# Workshop #4

***Worth: 0.7% 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 to attract a faculty-defined deduction with a minimum of **15%.**
- **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.
- Violating the **Single-Entry-Single-Exit** Principle in your code means **ZERO** for that part.
- **All files** you create must include the statement of authenticity, which is included in the provided files.

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 and execute a C language program that accepts data input from the user, stores the values in variables of the appropriate data type, apply the necessary sequence, selection, and iteration routines to enforce the logic and rules stated in the problem, and produce a working solution that displays the desired output.

## Topic(s)

- Logic

## Learning Outcomes

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

- Code a decision using a selection construct
- Code repetitive logic using an iteration construct
- Nest a logical block within another logical block
- Describe to your instructor what you have learned in completing this workshop

## Part-1 (10%)

### Instructions

Download or clone workshop 4 (**WS04**) 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. Carefully review the "Part-1 Output Example" (next section) to see how this program is expected to work
2. Code your solution to Part-1 in the provided "**w4p1.c**" source code file.
3. Your solution must not declare more than **TWO (2) integer** type variables, nor more than **ONE (1) char** type variable.
4. You will need to receive user input for **TWO (2) values** using a single ***scanf*** function call. The first value is of type char as it represents the desired loop type, and the second value represents the number of times to iterate using that specified iteration construct. Assigning the **single-character** input value can cause unexpected behaviour which you will learn about later in the semester, however, for now use the following ***scanf*** formatting specifier to avoid strange behaviour (notably the **single-space before the percent sign**):

```
scanf(" %c%d", ...
```

5. The program should only acknowledge **uppercase** loop type character inputs: **D**, **W**, **F**, and **Q**. Any other values should **display an error**. You will code the appropriate iteration construct as requested by the user for the number of iterations specified (**D** = **do..while**, **W** = **while**, and **F** = **for**)
6. The **number of requested iterations** should be **between 3 and 20 (inclusive)**, any other values should **display an error**.

Note

The exception to this is when the desire is to **quit**, when **only a 0 value should be accepted**, otherwise the program should **display an error**.

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

```
+----------------------+
Loop application STARTED
+----------------------+

D = do/while | W = while | F = for | Q = quit
Enter loop type and the number of times to iterate (Quit=Q0): R10
ERROR: Invalid entered value(s)!

D = do/while | W = while | F = for | Q = quit
Enter loop type and the number of times to iterate (Quit=Q0): D2
ERROR: The number of iterations must be between 3-20 inclusive!

D = do/while | W = while | F = for | Q = quit
Enter loop type and the number of times to iterate (Quit=Q0): D21
ERROR: The number of iterations must be between 3-20 inclusive!

D = do/while | W = while | F = for | Q = quit
Enter loop type and the number of times to iterate (Quit=Q0): W2
ERROR: The number of iterations must be between 3-20 inclusive!

D = do/while | W = while | F = for | Q = quit
Enter loop type and the number of times to iterate (Quit=Q0): W21
ERROR: The number of iterations must be between 3-20 inclusive!

D = do/while | W = while | F = for | Q = quit
Enter loop type and the number of times to iterate (Quit=Q0): F2
ERROR: The number of iterations must be between 3-20 inclusive!

D = do/while | W = while | F = for | Q = quit
Enter loop type and the number of times to iterate (Quit=Q0): F21
ERROR: The number of iterations must be between 3-20 inclusive!

D = do/while | W = while | F = for | Q = quit
```

```
Enter loop type and the number of times to iterate (Quit=Q0): Q-1
ERROR: To quit, the number of iterations should be 0!

D = do/while | W = while | F = for | Q = quit
Enter loop type and the number of times to iterate (Quit=Q0): Q1
ERROR: To quit, the number of iterations should be 0!

D = do/while | W = while | F = for | Q = quit
Enter loop type and the number of times to iterate (Quit=Q0): D3
DO-WHILE: DDD

D = do/while | W = while | F = for | Q = quit
Enter loop type and the number of times to iterate (Quit=Q0): W3
WHILE   : WWW

D = do/while | W = while | F = for | Q = quit
Enter loop type and the number of times to iterate (Quit=Q0): F3
FOR     : FFF

D = do/while | W = while | F = for | Q = quit
Enter loop type and the number of times to iterate (Quit=Q0): W20
WHILE   : WWWWWWWWWWWWWWWWWWWW

D = do/while | W = while | F = for | Q = quit
Enter loop type and the number of times to iterate (Quit=Q0): F20
FOR     : FFFFFFFFFFFFFFFFFFF

D = do/while | W = while | F = for | Q = quit
Enter loop type and the number of times to iterate (Quit=Q0): D20
DO-WHILE: DDDDDDDDDDDDDDDDDDD

D = do/while | W = while | F = for | Q = quit
Enter loop type and the number of times to iterate (Quit=Q0): d0
ERROR: Invalid entered value(s)!

D = do/while | W = while | F = for | Q = quit
Enter loop type and the number of times to iterate (Quit=Q0): w0
ERROR: Invalid entered value(s)!
```

```
D = do/while | W = while | F = for | Q = quit
Enter loop type and the number of times to iterate (Quit=Q0): f0
ERROR: Invalid entered value(s)!

D = do/while | W = while | F = for | Q = quit
Enter loop type and the number of times to iterate (Quit=Q0): Q0

+--------------------+
Loop application ENDED
+--------------------+
```

