CS 271 Lab 1 - C Basics

Goals:

- **Learn to use Kate editor to create and debug C programs.**
- **Learn to use gcc compiler with math library and one source file.**
- **Learn Linux commands: cd, pwd, cp, dir, mkdir, mv, and rm**
- **Write a C program that uses assignment, input, output, selection, and repetition.**

Grading (40 points total):

1. 5 points - program documentation including header comments (see the format below) and inline ("in between the lines of code") comments.

2. 5 points - program follows the course style guidelines for indentation and spacing

3. 20 points - the program solves the given problem, inputs the data correctly, uses the correct formula(s) and correct control structures

4. 10 points - the program produces the correct output for the test data during grading. The output appears in the format shown in the problem statement.

Step by Step Lab Instructions

1. Log in. Open the Kate editor.

   To make the Terminal window visible in Kate, do the following, go to the Settings menu.

   ➔ Configure Kate
   ➔ Plugins
   ➔ check the box next to Terminal Tool View
   ➔ then click OK.

   (If you haven't used Linux before, refer to the "Essential Linux Commands" handout.)

2. **Move the cursor to the Terminal Window:** Click in the terminal window area at the bottom of Kate.

   Type **pwd** and press Enter.

   Note the directory information. Type **cd** and press Enter.

   Type **pwd** again and note the difference in the directory information.

(continuing with Linux commands in the Terminal Window...)

- Type **mkdir cs271** and press Enter.
- Change directory to cs271 by typing **cd cs271.**
- Type **mkdir Lab1** and press Enter.
- Type **cd Lab1** to change directory to Lab1. This is where you will save your files

3. **Editor:** To practice using Kate, type in the program shown below.

   Save the file as program1.c in the cs271/Lab1 directory.

   Reminder: In the terminal window, you must set your working directory to the folder where you saved your program.

   Each time you log in, you must set your working directory again.

```
#include <stdio.h>
#include <math.h>

// The purpose of this program is to calculate the length of hypotenuse
// of a right triangle.

int main (void) {

   double leg1, leg2, hypotenuse;

   // input the lengths of leg1 and leg2

   printf("Enter the length of side A\n");
   scanf("%lf", &leg1);                          It's percent "el", not one.
   printf("\nEnter the length of side B: \n");
   scanf("%lf", &leg2);

   // calculate the hypotenuse

   hypotenuse = sqrt( leg1 * leg1 + leg2 * leg2 );

   // display the length of the hypotenuse with 2 decimal places

   printf("The hypotenuse is %.2f\n", hypotenuse);

} // end main function
```

4. **Terminal:** In the Terminal window, compile the program and produce an executable file named program1. Here's the command:

```
gcc program1.c -lm -o program1
```

> "flags" or options for the compiler are preceded by -.
>
> -o must appear immediately before the name of the executable
>
> -lm (an "el", not a one) means that the math library is needed

If you have syntax errors, go back to the editor window and fix them. Compile again and debug until the program compiles correctly.

5. **Terminal:** Run the program by typing a period, a slash, then the name of the executable:

```
./program1
```

Test the program by entering 3.0 and 4.0 for the legs. The output should be 5.00. If it's not, you have some debugging to do.

Run the program again. Test with values 5.0 and 8.0. (Determine what the expected output is **before** you run the program.)

6. **Editor:** Close program1.c.

7. **Terminal:** Copy program1.c to a new file named lab1.c. The command is:

```
cp program1.c  lab1.c
```

Now is a good time to show the TA or instructor that you know how to use Kate.

8. **Editor:** Write a C program that will solve the following problem.

An instructor wants to check the progress for each student in his class. The students have taken 4 exams and each exam score is a floating point number between 0 and 100 (inclusive).

The instructor wants to see numeric average of the 4 scores and the letter grade that corresponds to the exam average. Here is the grading scale:

0 <= average < 60	F
60 <= average < 70	D
70 <= average < 80	C
80 <= average < 90	B
90 <= average < 100	A

Echo print the 4 exam scores. Display the average with 2 decimal places. Display the letter grade as a capital letter.

The program should repeat the process of inputting 4 exam scores, calculating and displaying the average and letter grade, until the instructor enters -1 0 0 0 for the exam scores. (Note: the score sequence -1 0 0 0 is called the "sentinel".)

Don't display the sentinel in the output.

9. **Terminal:** Compile, debug, and compile again until the program compiles correctly.

   ```
   gcc lab1.c -o lab1                   to compile

   ./lab1                               to run
   ```

   Debug as needed until the program runs correctly with 2 students.

10. **Canvas:** Download the file "testdata.dat" from this assignment page in Canvas. Save the file as a plain text file in the same folder as your lab1.c program.

11. **Terminal:** Run the program again... this time using "input redirection".

    This will take all input from the file you specify (instead of the keyboard).

    ```
      ./lab1 < testdata.dat
    ```

    You can open the data file in your editor and determine the expected output of the program (shown below is an example of how the output for each student should be formatted). Compare the expected output to the actual program output.

    ```
    Exam Scores:   90  82  78  94
    Average:       86.00
    Letter Grade:  B
    ```

    Close the program and data file once you have the correct output.

12. Log off before you leave the lab.