Workshop #5

Worth: 1.5% of final grade

Breakdown

- Part-1 Coding: 10%
- Part-2 Coding: 40%
- Part-2 Reflection: 50%

Submission Policy

- Part-1 is due **1-day** after your scheduled LAB class by the **end of day 23:59** EST (UTC – 5)
- Part-2 is due **5-days** after your scheduled LAB class by the **end of day 23:59** EST (UTC – 5)
- Source (.c) and text (.txt) files that are provided with the workshop MUST be used or your work will not be accepted. Resubmission will be required attracting a **15% deduction**
- **Late submissions will NOT be accepted**
- **All work must be submitted by the matrix submitter – no exceptions**
- **Reflections will not be read or graded** until the coding parts are deemed acceptable and graded.
- **All files** you create or modify MUST contain the following declaration at the top of all documents:

```
/*****************************************************************************
            <assessment name example: Workshop - #5 (Part-1)>
Full Name  :
Student ID#:
Email      :
Section    :

Authenticity Declaration:
I declare this submission is the result of my own work and has not been
shared with any other student or 3rd party content provider. This submitted
piece of work is entirely of my own creation.
*****************************************************************************/
```

Notes

- Due dates are in effect **even during a holiday**
- You are responsible for **backing up your work regularly**
- It is expected and assumed that for each workshop, you will plan your coding solution by using the computational thinking approach to problem solving and that **you will code your solution based on your defined pseudo code algorithm**.

Late Submission/Incomplete Penalties

If any Part-1, Part-2, or Reflection portions are missing, the mark will be **ZERO**.

Introduction

In this workshop, you will code a C language program that implements simple validation on a series of user input values, and then analyzes the data to provide a statistical summary.

Topic(s)

- Logic

Learning Outcomes

Upon successful completion of this workshop, you will have demonstrated the abilities:

- to create a simple interactive program
- to code a decision using a selection construct
- to code repetitive logic using an iteration construct
- to nest a logical block within another logical block
- to describe to your instructor what you have learned in completing this workshop

Part-1 (10%)

Instructions

Download or clone workshop 5 (**WS05**) from https://github.com/Seneca-144100/IPC-Workshops
Note: If you use the download option, make sure you **EXTRACT** the files from the .zip archive file

1. Review the "Part-1 Output Example" (next section) to see how this program is expected to work
2. Code your program in the file named "**w5p1.c**" **IMPORTANT**: Do NOT use **arrays** in this workshop!
3. After the system library **#include**, and before the **main** function, define two (2) macros:

```
#define MIN_YEAR 2012
#define MAX_YEAR 2022
```

4. Inside the main function, **declare** two (2) **unmodifiable** integer variables "**JAN**" and "**DEC**" representing the first and last months of the year respectively (**initialize** "JAN" to **1** and "DEC" to **12**)
5. Display the title for the well-being log application
6. **Nest** inside an **iteration** construct the following:
   a) Display the following message:

```
>Set the year and month for the well-being log (YYYY MM): <
```

   b) **Read** from standard input (keyboard) the **year** and **month** (entered on the same line with a space between) assigning the input values to two **integer** variables (having **meaningful names** representing the data they store)
   c) Apply what you have learned about **selection** to define the necessary logic that will validate the values entered for the year and month.
      - The entered year value must be between **MIN_YEAR** and **MAX_YEAR** inclusive
      - The entered month value must be between **JAN** and **DEC** inclusive
      - If any of the above validations fail, the respective error message(s) should be displayed (see example output to see what each error message should display)
7. Step #6 should continue to iterate until a valid year and month value is entered
8. When a valid year and month is entered, display a message indicating the log starting date has been successfully set:

```
>*** Log date set! ***<
```

9. Display the log start date in the format: **YYYY-MMM-DD**
   YYYY: The year as 4-digits
   MMM: First 3-characters of the month name
   DD: The 2-digit day

   Note: The log will start on the 1st day of the month entered by the user

   Hint: You need to implement alternative/multiple selection to map the month integer value to the respective 3-character month representation. There are a couple of constructs available to you that will make this possible!

