Arrays

Workshop 4 (out of 10 marks - 3% 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 declare an array structure to store data of common type
- to use parallel arrays to associate related data
- to use an iteration construct to process the elements of an array
- 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 lab is due on the day of your next scheduled workshop (23:59).

All your work (all the files you create or modify) must contain your name, Seneca email and student number.

You are responsible to regularly back up your work.

Late submission penalties:

- In-lab submitted late, with at-home: Maximum of 20/50 for in-lab and Maximum of 50/50 for at home

- Workshop late for one week: in-lab, at-home <u>and</u> reflection must all be submitted for maximum of 50 / 100
- Workshop late for more than one week: in-lab, at-home <u>and</u> reflection must all be submitted for maximum of 30 / 100
- If any of in-lab, at-home or reflection is missing the mark will be <u>zero.</u>

IN-LAB: ITEM CLASS (50%)

Download or clone workshop 3 from https://….

Code a program that does the following:

1- All temperatures entered by the user must be stored in matching arrays.

2- Print the title of the application.

> ---=== IPC Temperature Analyzer 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 arrays:

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

*stdin: what the user types in
```

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
```

For submission instructions, see the SUBMISSION section below.

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 **w4_lab.c** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account: (replace profname.proflastname with your professors Seneca userid)

```
~profname.proflastname/submit 144_w4_lab <ENTER>
```

and follow the instructions.

AT_HOME: (40%)

After completing the in_lab section, upgrade your code to:

- 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: 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

Enter a number between 1 and 4 to see the average temperature for the entered number of
days, enter a negative number to exit: 5

Invalid entry, please enter a number between 1 and 4, inclusive: 3

The average temperature up to day 3 is: 2.50
```

```
Enter a number between 1 and 4 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: 2.75

Enter a number between 1 and 4 to see the average temperature for the entered number of
days, enter a negative number to exit: -1

Goodbye!
```

AT-HOME REFLECTION (10%)

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

   1) Why do the arrays in this program have to be declared to hold 10 elements?
   2) What are the advantages and disadvantages of using matching arrays?
   3) Was the at home portion of this workshop too easy, just right or too hard? Why?

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 **w4_home.c** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account: (replace profname.proflastname with your professors Seneca userid)

```
~profname.proflastname/submit 144_w4_home <ENTER>
```

and follow the instructions.

Please note that 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.