AU 2016 CSE 2421 LAB 2

Assigned: Wednesday, September 14th
Due: Thursday, September 22, by 11:30 p.m.

Objectives:

· Pointers
· Dynamic memory allocation
· Functions
· Arrays (dynamically allocated)

REMINDERS and GRADING CRITERIA:

- You are allowed to work in pairs for this assignment (but your partner has to be enrolled in the same CSE 2421 section).
- **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:
  - · Student name(s) - to avoid the 10% deduction (as explained in the course syllabus) if working with a partner
  - · Effort distribution for each contributor (assumed to be 100% if you are not working with a partner)
  - · Total amount of time to complete the entire lab
  - · Short description of any concerns, interesting problems or discoveries encountered, or comments in general about the contents of the lab
- You should aim to always hand an assignment in on time. 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 and run **WILL RECEIVE AN AUTOMATIC GRADE OF ZERO**. 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) on stdlinux and execute on stdlinux without crashing, using the following command:

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

- 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 SETS CALCULATOR (100%) **Mandatory filename →** **lab2.c**

PROBLEM:

- The user of your program will use it to do some elementary calculations for an unknown number of simple data sets. The data in each set consists of a number of floating point values, but the number of

the values to be entered will be input by the user, so you do not know in advance how many values there will be in each data set. Therefore, you cannot use a static array in C to store these numbers, because a statically declared array has a fixed size, which cannot be changed after the program begins running.

- First, you should prompt the user to enter the number of data sets. The user will enter an integer greater than 1 to indicate the number of data sets.

- You should then prompt 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 sets in the input, and that the integer entered first to indicate the number of values for each data set is actually the number of floating point values which 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.

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

- Select the data set to do operations on by number, with 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.), and 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, and then do what is described immediately below.

- 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 six 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. Exit the program.

    After the user selects one of these six options, your program should do the necessary calculation or printing of the data based on the user's choice (or terminate the program), and output the result with an appropriate message, for example:

    The maximum value in the data set is: 569.45

The results for options 1, 2, 3, and 4 should be printed out as floating point values with 2 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 two digits of precision for each value, and with any two values separated by a tab (\t).

➢ 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 six options, until the user selects option 6 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, and for any number of values in each data set (up to the limits or 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 5). **You cannot 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).
· Also use a function to get the input from the user about the number of data sets, to get the number of values in each data set, and to read in the values in the data set; use a separate function to get the user's choice of the calculation to perform. Be sure to document what each function does.

LAB SUBMISSION

You should submit all your lab assignments electronically using the submit command.

Be sure that all the files you are submitting are in the directory from which you run submit!!!!! Here are the submit commands for the lab2 assignment for Mike Green's three sections of CSE 2421 this semester (the red type is to aid clarity; all the characters in the command can be typed in black):

9:10 section:	submit c2421ad lab2 *files-to-submit*
10:20 section:	submit c2421ab lab2 *files-to-submit*
1:50 section:	submit c2421aa lab2 *files-to-submit*

where *files-to-submit* is a list of the files that make up the lab, with file names separated by spaces only (no other punctuation). For more information about the submit command, review that section in your prelab practice problem.

Be sure to submit the following files: lab2.c lab2Readme
Do not submit executable versions of your program!

See the next page for sample data set input to the program.

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
2
1
3
2
4
3
5
4
6
5
6
```