EEE125 – C Programming: Program Writing Assignment

At the beginning of the course, you were introduced to the make-up of this module. It was explained that, in Semester 1, the module would be assessed by means of a Blackboard test (worth 33.3% of the total module marks) and this C program writing assignment (also worth 33.3% of the total module marks). In Semester 2, you will have a final in-class exam which covers both Matlab and more advanced C programming (this covers the remaining 33.3% of the marks for the module).

For this assessed exercise you are asked to write a program in 'C' to carry out the particular task detailed here.

1 Administrative details

Your program must be handed in via Blackboard. **The hand-in date for this exercise is <u>16:49 (4.59pm) on Tuesday 13th December</u>.** You can hand your work in there anytime between the beginning of week 10 and the deadline. You are very strongly urged to not leave your submission to the latest possible moment to avoid computer related submission errors.

- Late submission will result in a deduction of 5% of the total mark awarded for each working day after the submission date, this is Faculty policy. (Working days – Monday to Friday - <u>include</u> working days within standard vacation times). The only exceptions to this will usually be where illness or other serious extenuating circumstances have meant missing the hand-in date (medical evidence will often be needed to sustain this exception). In such circumstances, you **<u>MUST</u>** submit an **<u>Extenuating Circumstances</u>** form (<u>not</u> a self-certification form), available from www.sheffield.ac.uk/ssid/forms/circs, (before completing such a form make sure you read the explanatory notes here first: http://www.sheffield.ac.uk/ssid/forms/circsnotes) hand in to the department Student Support Office.
- If you believe you have good reason to ask for an extension to this hand-in date, you may discuss the circumstances with Matthew Hobbs and, at the discretion of the department, a later hand-in date may be negotiated as per standard departmental policies. But please do the discussing **well** in advance of the original hand-in date expiring. Please note: being disorganised or lazy and so failing to meet the hand-in date by simply leaving tackling the exercise until it's too late is <u>not</u> a "good reason" and will <u>not</u> normally result in an extension of the hand-in date!

---

2 What your program should do

Let us pretend that you have been asked to write a computer program by a company that make resistors. They make large quantities of resistors, so, in order to check the quality of their resistors, they have some sample resistors and measure and record each resistor's value. Once they have a set of such sample measurements, they then want to enter the resistor values into a computer program to analyse the measurements. **<u>You</u>** are going to write that computer program.

Specification for your program:

- Firstly the program should ask the user to **enter** the values of the resistors measured and **collect** them.
- Next it should **display** all the resistor values entered on the screen.
- Next, the program should ask the user to choose one of the following three options:

  1. calculate and display the **mean value** for this set of resistor values,
  2. calculate and display the **sample standard deviation** within this set of resistor values,
  3. identify and display the **largest value** in this set of resistor values.

- After this, the program should allow the user to go back and repeat the previous step, allowing the user to either choose one of the three options again or move on.
- Finally, the program should ask the user if they wish to start the program again and enter new resistor values, or to end the program.

The program you write must be written in C using the techniques you have been taught in the classes so far.

2.1 How to calculate the mean value for a set of data.

Consider that you have a set of ***N*** sample data values which we will represent as follows:

$$\boldsymbol{x_1, x_2, x_3, x_4, \dots x_N}$$

We will use ***N*** to represent the total number of actual values in our set so ***N = 5*** in the above example. The following equation would allow us to calculate the **mean value**, $\bar{x}$ for the set of data values above:

$$\bar{x} = \frac{\sum_{n=1}^{N} x_n}{N}$$

Where, if N = 5:

$$\bar{x} = \sum_{n=1}^{N=5} x_n = \frac{x_1 + x_2 + x_3 + x_4 + x_5}{5}$$

So, let us consider the following example. Our data set might be:

0.0034 , 0.0129 , 0.0076 , 0.0012 , 0.0048 Ω

Applying the equation given above, the **mean** of this set of values would be:

$$\bar{x} = \frac{(0.0034 + 0.0129 + 0.0076 + 0.0012 + 0.0048)}{5} = 0.00598\,\Omega$$

2.2 How to calculate the sample standard deviation for a set of data.

Consider again a set of ***N*** data values, then the sample **standard deviation *S***, can be found as follows:

$$\boldsymbol{s} = \sqrt{\left\{\frac{\sum_{n=1}^{N}(\boldsymbol{x_n} - \overline{\boldsymbol{x}})^2}{\boldsymbol{N} - \mathbf{1}}\right\}}$$

So, let us consider the following example. Our data set might be:

