receives an integer value as an argument and returns an integer value after the function, \(x^2\), is evaluated.

1. For this part of the lab, you will be taking working code, specifically your summation code from Lab 4, and introducing errors. You will then attempt to compile the code, and make note of the errors it generates. This should help narrow down problems in the future. If your code does not have the necessary lines in order to introduce an error, please add them.

   \textbf{Introduce the following errors, removing each before introducing the next:}

   \begin{itemize}
   \item Remove a semicolon from a line. Make note of which line.
   \item Comment out the declaration of a variable that you use in multiple places.
   \item Remove one brace from around your loop
   \item Remove the parentheses on a function call.
   \item Pass an incorrect type to a function call.
   \item Remove the \& from the scanf.
   \end{itemize}

2. Fix this code. There are >10 errors.

```c
/**
 * Original Author: Bob Smith
 * File: myar.c
 * Created: 2012 January 26, 4:00 by smithb
 * Last Modified: 2011 January 26, 4:30 by smithb
 *
 * Description: This file contains a program that
 * tells the user whether the number they entered
 * was a prime number
 * Errors including style violations: at least 10
 **/

#include <stdio.h>
#define PROMPT "Please enter a whole number:"
#define NOT_PRIME "The number %s is not a prime number.\n"
#define PRIME "The number %d is a prime number.\n"
#define DONE 0 /* ends successful program */
#define FIRST_FACTOR 2 /* initial value in for loop */
```
3. In this exercise, you will be writing and using three functions. The first counts from 0 to the parameter and prints the numbers out in a single line. The second function counts down to 0 from the parameter. The last function should count between the two parameters. If the first number is smaller than the second number, the function should count up. If the first number is larger, the function should count down.

Once all three functions are written, write a main function that prompts the user for two positive numbers, then uses the first for the count up and count down functions and both numbers for the count between function. After doing this once, the program should ask the user if they would like to quit or re-run the program.

Add error checking to your code above so that if the user doesn’t enter a positive integer they get another prompt and chance. They should get as many prompts and chances as it takes.

**Example output:**
Please enter 2 ints.
3 9
0 1 2 3
3 2 1 0
Do you want to run this again with 2 different ints? (Enter 1 for yes, 2 for no) 1

Please enter 2 ints.
10 6
0 1 2 3 4 5 6 7 8 9 10
10 9 8 7 6 5 4 3 2 1 0
10 9 8 7 6

Do you want to run this again with 2 different ints? (Enter 1 for yes, 2 for no) 2

Extended Learning:

Write a program that walks an ‘*’ to the right and back to the left exactly like this demonstration. The first thing your program should do is ask the user how far to walk. You will write functions and not put the movement code in main.

Your main function must look like this:

```c
int main(){
    int x;
    printf("Enter how far to walk. ");
    scanf("%d", &x);

    /* add some checking to see if x is <=1 or >10 */

    /* * forms a triangle */
    go_there_and_back(x);

    /* * stays on the same line */
    go_straight_there_and_back(x);
    return 0;
}
```

Hints:

- Pause 1 second:
  ```c
  sleep(1);
  ```
- Move backward 1 space:
  ```c
  printf("%c", \b);
  ```
- Flush the output buffer and print my printf statement now:
  ```c
  fflush(stdout);
  ```