Part-1 Output Example (Note: Use the YELLOW highlighted user-input data for submission)

```
General Well-being Log
======================
Set the year and month for the well-being log (YYYY MM): 2011 1
   ERROR: The year must be between 2012 and 2022 inclusive
Set the year and month for the well-being log (YYYY MM): 2023 1
   ERROR: The year must be between 2012 and 2022 inclusive
Set the year and month for the well-being log (YYYY MM): 2022 0
   ERROR: Jan.(1) - Dec.(12)
Set the year and month for the well-being log (YYYY MM): 2022 13
   ERROR: Jan.(1) - Dec.(12)
Set the year and month for the well-being log (YYYY MM): 2011 0
   ERROR: The year must be between 2012 and 2022 inclusive
   ERROR: Jan.(1) - Dec.(12)
Set the year and month for the well-being log (YYYY MM): 2023 13
   ERROR: The year must be between 2012 and 2022 inclusive
   ERROR: Jan.(1) - Dec.(12)
Set the year and month for the well-being log (YYYY MM): 2022 2

*** Log date set! ***

Log starting date: 2022-FEB-01 always start with 1
```

Part-1 Submission

1. Upload (file transfer) your source file "**w5p1.c**" to your matrix account
2. Login to matrix in an SSH terminal and change directory to where you placed your workshop source code.
3. Manually compile and run your program to make sure everything works properly:
   ```
   gcc -Wall w5p1.c -o w5 <ENTER>
   ```
   If there are no errors/warnings generated, execute it: ***w5*** *<ENTER>*
4. Run the submission command below (replace **profname.proflastname** with **your professors** Seneca userid and replace **NAA** with your section):
   ```
   ~profName.proflastname/submit 144w5/NAA_p1 <ENTER>
   ```
5. Follow the on-screen submission instructions

Part-2 (40%)

Instructions

In a new source code file "w5p2.c", upgrade the solution to Part-1 to include data input for a specified number of days that records the user's self-diagnosed "wellness" rating for the morning and evening periods of each day. The application will end with a summary of statistics about the data entered.

1. Review the "Part-2 Output Example" (next section) to see how the program is expected to work
2. Add another macro to define the maximum days (3) of data to collect from the user:

```
#define LOG_DAYS 3
```

   Note: This program must be coded in such a manner that it will work no matter what value is set for LOG_DAYS, from 3 – 28.
3. Continuing from Part-1, use a **for** iteration construct to loop the necessary number times based on the defined LOG_DAYS

   Note: You will need to create additional variables. Be sure to place them at the beginning of the main function so all variables are organized and grouped together and in one place
4. **Nest** inside the **for** construct, the following:
   a) Display the current log date in the format: YYYY-MMM-DD as described in Part-1

      Note:

      The day value must be derived from a variable and not hard-coded, and remember the log always begins on the 1st day of the month
   b) For each day, you need to read two (2) **double** floating-point user input values that represent a **morning** and an **evening** self-diagnosis rating value
   c) Display a prompt to get the user input value for the "**morning**" diagnosis. This is a value that should be between **0.0 and 5.0 inclusive** (refer to the example output)
   d) Validate the rating value entered by the user. An incorrect value that is out of range, should display the appropriate error message and prompt again for a value and repeat as many times as is necessary until a valid value is entered
   e) Repeat the same logic from step: c) above only for the "**evening**" diagnosis.
   f) Repeat from step #4 until the number of desired days is reached
5. After all the data is entered by the user, a summary should be displayed consisting of the following:
   - The sum of all the valid values entered for the **morning** ratings
   - The sum of all the valid values entered for the **evening** ratings
   - The sum of all the valid values entered for the **combined** morning and evening ratings sum of day
     - Note: Display all sums to 3-decimal precision points
   - The average **morning** rating based on the number of **LOG_DAYS** of data entered
   - The average **evening** rating based on the number of **LOG_DAYS** of data entered
   - The average **combined** morning and evening rating based on the number of **LOG_DAYS** of data entered
     - Note: Display all averages to 1-decimal precision point

