AU 2018 CSE 2421 LAB 2

Due: Monday, September 24th, by 11:30 p.m.

Objectives:

· Pointers
· Pointer arithmetic
· Dynamic memory allocation
· Functions: parameters, return values and reusability
· Arrays (dynamically allocated)
· Software engineering principles in C

REMINDERS and GRADING CRITERIA:

- This is an individual lab.

- **Every lab requires a Readme file** (for this lab, it should be called **lab2Readme – use this name, without an extension or any other modification**). This file should include the following:
  - · Your name
  - · Total amount of time (effort) it took for you to complete the lab
  - · Short description of any concerns, interesting problems or discoveries encountered, or comments in general about the contents of the lab
  - · Describe how you used gdb to find a bug in your program while debugging it. Include how you set breakpoints, variables you printed out, what values they had, what you found that enabled you to fix the bug.

- You should aim to always hand an assignment in on time or early. If you are late (even by a minute – or heaven forbid, less than a minute late), you will receive 75% of your earned points for the designated grade as long as the assignment is submitted by 11:30 pm the following day, based on the due date given above. If you are more than 24 hours late, you will receive a zero for the assignment and your assignment will not be graded at all.

- Any lab submitted that does not compile – without errors - and run **WILL RECEIVE AN AUTOMATIC GRADE OF ZERO**. If your code generates warnings, you will lose credit for each warning generated, depending on the warning. No exceptions will be made for this rule - to achieve even a single point on a lab, your code must minimally build (compile to an executable without errors) on stdlinux and execute on stdlinux without crashing, using the following command:

```
% gcc -ansi -pedantic  lab2.c -o lab2
```

Add the -g option to use the gdb program

- You are welcome to do more than what is required by the assignment as long as it is clear what you are doing and it does not interfere with the mandatory requirements.

LAB DESCRIPTION

DATA SET CALCULATOR (100%) **Mandatory filename → lab2.c**

PROBLEM:
The user of your program will use it to do some elementary floating point calculations for an unknown number of simple data sets. Each data set consists of a list of floating point values, but the number of values to be entered in each data set will be specified by the user, so you do not know in advance how many values there will be in each data set. Additionally, you do not know in advance how many data sets the user will enter. Therefore, you cannot use a static array to store any of these values.

- First, you should prompt the user to enter the number of data sets. The user will enter an integer greater than or equal to 1 to indicate the number of data sets. (NOTE: Make a point to test your program to ensure entering just 1 data set works.)

- You should then print a single prompt for all the data sets (do not output a separate prompt for each data set) to the user to enter the number of floating point values in each data set (which the user will enter as an integer greater than 0), followed by the floating point values themselves on the same line (newline will follow the last floating point value on the line). You can assume that the user will enter the input in this format, so you do not need to check to make sure that the format of the input meets this description, and you do not need to reject input which is not properly formatted. You can also assume that the user's input is correct. That is, that the number of data sets entered is actually the number of data sets in the input and that the integer entered first on each line to indicate the number of values for the data set given on the rest of the line is actually the number of floating point values that follows on the same input line. Your program needs to read the user input and store the floating point values in each data set in a dynamically allocated array of the appropriate size. If you do not completely understand this description jump to the bottom of this file and check out the example data. (NOTE: Make a point to test your program to ensure that entering just 1 floating point value on a line works. e.g. there is only one float value in the dataset and all of the functions described below calculate correctly.)

- After getting the values in each data set, your program should **repeatedly do the following two things:**

1. Print the following prompt:

   Enter the number of the data set on which you wish to do calculations:

   The user will enter an integer value, followed by newline, in response to this prompt (the user will enter 1 for the first data set, 2 for the second, etc.), then based on the value entered, your program must be able to access the values in the appropriate data set in the dynamically allocated storage which you have created. Then do what is described immediately below.

2. Your program should then prompt the user to choose one of the following options for a calculation based on the data set chosen by the user (ask the user to enter one of the seven numbers, followed by enter):

1) Find the minimum value.
2) Find the maximum value.
3) Calculate the sum of all the values.
4) Calculate the average of all the values.
5) Print the values in the data set.
6) Sort the data set, and store the sorted values in the same addresses in memory where the unsorted data set was stored before calling this sorting function.
7) Exit the program.

After the user selects one of the seven options, your program should perform the necessary calculation or print the specified data or terminate the program. The program should output the result with an appropriate message, for example:

The maximum value in data set 4 is: 567.37

The results for options 1, 2, 3, and 4 should be printed out as floating point values with 3 digits of precision, and the result for option 5 should be to output the values in the data set in the order in which they were input, with three digits of precision for each value, and with any two values separated by a tab (\t). The output for option 6 should be the values output in sorted order, by calling the function for option 5

➢ After your program outputs the result of the operation, **it should prompt the user again to select one of the data sets**, and then one of the seven options until the user selects option 7 to exit the program.

CONSTRAINTS:

· You cannot use statically declared arrays for this lab, as explained above.
· Your code should work correctly for ANY NUMBER of input data sets (including just 1), and for any number of values in each data set (including just 1 and up to the limits of available memory, of course), and these numbers are not known in advance.
· You will need to use pointers to do all of the calculations (options 1 to 6). **You will not be able to access any of the allocated storage space using indexes, as is usually done for a conventional array, but only by using pointers and pointer arithmetic.**
· Use **a separate function** to do each type of calculation (some of these functions might call other ones for example, the code for option 4 might call the function for option 3).
· Use a function to get the input from the user about the number of data sets, and a separate function to get the number of values in each data set, and to read in the values in the data set (so there will be two functions total to get the input for the data sets).. Also use a separate function to get the user's choice of the calculation to perform. Use a different function for each of the 6 calculation options, and also use a separate function to exit the program (after deallocating all dynamically allocated memory, as described below). Be sure to document what each function does.
· Even though you will have several functions, all code must be in a single file named lab2.c
· Your program should be able to work whether it receives input from the command line (e.g.keyboard) or via redirected (stdin) input from a file. Note: Since input from a file is being accomplished via redirection of stdin from the command line, no changes to your code should be necessary. Testing it both ways would be a great way to verify this.
· Be sure to follow the Software Design Principles in C identified in the slide set posted on Carmen.

LAB SUBMISSION

You should submit all your lab assignments to Carmen Canvas by running a firefox web browser, logging in to Carmen, and uploading your zipped folder with your files, as explained for Lab 1

Be sure to submit the following files: lab2.c lab2Readme
You should zip the folder with your files before submitting it.
Do not submit an executable version of your program, but only C source code!

See below for a sample data set input to the program. Your program should work for all valid data sets not just the one given below. Be creative with your test data!!! ☺

Sample Data Set Input (This does not include user responses to the prompts to select a data set or to select an operation to perform)

```
6
8   3.45 2.37 85.32 34.5 569.45 335.2 193.4 74.39
6   23.45 32.37 185.32 364.5 179.4 144.39
7   35.45 121.47 42.32 44.5 249.75 385.9 113.4
4   44.45 567.37 311.32 131.5
9   3.25 332.37 851.32 314.4 249.47 385.5 173.65 154.32 244.47
11  22.45 2.37 85.32 34.5 569.45 335.2 193.4 74.39 122.45 413.2 89.32
```