0.0034 , 0.0129 , 0.0076 , 0.0012 , 0.0048 Ω

Applying the equation given above, the **sample standard deviation**, $\boldsymbol{s}$, of this sample set of values (and using the value for **mean**, $\bar{\boldsymbol{x}}$ already found for this same example set of samples in **2.1** above: $\bar{\boldsymbol{x}}$ = 0.00598) would be:

$$\mathrm{s} = \sqrt{\left\{\frac{(0.0034 - 0.00598)^2 + (0.0129 - 0.00598)^2 + (0.0076 - 0.00598)^2 + (0.0012 - 0.00598)^2 + (0.0048 - 0.00598)^2}{5 - 1}\right\}}$$

$$\mathrm{s} = \sqrt{\left\{\frac{(-0.00258)^2 + (0.00692)^2 + (0.00162)^2 + (-0.00478)^2 + (-0.00118)^2}{5 - 1}\right\}}$$

$$\mathrm{s} = \sqrt{\left\{\frac{6.6564 * 10^{-6} + 4.78864 * 10^{-5} + 2.6244 * 10^{-6} + 2.28484 * 10^{-5} + 1.3924 * 10^{-6}}{5 - 1}\right\}}$$

$$\mathrm{s} = \sqrt{\frac{8.1408 * 10^{-5}}{4}}$$

$$\mathrm{s} = \sqrt{2.0352 * 10^{-5}}$$

$$\boldsymbol{s} = \mathbf{0.0045113}\ \boldsymbol{\Omega}$$

2.3 <u>Finding the largest value in a set</u>

Let us consider the following example resistor data. Our data set might be:

320.14 , 219.62 , 453.01 , 92.94 , 765.04 , 107.9 Ω

Therefore, for this set of data, the **largest value = 765.04 Ω**

2.4 What your program should do in more detail:

i. Firstly the program should announce its purpose to the user.

ii. Then the program should ask the user how many sample resistor values the user wishes to enter. The program should only accept a whole number in the range **2** to **10** only. The program should reject any entry that is not within this range.

iii. Next the program should prompt the user to enter all the resistor values. Resistor values must be within the range +1 mΩ (1×10^{-3} Ω) to +1 MΩ ($1x10^{6}$ Ω) only. The program should reject any entry that is not within this range.

iv. The program should then display all these sample values on the screen, laid out neatly.

v. Next, the program should ask the user to choose one of the following three options to perform:
   1. calculate & display **mean value** of this set of sample values,
   2. calculate & display the sample **standard deviation**,
   3. identify & display the **largest value** in the set

vi. The program should then carry out the behaviour chosen in step v. above.

vii. Next, the program should offer the user the chance to perform another option (or the same option again) from the list in part v. above on the same, already entered, sample data, or allow the user to move on (which will prompt the final step below).

viii. Finally, the program should ask the user if they either wish to start the program again and enter new resistor values, or to end the program.

2.5 Some important advice to consider

- Your program **<u>must</u>** be written in 'C' and not 'C++'. ('C' is taught in EEE125, not C++, so sticking to techniques you have been taught will mean you will be OK). Make sure you save your file with a '`.c`' filename extension <u>NOT</u> '`.cpp`'. When using Dev-C, make sure when saving your file that the "**Save File**" dialogue has the "**Save as type**" drop-down menu set for "**C source files (*.c)**".

- By the time this sheet is handed out, you will have already covered, in video lectures and lab sheets, **<u>all</u>** the material necessary to allow you to write a suitable program. However, background reading of your own may be of additional help.

- Make your program code as easy to "**read**" (by a human) as possible. For example, partition it (in other words, break it up) by making your own **functions**. (Programs that consist of everything inside the single function **main** only will earn fewer marks than programs that are divided into more than one function). (See example solution to Lab Sheet 12 exercise 4 (file lab12-ex4d.c) in the **EEE125 BLACKBOARD** course, in the '**Lab Classes**' section, in the '**Lab Sheet 12**' folder showing use of functions). Use well-chosen **identifiers** for **function names** and **variable names** so that the purpose of a particular function or variable is hinted by its name. **<u>This example solution will be made available during week 8 of the semester.</u>**

- Make sure the program is well **commented** (i.e. using /* */ marks) so that the purpose of each part is clear. These comments should be concise. Note: there is no need to use comments to explain how a particular 'C' construct works, you can assume the reader understands the 'C' language itself (e.g. you don't need to explain how a 'for' loop works as such). However you should add comments to explain what your program aims to achieve with particular 'C' constructs where it isn't immediately obvious (e.g. explain what useful task a particular 'for' loop is performing for you in your particular program). This is an important skill needed by programmers to ensure their program code is 'readable' and 'understandable' by others who may have cause to examine them or modify them later. You have been given clear guidance in the lectures about how to lay out a program and how to add blocks of comments – check your lecture slides to see what was said.