Part-2 Output Example (Note: Use the YELLOW highlighted user-input data for submission)

```
General Well-being Log
======================
Set the year and month for the well-being log (YYYY MM): 2011 1
   ERROR: The year must be between 2012 and 2022 inclusive
Set the year and month for the well-being log (YYYY MM): 2023 1
   ERROR: The year must be between 2012 and 2022 inclusive
Set the year and month for the well-being log (YYYY MM): 2022 0
   ERROR: Jan.(1) - Dec.(12)
Set the year and month for the well-being log (YYYY MM): 2022 13
   ERROR: Jan.(1) - Dec.(12)
Set the year and month for the well-being log (YYYY MM): 2011 0
   ERROR: The year must be between 2012 and 2022 inclusive
   ERROR: Jan.(1) - Dec.(12)
Set the year and month for the well-being log (YYYY MM): 2023 13
   ERROR: The year must be between 2012 and 2022 inclusive
   ERROR: Jan.(1) - Dec.(12)
Set the year and month for the well-being log (YYYY MM): 2022 2

*** Log date set! ***

2022-FEB-01
   Morning rating (0.0-5.0): -0.8
      ERROR: Rating must be between 0.0 and 5.0 inclusive!
   Morning rating (0.0-5.0): 5.01
      ERROR: Rating must be between 0.0 and 5.0 inclusive!
   Morning rating (0.0-5.0): 4.22
   Evening rating (0.0-5.0): -0.7
      ERROR: Rating must be between 0.0 and 5.0 inclusive!
   Evening rating (0.0-5.0): 5.01
      ERROR: Rating must be between 0.0 and 5.0 inclusive!
   Evening rating (0.0-5.0): 5

2022-FEB-02
   Morning rating (0.0-5.0): 1
   Evening rating (0.0-5.0): 4.6

2022-FEB-03
   Morning rating (0.0-5.0): 4.8
   Evening rating (0.0-5.0): 0

Summary
=======
Morning total rating: 10.020
Evening total rating:  9.600
---------------------------
Overall total rating: 19.620

Average morning rating:  3.3
Average evening rating:  3.2
---------------------------
Average overall rating:  3.3
```

Reflection (50%)

Instructions

Record your answer(s) to the reflection question(s) in the provided "**reflect.txt**" text file

1. Why do we try to use variables and macros to represent information rather than hardcode "magic" numbers (constant literals)? Using the details of this workshop, describe at least three advantages this provides us?

2. The rating values entered by the user required iteration to perform basic validations with each iteration concentrated on testing for values to be within a specific range. Why do we not include additional logic that sums and performs the average? Instead, the sum and average logic is placed outside of any validation routine. Briefly explain two reasons why this is better design and refer to the details of this workshop to justify your answer.

Academic Integrity

It is a violation of academic policy to copy content from the course notes or any other published source (including websites, work from another student, or sharing your work with others).

Failure to adhere to this policy will result in the filing of a violation report to the Academic Integrity Committee.

Part-2 Submission

1. Upload your source file "**w5p2.c**" to your matrix account
2. Upload your reflection file "**reflect.txt**" to your matrix account (to the same directory)
3. Login to matrix in an SSH terminal and change directory to where you placed your workshop source code.
4. Manually compile and run your program to make sure everything works properly:

   ```
   gcc -Wall w5p2.c -o w5 <ENTER>
   ```

   If there are no errors/warnings generated, execute it: ***w5*** *<ENTER>*

5. Run the submission command below (replace **profname.proflastname** with **your professors** Seneca userid and replace **NAA** with your section):

   ```
   ~profName.proflastname/submit 144w5/NAA_p2 <ENTER>
   ```

6. Follow the on-screen submission instructions