# CS&E 1222

## Lab 6 – For Loop  Lab Assignment – 20 points

---

- ✓ The *lab* must be accomplished solely by you:
  - ➢ DO NOT look at anyone's code other than your own, including code from another's student in your section or another section of the course, or any third party source, e.g. the Internet
  - ➢ DO NOT share or copy anyone else's code for any graded assignment
  - ➢ DO NOT work in pairs or groups
- ✓ All cases of academic misconduct will be reported to the *Committee On Academic Misconduct* (COAM).

## Setting up the Programming Environment

Effective commenting and tabbing will affect your grade. The "style" of your program should follow the style of the sample programs in the lecture notes. Also see the example code from Lab #1. Your program should have the file name, your name, creation and last modification dates and a brief description of the program in the comments. ***In addition, read the document on "Commenting" found in the Content tab on Carmen under "Tutorials"***.

1. At the Linux command line type `mkdir lab6`. This will create a new directory named **`lab6`**. Work out of this directory. In order to do that, type `cd lab6`. This changes the current working directory to the directory **`lab6`**.
2. If you have created the directory **`lab6`**, then just type `cd lab6`.
3. Copy the files **`alt_harmonic_solution.exe`** and **`pyramid_solution.exe`** in the directory **`/class/cse1222/9643/lab6`** by typing

```
cp /class/cse1222/9643/lab6/alt_harmonic_solution.exe .
cp /class/cse1222/9643/lab6/pyramid_solution.exe .
```

Be sure to include ***9643*** (this is your course section indicator) and the period, ".".

## Programming Assignment

You will write two programs using *for* loops to perform the required computation in this lab. Only use *while* loops where you are asked to do so to receive full credit for the lab.

Write a program **`alt_harmonic.cpp`** which reads a whole number $n \geq 1$and computes the alternating harmonic series:

$$\sum_{k=1}^{n} \frac{(-1)^{k+1}}{k}$$

This series converges to the value of *ln*(2) (natural logarithm of 2) and so may be used to approximate this value. Enter *ln*(2) in your calculator and see what answer you get. Using a larger value of *n* to compute the series results in a better approximation to *ln*(2). I.e., your program's answer will match your calculator's answer better with larger values of *n*.

Write a program **`pyramid.cpp`** that reads a positive odd whole number *n* and prints a pyramid where the last row contains all numbers from *n* down to 1 (i.e., descending order), the second to last row displays all numbers from *n* – 1 down to 2, the third to last row displays all numbers from *n* – 2 down to 3, etc. The first row will contain only a single value, i.e. the middle value in the range, i.e. *n* down to 1. Each successive row contains two more values than the previous row.

For example, if *n* is 9, then the program will output:

```
    5
   654
  76543
 8765432
987654321
```

If *n* is 13, then the program will output:

```
      7
     876
    98765
   0987654
  109876543
 21098765432
3210987654321
```

The *i*'th row contains 2*i* – 1 values. Each column displays the same number. If a row has more than 10 digits, the digit after 0 should start again from 9.

Run **`alt_harmonic_solution.exe`** and **`pyramid_solution.exe`** to see examples of these programs. Your programs must behave exactly like these programs with the same input and output.

1. Program **`alt_harmonic.cpp`**:

    a. Prompt and read the value of *n* from the user. This value must be ≥ 1. Use a ***while loop*** to repeatedly prompt and re-read this value from the user if a value entered is invalid.

    b. Use ***for loop*(*s*)** to compute the alternating series (see above).

c. Print the resulting sum.

d. Debug your code by trying your program with larger values of *n*. If your sums approach the value of *ln* (2) on your calculator then your program is probably correct.

2. Program **`pyramid.cpp`**:

   a. Prompt and read the value of *n* from the user. This value must be positive and an odd number. Note that zero is not positive. Use a ***while loop*** to repeatedly prompt and re-read this value from the user if a value entered is invalid.

   b. Use ***for loop*(*s*)** to print the pyramid.

   c. Debug your program by running it on different values of *n*, e.g. 1, 3, 11, 51, etc.

3. Be sure to add the header comments "File", "Created by", "Creation Date" and "Synopsis" at the top of the file. Each synopsis should contain a brief description of what the program does.

4. Be sure that there is a comment documenting each variable.

5. Be sure that your *if* statements, *while* loops, and blocks are properly indented.

6. Check your output against the output from the solution executables provided.

## Submit Your Work

**Important: Any program which does not compile and run will receive no credit!**
If you are not sure what this means please ask your instructor.

Submit the files **`alt_harmonic.cpp`** and **`pyramid.cpp`** using the *Lab6* drop box on Carmen. **DO NOT** submit the file **`a.out`**. **DO NOT** submit work from other assignments. This will not be graded.