- A well designed program should make it relatively straightforward for some other programmer to modify your program at a later date.

- Don't forget: **planning** your program thoroughly **on paper** first is by far the most effective way of quickly writing a good program. Don't rush to coding at the PC too soon. Start by breaking the task down into manageable portions, then plan the sequence of events for each portion with flow charts and pseudo code etc. Flow charts in particular will help you identify the appropriate flow control statements ('while', 'do..while', 'for', 'if' etc.) to use.

- Test the output from your program by using various sets of data. You have been given some example data and the expected results in sections 2.1 to 2.3 above. Don't forget to check your program's behaviour with illegal data (i.e. data outside the range defined for input – resistances smaller than +1 mΩ ($1x10^{-3}$ Ω) or larger than +1 MΩ ($1x10^{6}$ Ω)). Does it behave appropriately?

- Don't forget, “divide and conquer” is a wise approach. Break the task down into manageable portions and tackle them one at a time, don't try and code up the whole program in one go only to find it does not work. The task of finding a fault (or more likely many faults) in a large program is like ‘*looking for a needle in a haystack*’ - make sure you have only a small ‘*haystack’* to search! Start off with a small program which only does the first few steps of the task, get that working first. Next add a bit more to your program and get that working too. Continue in this way until the program can carry out the complete task.

- Above all, be sure to hand something in by the deadline. If you examine the marking scheme carefully (see later), you will see that only a small proportion of the marks are available for **accuracy** of **results**, etc. there are many more marks available for other aspects of the work. If you only manage to produce a program that asks for, stores and re-displays the resistance values, and also calculates the mean value but does not even attempt to calculate anything else, then you can at least be given some marks for that much (which is better than no marks if you hand in nothing at all).

- If you need help:
  a) Refer back to the teaching resources on EEE125 Blackboard page. If there is a particular construct you wish to revisit, please refer to the teaching plan regarding where to find it within the video lectures and lab sheets.
  b) Please talk to a demonstrator / GTA during a timetabled afternoon computer lab session (either at the Tuesday or Friday session). You have also all been allocated a named demonstrator / GTA for e-mail contact outside of the scheduled lab sessions as well. However, it is preferred if you could bring your questions to the lab sessions themselves (you are likely to find this more beneficial).
  c) Make use of the 'C' books in the library and any web resources. Some books suggestions can be found on the EEE125 Blackboard page.

3 What should you hand in?

- You should hand your ‘C’ program file itself in via **Blackboard (see instructions in section 3.2 below)** and it should be in the form of a **source file** of ‘C’ code (that is the **ASCII text file** created in the usual way by using the **Dev-C** Editor when writing a program and saved with the filename and the file name extension for C Source Files: “**`resistance.c`**”). You **MUST** name the file “**`resistance.c`** " Please use this name only, it helps to process your work quickly. It must not be called “**`resistance.cpp`**"! (*.cpp implies a C++ file and you must not write in C++!) **Use of an incorrect filename will result in lost marks.**
- You **MUST** write your **Registration number** into a comment string as the **first line of your program** like this:

```
/* My Reg Number: 170113134 */
#include <stdio.h>
```

…and so on. You are not required to include your name (all programs will be marked anonymously, so please avoid including your name).

- If you choose to write and test your program using a `C' compiler other than the Bloodshed Dev-C environment provided for you on the University's Computer network (as used in the lab classes), you are strongly recommended to bring the final version of your source code into the University and check it compiles and runs correctly using the Dev-C compiler on the University's Computer network **before** handing it in. It is this same Dev-C compiler that will be used to compile and test you program in order to assess it.

3.2 To submit your program via Blackboard:

To submit your program via Blackboard, follow these instructions (follow them carefully as you only have **one** chance to submit!):

a) You only have <u>**one**</u> chance to submit your file, once submitted you cannot change it, so be sure you are finished and have chosen the correct file to submit and attach it correctly.

b) Once you have finished your C source file and you are happy that it is ready to submit it, open a web browser and either login in to **Blackboard** in the usual way.

c) On the "**EEE125 Programming**" Blackboard course home page, look for the '**Assessment and Feedback**' section in the left-hand menu.

d) In the '**Assessment and Feedback**' section of this left-hand menu you should see a heading called '**Assessment**'. Click on this.

e) You should see an item titled "**C Programming Writing Assignment - Submission**" (this will become available at the beginning of week 10), make sure you <u>**click on this title**</u>.

f) Within this submission page, first enter your registration number as the "submission title".

g) Next, choose the file you wish to submit from your computer, Google Drive or otherwise. Make sure you select your 'C' source file **`resistance.c`** in the conventional way. (Make absolutely sure you are selecting the **C** source file - the one you typed – not any of the other associated files the compiler generates when you compile and run your program). To do this, look at the icon displayed beside the file name, make sure you pick the file with the icon showing a small blue '**.c**' in the corner. Double click on your **`resistance.c`** file to attach it. When you have choose your file, press "**Upload"**.

h) A preview of your file should now appear. A preview of your file will appear; please **check that the file you uploaded is the correct one**. If it is, press **"Confirm"**. After you have done this, you should get an on screen confirmation and an e-mail receipt from TurnItIn; **please keep this receipt**.

