**CS120L – C Programming Language Labs**
Fall 2014
**Week 5: Basic Coding for Simple Problem Solving**

**Experimenting with `diff` and `WinMerge`**
Below are several files for you to use. Each pair has a "correct" output and a student's output. Look at each pair of files to see if you can spot the differences.

Correct output #1 Student output #1
Correct output #2 Student output #2
Correct output #3 Student output #3

Then, compare them using the `diff` program, which is part of the Cygwin utilities. For example, to compare the first set of files, you would type this at the command prompt:

```
diff output.master1.txt output1.txt
```

This will display the differences to the screen. If you want to capture the differences in a file (for later viewing), you need to redirect the output into a file like this:

```
diff output.master1.txt output1.txt > differences.txt
```

This will place the output in a file called `differences.txt`. You can choose to name the file anything you like. This is exactly how we can tell if your program output is correct.

At this point, it's not important that you understand exactly how `diff` works. You should, however, be able to see where the lines are different. Using Notepad++ Editor (or any text editor), modify the student output file and see how `diff` reacts.

After you've seen how `diff` works, compare the files using `WinMerge`. `WinMerge` is a graphical program that displays the files side-by-side and highlights the lines that are different. Run `WinMerge` (from All Programs in the Start menu, or choose Run... and type `winmerge`) then drag and drop the two files onto the program. Or, you can select two files in Explorer, right-click on one of them and choose `WinMerge`.

It will be obvious where the differences are. Use the "up" and "down" arrows at the top of the program to navigate between differences in the files.

**Exercises**
1) Write a program that prompts the user for 3 values and displays the output of several calculations. An example looks like this:

```
Enter 3 numbers between 0 and 9.999:
Number      sin      cos      tan      atan
-----------------------------------------
1.00000    0.841    0.540    1.557    0.785
2.00000    0.909   -0.416   -2.185    1.107
3.00000    0.141   -0.990   -0.143    1.249
```

You'll need to pay close attention to the alignment and formatting. Also, you'll need to include `math.h` for the functions. Using the math functions, you simply need provide the value you want to calculate. For example:

```
float input1 = 7.14f;
printf("%f, %f\n", sin(input1), tan(input1));
```

will print out:

```
0.755760, 1.154100
```

You are to use these inputs to test your program:

**Input #1:**

```
1
2
3
```

**Input #2:**

```
3.141592653589793238 46
0.318309886183790671538
0.785398163397448309616
```

**Input #3:**

```
0
4.5
9.0
```

You'll compile your program with this command:

```
gcc -Wall -Wextra -Werror -ansi -pedantic q1.c -o q1.exe
```

If you want to compare your output to the correct output, you'll need to redirect your output to a file (e.g. `output1.txt`) and then, using `diff`, compare your output with the correct output. You can get the correct output (via redirection) for each run by referring to these files:

Correct output #1
Correct output #2
Correct output #3

The purpose of this assignment is for you to learn how to generate the **exact** output that is specified.

**Hint:** use 32-bit floating point precision for your variables.
You should save the program in a file: "`q1.c`".

2) Your task is to find the $n^{th}$ digit of an integer number, counting from the right. Your number can be 0 or negative. If your number has fewer than n digits, then your program should print out 0.

   Sample input 1: files\q2_1.in
   Sample output 1: files\q2_1.out

   Sample input 2: files\q2_2.in
   Sample output 2: files\q2_2.out

   Important: You should **NOT** use `char`, `%c` or `%s` in your program.

   You should save the program in a file: "`q2.c`".

   You'll compile your program with this command:

```
gcc -Wall -Wextra -Werror -ansi -pedantic q2.c -o q2.exe
```

**Grading scheme:**

- 100 marks in total (50 marks each question).
- 10 marks for successful build.
- 10 marks for successful execution and correct output (2 marks * 5 hidden test cases).
- 20 marks for correct implementation.
- 10 marks for good programming style. For the complete guide, please see the Assignment Guideline on CS120 page.

  In general, you should have clear variable names, reasonable comments to explain your code, and consistent indentation. Please use **2-blank spaces** instead of tab spaces.

**Submission:**
Please name your source code for question 1 as **q1.c**, question 2 as **q2.c**, put them in a folder named **cs120b__<your Digipen login id>_<labnumber>**, in which <**labnumber**> is 5 this week, and zip them in **cs120b__<your Digipen login id>_<labnumber>.zip** for submission.

Wrong submission file/folder name will cause 10 marks deducted.
Note that from this lab onwards, the file and folder names must be lowercase.

The deadline of submission is 3rd October 23:59 and late submission will receive zero mark.