COEN 44 Applied Programming **Fall 2016**

Lab 5

It is very common that one will want to make a number of comparisons between a set of data elements, and then make similar comparisons across many different data sets. To do this one implements a number of utility functions to perform the comparisons. Using a group of functions to do the comparisons is better than simply writing a main program to do all the comparisons for two reasons: some of the functions may be useful in other contexts and it's easier to reuse them if they are self-contained, and it's easier to add other comparisons at a later point because the changes made to the original program will be confined to the new functions written. This lab will give you practice in writing utility programs.

Objectives:
1. Become familiar with the "if" and "if - else" statements
2. Utilize functions with parameters
3. Gain additional experience in formatting tabular output
4. Gain experience with local variables
5. Learn about input and output redirection

Part 1: Design and implement a program to determine certain properties of three numbers

This program will input three real (floating point) numbers from the keyboard and then calculate a number of properties about the numbers. The three values should be read in using a single scanf() statement. Then you will call each of five functions you will write. To each of these functions you will pass the three values you input, the function will perform the requested calculation, and then return the required result. The functions to be written (and their prototypes) are:

1. Determine which of the three is the largest and return that value.

```
float maxval (float v1 , float v2 , float v3);
```

2. Determine which of the three values is the smallest and return that value.

```
float minval (float v1 , float v2 , float v3);
```

3. Determine if all three numbers are equal to each other. Return the character '=' if they are, or the character '#' if two or more have different values.

```
char equalval (float v1 , float v2 , float v3);
```

4. Calculate and return the average of the three values

```
float averageval (float v1 , float v2 , float v3);
```

5. Check if the first two numbers entered are “nearly” equal, i.e. if value 1 and value 2 differ by less than some tolerance called EPSILON (define this globally with a #define statement). If they are nearly equal, return the character ‘Y’ (for yes) and if not, return the character ‘N’ (for NO).

```
char nearequal (float v1 , float v2);
```

Note: for functions maxval() and minval(), if all three values enter happened to be equal, then you can return any of the values as the max value or min value, since they are all equal. For the nearequal() function, use a value of EPSILON of 0.1. EPSILON should be a double, not a float.

Format your output as a table such as shown in the example below. The \t escape sequence in the format for your printf statement will insert a “tab” character in your output. Use that to get your output to line up correctly.

Your program might look something like this when run:

```
This program will calculate several properties with respect to
three floating point values entered.
Enter three real numbers> 6.05 6.1 7.5
-----------------------------------------------------------------
|Val 1 Val 2 Val 3 Max  Min  Avg  All-Equal Near-Equal Epsilon |
-----------------------------------------------------------------
|6.05  6.10  7.50  7.50 6.05 6.55 No        Yes        .1      |
-----------------------------------------------------------------
```

Part 2: Program execution
Design several sets of test data to allow you to verify that all five functions operate correctly. Try different values of EPSILON, from small values to large values. At least once, use a very small value of EPSILON such as 0.000000001.
Lab Steps:

1. Turn in your written program outline to the TA
1. Open Visual Studio and create a new project called Lab5
2. Develop your outline for the program
3. Create your program
4. Debug and test the program
5. When working properly, demonstrate for the TA
6. Turn in a program listing to Camino

Part 3: modify and re-execute your program
Now modify your program to ask the person at the keyboard if they want to enter another set of data values, and continue to calculate results until that person says they have no more values to enter. You may assume they have at least ONE set of values to

enter.
Lab Steps:

1. Modify your program from Part 2 to satisfy the new requirements of Part 3.
2. Test and debug your program
3. When working properly, demonstrate it for the TA
4. Turn in a program listing for Part 3 to Camino