---

## Part-1 Submission

1. Upload (file transfer) your source file "**w4p1.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 w4p1.c -o w4** *<ENTER>*
   *If there are no errors/warnings generated, execute it:* ***w4*** *<ENTER>*
4. Run the submission command below (replace **profname.proflastname** with **your professors** Seneca userid and replace **NAA** with your section):
   **~profName.proflastname/submit 144w4/NAA_p1** *<ENTER>*
5. Follow the on-screen submission instructions

---

## Part-2 (40%)

Instructions

1. Review the "Part-2 Output Example" (next section) to see how the program is expected to work
2. Code your solution to Part-2 in the provided "**w4p2.c**" source code file.
3. The first thing that must be done, is to set the number of **apples**, **oranges**, **pears**, **tomatoes**, and **cabbages** you need to shop for. Assigning a **zero (0)** value indicates the item is **not needed**, however a **negative value isn't allowed** and should **display an error message**.

4. The picking stage should prompt for only the items where **positive quantities were assigned**
5. Picking more than the number of items set for a given product should display an error **including the remaining quantity** to be picked.
6. Picking **zero or fewer** is also not accepted and should **display an error message**.
7. When an item has been successfully picked, the next outstanding item on the list should be picked.
8. When all the shopping items have been picked, the process should be repeated for another shopping session until the user inputs a zero (0) value to indicate all shopping is done.

