Arrays

Workshop 4 (out of 10 marks - 7% of your final grade)

In this workshop, you will code a user-friendly C-language program with an array data structure that processes the elements of the array logically

LEARNING OUTCOMES

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

- to store data of common type using an array structure
- to associate related data using parallel arrays
- to process the elements of an array using an iteration construct
- to describe to your instructor what you have learned in completing this workshop

SUBMISSION POLICY

The "in-lab" section is to be completed during your assigned lab section. It is to be completed and submitted by the end of the workshop period. If you attend the lab period and cannot complete the in-lab portion of the workshop during that period, ask your instructor for permission to complete the in-lab portion after the period. If you do not attend the workshop, you can submit the "in-lab" section along with your "at-home" section (with a penalty; see below). The "at_home" portion of the workshop is **due no later than four (4) days following the in-lab assigned date (even if that day is a holiday) by 11:59PM.**

Your work (all files you create or modify) **must** contain your:

- Full name
- Student number
- Seneca email address
- Section

You are responsible to back up your work regularly.

Late Submission Penalties:

- In-lab portion submitted late, with at-home portion: 0 for in-lab. Maximum of 70/70 for at-home and reflection
- If any of in-lab, at-home or reflection portions is missing the mark will be zero.

IN-LAB: (30%)

Download or clone workshop 4 (**WS04**) from https://github.com/Seneca-144100/IPC-Workshops

Code a program in a file called **temps2.c** that does the following:

1. All temperatures entered by the user must be stored in matching (parallel) arrays.

2. Print the title of the application.

   >---=== IPC Temperature Calculator V2.0 ===---<

3. Prompt the user to enter the number of days for which the temperature will be tracked. The value entered must be between 3 and 10, inclusive.

   Please enter the number of days, between 3 and 10, inclusive:

4. If the user does not enter a value in the correct range, print the following error message:

   Invalid entry, please enter a number between 3 and 10, inclusive:

   Keep doing this until a valid number is input by the user.

5. Using a for loop, prompt the user to enter the high and low temperature until data is entered for the required number of days, store the values entered in matching (parallel) arrays:

   Day 1 – High: (read user input from stdin*)
   Day 1 – Low: (read user input from stdin*)

   *stdin: what the user types in (keyboard)

6. When the process is finished, display the values entered.

Output example:

```
---=== IPC Temperature Calculator V2.0 ===---
Please enter the number of days, between 3 and 10, inclusive: 2

Invalid entry, please enter a number between 3 and 10, inclusive: 4

Day 1 - High: 6
Day 1 - Low: -2
Day 2 - High: 8
Day 2 - Low: -1
Day 3 - High: 7
Day 3 - Low: -3
Day 4 - High: 9
Day 4 - Low: -4

Day  Hi  Low
1    6    -2
2    8    -1
3    7    -3
4    9    -4
```

IN_LAB SUBMISSION:

To test and demonstrate execution of your program use the same data as the output example above, including the erroneous entries (the mistakes).

If not on matrix already, upload your **temps2.c** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account and follow the instructions (replace profname.proflastname with your professors Seneca userid and replace **SAA** with your section):

```
~profname.proflastname/submit 144w4/SAA_lab <ENTER>
```

Please Note

- A successful submission does not guarantee full credit for this workshop.
- If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.

AT_HOME: (30%)

After completing the **in_lab** section, upgrade your code in **temps2.c** to:

- Display the highest temperature, and the day on which it occurred
- Display the lowest temperature, and the day on which it occurred
- Calculate and display the mean (average) temperature for a period entered by the user, until the user enters -1.

Output Example

```
------ IPC Temperature Calculator V2.0 ======
Please enter the number of days, between 3 and 10, inclusive: 11

Invalid entry, please enter a number between 3 and 10, inclusive: 5

Day 1 - High: 6
Day 1 - Low: -2
Day 2 - High: 9
Day 2 - Low: -1
Day 3 - High: 7
Day 3 - Low: -3
Day 4 - High: 8
Day 4 - Low: -9
Day 5 - High: 5
Day 5 - Low: -8

Day  Hi  Low
1    6    -2
2    9    -1
3    7    -3
4    8    -9
5    5    -8

The highest temperature was 9, on day 2
The lowest temperature was -9, on day 4

Enter a number between 1 and 5 to see the average temperature for the entered number of days, enter a negative number to exit: 6

Invalid entry, please enter a number between 1 and 5, inclusive: 7

Invalid entry, please enter a number between 1 and 5, inclusive: 3
```

```
The average temperature up to day 3 is: 2.67

Enter a number between 1 and 5 to see the average temperature for the entered number of days, enter a negative number to exit: 0

Invalid entry, please enter a number between 1 and 5, inclusive: 0

Invalid entry, please enter a number between 1 and 5, inclusive: 1

The average temperature up to day 1 is: 2.00

Enter a number between 1 and 5 to see the average temperature for the entered number of days, enter a negative number to exit: 2

The average temperature up to day 2 is: 3.00

Enter a number between 1 and 5 to see the average temperature for the entered number of days, enter a negative number to exit: 5

The average temperature up to day 5 is: 1.20

Enter a number between 1 and 5 to see the average temperature for the entered number of days, enter a negative number to exit: -8

Goodbye!
```

AT-HOME REFLECTION (40%)

Please provide answers to the following in a text file named **reflect.txt.**

In three or more paragraphs and a **minimum of 150 words**, explain what you learned while doing this workshop. In addition to what you have learned, **your answer should at least include the following**:

- What was good about using arrays - what were the alternatives (how else could you have done it)?
- Comment on the use of parallel (matching) array's. In what ways were they good and/or bad?
- Iterating (looping) through arrays is a common practice; why is it a best practice to initialize iterator variables to a value of zero (0)?

Reflections will be graded based on the published rubric (https://github.com/Seneca-144100/IPC-Workshops/tree/master/WS04/Reflection Rubric.pdf**).**

Example:
An example reflection answer for Workshop #2 is available demonstrating the minimum criteria:
https://github.com/Seneca-144100/IPC-Workshops/tree/master/WS04/Example Reflection-WS_2.pdf

AT-HOME SUBMISSION

To test and demonstrate execution of your program use the same data as the output example above.

If not on matrix already, upload **temps2.c,** and **reflect.txt** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account and follow the instructions (replace profname.proflastname with your professors Seneca userid and replace **SAA** with your section):

```
~profname.proflastname/submit 144w4/SAA_home <ENTER>
```

Please Note

- A successful submission does not guarantee full credit for this workshop.
- If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.