Fall 2016 CST8234 – C Programming

Lab 01: Input and Output in C

Setup:

1. You are to use a Linux virtual machine, Ubuntu or Fedora recommended, 64-bit or 32-bit is OK
2. Create an account as your `username` (from Algonquin)
3. Create a directory `Lastname_01` You are going to develop your lab here!
4. Copy `01_Input_Skeleton.c` and rename it as `01_Input.c`
5. Copy `01_Input.sh` to your `Lastname_01` directory, you are going to use this script to test your first program

Program #1:

Write a small C program: `01_Input.c` that:

1. Giving a range, `MIN` and `MAX` , reads an integer number from the keyboard
2. If the format of the number is incorrect, it is not an integer, your program should print the message "`Incorrect input format`", and exits with `EXIT_FAILURE`
3. If there are extra character at the end of the number, your program should print the message "`Extra characters`", and exits with `EXIT_FAILURE`
4. If the number is out of range, your program should print the message "`Input number out of range`", and exits with `EXIT_FAILURE`
5. If the number was read successfully, and it is between the range, your program should print the number on `stdout` and returns `EXIT_SUCCESS`

The following demonstrates the execution of the program:

```
# ./01_Input
Enter a number in between [10-100]:  10
Read 10
# echo $?
0
# ./01_Input
Enter a number in between [10-100]:  a123
Incorrect input format
# echo $?
1
# ./01_Input
Enter a number in between [10-100]:  8.34
Extra characters
# echo $?
1
# ./01_Input
Enter a number in between [10-100]:  123
Input number out of range
# echo $?
1
                         SAMPLE TEST OUTPUT:  01_INPUT
```

In order to successfully complete this program and obtain all the marks, you will need to:

1. Use the macros `EXIT_FAILURE` and `EXIT_SUCCESS` define in the library `stdlib.h` to indicate unsuccessful or succesful termination of your program
2. Define `MIN` and `MAX` as macros in your program. Use 10 and 100 to test your program.

3. Write a funtion, with function prototype `int intGet( int, int )`
   (A) `intGet( )` should use `scanf( )` to read an interger from the keyboard
   (B) If the number supply is not recognized as an `int`, set a global variable `errorno` to `1`
   (C) If the number supply has extra characters at the end, that is not the `newline` character, set a global variable `errorno` to `2`
   (D) If the number supply is outside the range `MIN`, `MAX`, set a global variable `errorno` to `3`
   (E) Return the number read from `scanf()`
4. Write a function with function prototype, `int error( void )`
   (A) `error( )` checks the global variable `errorno` and displays an appropiate message to the user
   (B) `error( )` terminates with `EXIT_FAILURE`
5. Your `main( )` function should:
   (A) Ask a user for a number in a given range
   (B) Call `intGet( )` to read the number
   (C) If errorno is set, call `error( )`
   (D) Print the number read and exit with `EXIT_SUCCESS`
6. Check the exit status of your program from the command line `echo $?` should display a successful exit (value of 0) or unsuccesful (value of 1)
7. Your program should be compiled with the flags `-Wall -ansi -pedantic`

Program #2:

Copy your `01_Input.c` to a new program `01_Input_B.c` and
1. Modify your function `intGet( )` to read from the standard input `stdin`
   (A) Use the function `fscanf( )` instead of the function `scanf( )`.
   (B) Read from the `stdin` (keyboard) file descriptor
   (C) If `fscanf( )` reads the `EOF` character, return `EOF` otherwise return the number read
2. Modify your `main( )` function to
   (A) Read a variable `min` and `max` to use as a range, instead of the macros `MIN`, `MAX`
   (B) Use your function `intGet( )` to read numbers until you find an `EOF` character (`CTL-D`)
   (C) Keep a counter of the number of valid numbers read
   (D) Keep a counter of the number of invalid numbers read
   (E) Add up all the valid numbers
   (F) Display in a table all the information collected. This is an important requirement, you should present the information in a organized and easy to read format

The following demonstrates the execution of the program:

```
# ./01_Input_B < 01_num.txt
---------------------------------------------------
     Entry    Invalid      Valid     Number
         0                    *          3
         1          *
         2          *
         3                    *          1
         4          *
         5          *
         6          *
         7                    *         13
----------------------------------------------------
         8          5         3         17
                      SAMPLE TEST OUTPUT:  01_INPUT_B
```

No testing script is provided for this part of your lab. Review the specifications giving to you and make sure that you test your program appropriately.

Two files `01_num.txt` and `02_num.txt` are provided. Each file contains in the first line the min and max numbers to use as range and a number per line. Notice the redirection symbol (`<`) used to call your program using the file as input. You can use this idea to test your program without having to do all the typing! Your lab instructor may use different files to test your program. Be sure that you make some extra files, for example, use negative numbers, wrong numbers at the beginning / end of the file, large amount of data, large numbers, etc