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

```
Grocery Shopping
================
How many APPLES do you need? : -1
ERROR: Value must be 0 or more.
How many APPLES do you need? : 10

How many ORANGES do you need? : -1
ERROR: Value must be 0 or more.
How many ORANGES do you need? : 0

How many PEARS do you need? : -1
ERROR: Value must be 0 or more.
How many PEARS do you need? : 5

How many TOMATOES do you need? : -1
ERROR: Value must be 0 or more.
How many TOMATOES do you need? : 0

How many CABBAGES do you need? : -1
ERROR: Value must be 0 or more.
How many CABBAGES do you need? : 15

-------------------------
Time to pick the products!
-------------------------

Pick some APPLES... how many did you pick? : 40
```

```
You picked too many... only 10 more APPLE(S) are needed.
Pick some APPLES... how many did you pick? : -2
ERROR: You must pick at least 1!
Pick some APPLES... how many did you pick? : 0
ERROR: You must pick at least 1!
Pick some APPLES... how many did you pick? : 5
Looks like we still need some APPLES...
Pick some APPLES... how many did you pick? : 40
You picked too many... only 5 more APPLE(S) are needed.
Pick some APPLES... how many did you pick? : 5
Great, that's the apples done!

Pick some PEARS... how many did you pick? : 40
You picked too many... only 5 more PEAR(S) are needed.
Pick some PEARS... how many did you pick? : -2
ERROR: You must pick at least 1!
Pick some PEARS... how many did you pick? : 0
ERROR: You must pick at least 1!
Pick some PEARS... how many did you pick? : 4
Looks like we still need some PEARS...
Pick some PEARS... how many did you pick? : 40
You picked too many... only 1 more PEAR(S) are needed.
Pick some PEARS... how many did you pick? : 1
Great, that's the pears done!

Pick some CABBAGES... how many did you pick? : 40
You picked too many... only 15 more CABBAGE(S) are needed.
Pick some CABBAGES... how many did you pick? : -2
ERROR: You must pick at least 1!
Pick some CABBAGES... how many did you pick? : 0
ERROR: You must pick at least 1!
Pick some CABBAGES... how many did you pick? : 12
Looks like we still need some CABBAGES...
Pick some CABBAGES... how many did you pick? : 40
You picked too many... only 3 more CABBAGE(S) are needed.
Pick some CABBAGES... how many did you pick? : 3
Great, that's the cabbages done!
```

```
All the items are picked!

Do another shopping? (0=NO): 1

Grocery Shopping
================
How many APPLES do you need? : 0

How many ORANGES do you need? : 20

How many PEARS do you need? : 0

How many TOMATOES do you need? : 15

How many CABBAGES do you need? : 0

--------------------------
Time to pick the products!
--------------------------

Pick some ORANGES... how many did you pick? : 40
You picked too many... only 20 more ORANGE(S) are needed.
Pick some ORANGES... how many did you pick? : -2
ERROR: You must pick at least 1!
Pick some ORANGES... how many did you pick? : 0
ERROR: You must pick at least 1!
Pick some ORANGES... how many did you pick? : 5
Looks like we still need some ORANGES...
Pick some ORANGES... how many did you pick? : 40
You picked too many... only 15 more ORANGE(S) are needed.
Pick some ORANGES... how many did you pick? : 15
Great, that's the oranges done!

Pick some TOMATOES... how many did you pick? : 40
You picked too many... only 15 more TOMATO(ES) are needed.
Pick some TOMATOES... how many did you pick? : -2
ERROR: You must pick at least 1!
Pick some TOMATOES... how many did you pick? : 0
```

```
ERROR: You must pick at least 1!
Pick some TOMATOES... how many did you pick? : 12
Looks like we still need some TOMATOES...
Pick some TOMATOES... how many did you pick? : 40
You picked too many... only 3 more TOMATO(ES) are needed.
Pick some TOMATOES... how many did you pick? : 3
Great, that's the tomatoes done!

All the items are picked!

Do another shopping? (0=NO): 0

Your tasks are done for today - enjoy your free time!
```

---

## Reflection (50%)

Instructions

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

1. After using different types of iteration constructs in this workshop, you learn that you can use any of the iteration constructs do/while, while or for to solve a programming task. Compare the use of the while construct and the do/while construct in getting and evaluating the user's input. Explain.

2. In this workshop, you used "if" optional path constructs and "if/else if" alternative path logic. Give an example from the workshop where using the "if/else if" alternative path construct is more efficient than the "if" optional path.
3. With the completion of this workshop, you must have begun to appreciate why code formatting is important. Explain how you made your code – especially the iteration and selection logic parts – easy to read and maintain.

## Part-2 Submission

1. Upload your source file "**w4p2.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 w4p2.c -o w4 <ENTER>
   ```

   *If there are no errors/warnings generated, execute it:* ***w4*** *<ENTER>*
5. Run the submission command below (replace **profname.proflastname** with **your professors** Seneca userid and replace **NAA** with your section):

   ```
   ~profName.proflastname/submit 144w4/NAA_p2 <ENTER>
   ```
6. Follow the on-screen submission instructions