APS 105 — Computer Fundamentals

Lab 1: Simple Computation in C and handing in Assignments

Winter 2021

The goal of this laboratory is to create a simple C program that receives typed input from the user of the program, and calculates a simple output. In addition, we'd like you to get used to the methods of handing in assignments in this course, which have two parts -

1. Marking of program *style* and questions in the lab by your TA, and
2. Electronic submission of your program for grading by a computer program.

All labs in this course are due at 11:59 p.m. on the day of your scheduled lab period. This means that you must have both been graded by your TA, and you must have gone through the submission process described in this lab. The same process will be used in every one of the coming eight labs! Lab 1 is due in the second week of the term.

Preparation

Read through this entire document carefully, and do the work to create the program that is described in the **Program to Write** section below, and try to make it work. You can do this on your own home computer using CodeLite as described in Lab 0. If you are having difficulty making your program work prior to the lab period, you can do one of several things:

1. Read the class bulletin board on Piazza, to see if others had similar problems. If you don't see anything helpful there, ask a question. Don't ask for, or ever give, though, a solution in the form of the full computer program!
2. The TA in your lab can give you help.
3. A TA will be available in your weekly tutorial period, simply to give help.

Program to Write

Create a C-language program that reads in three integer numbers (call them inputNumber1, inputNumber2 and inputNumber3) from the user through the keyboard. The program should then output half of their sum (inputNumber1 + inputNumber2 + inputNumber3)/2, twice their product (inputNumber1 * inputNumber2 * inputNumber3)*2, and average.

Below we give a description of the example inputs and outputs as they must be in your program. The text that is to be entered by the program user is displayed using a **bold** font. The text <**enter**> stands for the user pressing the enter key on the keyboard. On some keyboards, the enter key may be labeled return. When you execute your programs, the user's text will not be displayed in bold and <enter> will not be shown.

Here is an example output from an execution of the program:

Enter First Number: **3**<**enter**>

Enter Second Number: **9**<**enter**>
Enter Third Number: **4**<**enter**>
Half the Sum: 8.00
Twice the Product: 216.00
Avg: 5.33

Here is second example output from an execution of the program:

Enter First Number: **1**<**enter**>
Enter Second Number: **3**<**enter**>
Enter Third Number: **4**<**enter**>
Half the Sum: 4.00
Twice the Product: 24.00
Average: 2.67

Notes:

1. You can assume that the user enters valid numbers.
2. For output of the double numbers, be sure to just use the output format code %.2lf.
3. **Important:** The marking program will be looking for the exact letters as described in the output above, including the capitalization. When you test your program using the exercise program given below, you will see that it is expecting the output to be exactly this, so you'll have to use it to see if you have this output correct.
4. Notice that there is a single space after the colon (:) in each output line.

Grading by TA and Submitting for Program for Auto-Marking

You *must* put your C program in a file named Lab1.c (be sure to make the capitalization in the file name correct). You can set the name of the file when you first create the project (as described in the CodeLite document) or change the name of the **main.c** file in CodeLite by right clicking on the name, and selecting **Rename...** You can find that file by looking in the Projects folder (which will be in your home directory on a mac or Linux, and in your Documents directory on Windows) and then looking in the folder with the name of the project in it.

There are a total of **10 marks** available in this lab, marked in two different ways:

1. **By your TA, for 4 marks out of 10**. Once you are ready, show your program to your TA so that we can mark your program for style, and to ask you a few questions to test your understanding of what is happening. Programs with good style have been described in class, but briefly, they are:
    - Clear comments that describe what is happening in the program. It should include the lab purpose and your name at the top.
    - Good choices for variable names that indicate their purpose (e.g. **inputNumber1** or Twice**ProductOutput** in the case of this lab). Please adopt the following naming convention illustrated above - if have you a variable that is described by multiple

words, user lower case for the first letter of the first word, and Upper case for all subsequent words - e.g. **longestLengthString** or **closeWindowDelay**.

- Properly indented code that has appropriate spacing between lines for readability.

2. **By an auto-marking program for 6 marks out of 10**. You must submit your Lab1.c program file through the ECF computers for marking. We will use a software program to compile and run your program, and test it with different inputs. Long before you submit your program for marking, you should run the **exercise** program that compiles and runs your program and gives it sample inputs, and checks that the outputs are correct. To do this you should do the following:

   - Open a terminal window on ECF, using xterm, as described in the Getting Started with Linux and ECF document from Lab 0.
   - Change into the directory (folder) containing your Lab1.c file, using the Linux cd command.
   - Run the following program by typing:

     /share/copy/aps105s/lab1/exercise

     This program will look for the file Lab1.c in your directory, compile it, and run it on a *some* of the test cases that will be used to mark your program automatically later. If there is anything wrong, the exercise program will report this to you, so read its output carefully.

     A key part of what you are to learn in this lab is the use of this program - most software programs give this kind of report. Read through the output of the **exercise** program and see if it is happy with everything, or if it is reporting an error. If there is an error, it will say so, and then it is up to you to fix what is wrong and try again. You'll need to do this for every subsequent lab.

3. Once you have determined that your program is as correct as you can make it, then you must submit your program for auto-marking. This must be done by the end of your designated lab period as that is the due time. To do so, go into the directory containing your Lab1.c file (and make sure this is the right place!) and type the following command:

   /share/copy/aps105s/lab1/submit

   This command will re-run the exercise program to check that everything looks OK. If it finds a problem, it will ask you if you are sure that you want to submit. Note that you may submit your work as many times as you want prior to the deadline; only the most recent submission is marked.

Important Note: You must submit your lab by 11:59 p.m. on the day of your assigned lab period. Late submissions will not be accepted, and you will receive a grade of zero.

You can also check to see if what you think you have submitted is actually there, for peace of mind, using the following command:

/share/copy/aps105s/lab1/viewsubmitted

This command will download into the directory you run it in, a copy of all of the files that have been submitted. If you already have files of that same name in your directory, these files will be renamed with a number added to the end of the filename.

After the Final Deadline - Obtaining Automark

After all lab sections have finished, a short time later, you will be able to run the automarker to determine the automarked fraction of your grade on the code you have submitted. To do so run the following command:

/share/copy/aps105s/lab1/marker

This command will compile and run your code, and test it with all of the test cases used to determine the automark grade. You will be able to see those test cases' output and what went right or wrong.