COEN 44 Applied Programming **Fall 2016**

Lab 9

It is fairly common in programs dealing with the physical world to have to **approximate** values to some precision (think of needing to calculate 1/3 to 15 decimal digits – even with 15 digits it would still be an approximation of the real value). This program gives you a simple experience with doing so. It will also give you a chance to use a very simple structure in a program.

Objectives:

1. Develop an outline for the structure of a program from scratch.
2. Convert the outline to a C program.
3. Demonstrate good programming style: commenting, indenting blocks appropriately, good identifier names, etc.
4. Reinforce the concept of an approximation that has a needed precision.
5. Gain additional experience with loops, programmer-defined functions, and parameters. Use a structure in a program for the first time.

Part 1: Design and implement the program

A number of mathematical entities can be approximated by infinite series. Two examples are

$1/(1 - x) = 1 + x + x^2 + x^3 + x^4 + \ldots$ for $-1 < x < 1$

$\ln((1 + x)/(1 - x)) = 2x + 2x^3/3 + 2x^5/5 + 2x^7/7 + \ldots$ for $-1 < x < 1$

Your program will calculate the result of either one of these series for a value of x entered from the keyboard, to a precision also entered from the keyboard. The result will be sufficiently precise when the absolute value of the previous approximation minus the current approximation is less than the precision entered.

For example, if x was 0.5 and precision was 0.01 your program would need to calculate eight terms of the series for $1/(1 - x)$ to meet the precision requirement, and the final value would be 1.9921875.

Your main function should prompt for the user to enter which value to approximate, then read the values of x and precision. Remember to verify that the value entered for x meets the range restriction ($-1 < x < 1$). You should have two programmer-defined functions, each calculating one of the approximations. Each programmer-defined function should have two parameters (x and precision) **and should return both the final value and the number of terms calculated in a simple structure**. Define the structure using a **global** typedef. Then the main function should print out the final value and number of terms, and prompt for another approximation.

Lab Steps:

0. Turn in your outline to the TA
1. Open up VS and Create a new **Project**
2. This will be a **console program** in C
3. Type in your program using good style practices
4. Test and debug your program using your test data. You should test each approximation for a wide range of precisions and for both positive and negative values of x.
5. Demonstrate the execution for the TA
6. Submit your final program, with the outline of your solution as a comment at the beginning of the source code, to Camino.

Hints

This is not a hard lab. Just write each of the three components in a methodical fashion.

To declare the global typedef, your program structure should look something like

```
#include
#define
prototypes for programmer-defined functions
typedef struct {

} return_t;
```

You can then declare variables of type return_t in the main function and the programmer-defined functions.

You may find the library function pow (x, y) [which calculates x^y] useful. It's in math.h. Both x and y must be doubles, and pow returns a double.

There is no library function to calculate the absolute value of a double, but a simple if statement will do the same thing.