3.3 Marking Scheme

On the next page you will see the marking scheme that will be used to mark your work. Initially, as you can see, you will be marked in percentages, “full marks” would be 100%.

Aspect	Comment	Mark
Overall Program Design	Does the program try to meet the **specification** laid out in section 2 of this sheet? Is it designed to do **all** of the things asked for?	20
Choice of Variables	Have variables been created using appropriate types and do they make efficient use of storage space?	5
Quality of \`C' code	Have the many useful ‘C’ **language** features discussed in lectures and tutorials been exploited? Are the ‘C’ **constructs** used appropriately?	10
Readability of code	Is the program's purpose easily understood from the way it's structured e.g. does it have an appropriate hierarchical form (exploiting **functions**?). Is it well (concisely and clearly) **commented?** Have sensible, self explanatory, **identifiers** been chosen for **function** names and **variable** names?	5
Use of Functions	Have user functions other than **main** been written at all? Are they used sensibly? Is the choice of return types and parameters sensible?	20
Compile & Run	Does the program compile without errors or serious warnings? Does it run without crashing etc?	10
Accuracy of results	Are the results it produces correct?	20
User interface	Does the "**user interface**" clearly inform the user what she or he must do at each stage? Are the outputs presented clearly and in an appropriate fashion?	10
	Total	100

4 Unfair Means

The basic principle underlying assessed work is that **the work submitted for assessment must be entirely your own.** No one objects to you discussing the principles of C programming in general with others, but: **plagiarism** and **collusion** are **<u>not allowed</u>**. You must **<u>not</u>** discuss the details of how you will do this exercise with anyone other than Matthew Hobbs or one of the EEE125 programming class demonstrators (GTAs). In the context of this exercise, unfair means would include:

1 You **<u>MUST NOT</u>** allow anyone else to **write** or **dictate to you** your program, **in whole** or **in part**.

2 You **<u>MUST NOT</u>** look at someone else’s program for this exercise or **copy** from someone else’s work or exchange **emails** or **internet chat messages (Weibo, WhatsApp, Facebook or other forms of social media)** or by sharing your code via file sharing sites (like **Postbin** or similar**)** with someone else which include **code** from either your program or their program for this exercise. Note: this includes with the lab Demonstrators or students in other years.

3 Likewise you **<u>MUST NOT</u>** **write program code for, or share program code with**, another student on this course or from previous years of study or anyone else.

4 You **<u>MUST NOT</u>** team up with others to **write one program** (**in whole** or **in part**) together, then **all** hand in programs containing that **same** code (**in whole** or **in part**).

5 You **<u>MUST NOT</u> copy** portions of code from any sources (that are not of your own creation) such as a website or a book or from someone else's computer, website (even from a website where the language is not English) or memory device. (However: **you are granted exceptional permission** to copy from **any of the examples of code that have been given** to you **during this course** as lecture or lab sheet examples – if you do this, then attribute the source of those portions of code by clearly referencing it using a comment like this **`/* taken from EEE125 examples */`** **immediately before** and **immediately after** the code concerned.)

Remember: the basic principle underlying assessed work such as this is that **the work submitted for assessment must be entirely your own. We wish to only give marks for programs written by <u>you alone</u>!**

If you are in any doubt about what might constitute unfair means in the context of this exercise then please discuss any areas of uncertainty with **<u>Matthew Hobbs</u>**. If you experience any problems with this assignment, then seek help from Matthew Hobbs or the demonstrators (GTAs) present during lab classes.

Please note: We will be using a sophisticated program specially designed to detect plagiarism in a set of 'C' programs to help detect evidence of such unfair means in the assessed work that you hand in, so, please don't take the risk of copying work, submit only work done by **<u>you</u>** entirely **<u>on you own</u>**. Where we suspect unfair means to have been used**, the department reserves the right to give <u>zero marks</u> to <u>all</u> individuals concerned** and/or **refer you to the University Discipline Committee** and/or **place a note in your student record file** (We have caught people breaking these recommendations before and done all these things, so do please take note of this warning!).

Good Luck!

Matthew Hobbs