LAB 6 — Command-Line Arguments

Problem A

1. Specification

Implement a simple calculator that accepts input in the following format and displays the result of the computation:

calc [operand_1] [operator] [operand_2]

The operands `operand_1` and `operand_2` are <u>non-negative integers</u>. The operator is one of the following: addition (+), subtraction (-), multiplication (x), division (/) and modulo (%).

Note: For the multiplication operator, use letter **'x'**. If you use the asterisk **'*'**, your program will not work properly.

2. Implementation

- The program to be submitted is named `calc.c`. **Use the given template `calc.c`** and fill in your code. Complete functions `main()` in file `calc.c`.

- Note that the command-line arguments are all strings. Therefore you may need to implement a function to convert a string to an integer to obtain `operand_1` and `operand_2`.

- Sometimes users may forget the command syntax and they may type only the command "calc". In that case, display the following reminder message:

  `Usage: calc [operand_1] [operator] [operand_2]`

- Other than that, assume that all inputs are valid. No error checking is required on inputs.

- You may define your own variables <u>inside</u> functions `main()`.   Do not use global variables.

- You may define and implement your own function(s) inside file `calc.c` if needed.

- **Do not use any C library functions (e.g., atoi).**

- To compile the program, use the following command: **cc calc.c –o calc**

- There must be at least a white space between an operand and the operator. That is how the command-line arguments are separated.

3. Sample Inputs/Outputs

See file `calc.io` for sample inputs and outputs.

Problem B

1. Specification

Write a program that accepts a set of integers as command-line arguments from the user, and then displays the maximum, minimum and average values of the input integers.

2. Implementation

- The program to be submitted is named mmavg.c. **Use the given template mmavg.c** and fill in your code. Complete functions main() in file mmavg.c.

- Note that the command-line arguments are all strings. Therefore you may need to implement a function to convert a string to an integer to obtain the input integers. An integer could be positive or negative. That is, it may have a + or – sign.

- Sometimes users may forget the command syntax and they may type only the command "mmavg". In that case, display the following reminder message:

  `Usage: mmavg int1 int2 int3 ...`

- Other than that, assume that all inputs are valid and the number of input integers is less than 100. No error checking is required on inputs.

- You may define your own variables <u>inside</u> functions main(). Do not use global variables.

- You may define and implement your own function(s) inside file mmavg.c if needed.

- **Do not use any C library functions (e.g., atoi).**

- To compile the program, use the following command: **cc mmavg.c -o mmavg**

- There must be at least a white space between the input numbers. That is how the command-line arguments are separated.

3. Sample Inputs/Outputs

See file mmavg.io for sample inputs and outputs.

Common Notes

- Complete the headers in the files to be submitted with your student and contact information.

- Assume that all inputs are valid. No error checking is required on inputs.

- You may use either pointers or array indexing or both.