# CSE240 – Assignment 3
# Input/Output and Compile-Time Arrays

50 points

## Introduction

The aim of this assignment is to make sure you understand Input and Output and basic control structures in C.

By the end of this assignment, you should have:

- Created and used variables
- Used printf and scanf to work with input and output
- Use simple File I/O
- Used control structures to control the flow of logic
- Work with math and random numbers
- Create compile time single dimensional and two dimensional arrays
- Created these things from scratch

## Readings:

Textbook Chapter 2: 2.1, 2.2, 2.3, 2.4

## Programming Assignment

### Instructions:

In this assignment you will create your code from scratch. You are to create a C file named <lastname>_<firstname>_hw3.c

You will put all of your code in this file.

You may use any technique we have learned so far.

### Specifications:

This assignment is split into several parts. The goal is to get you used to working with C/C++ compile-time arrays and strings.

### Part 0 – Create a Menu

You should create a menu that gives access to each of the parts of the assignment.

Show this menu and make it function properly. If the user inputs something incorrect, correct them and loop the menu and prompt.

```
Welcome to Assignment 3!
Menu:
1 - Single Dimension Array Processing
2 - Two-D Processing
Choose an option:
```

## Part 1 – Single Dimension Array

Write a void function to drive Part 1.

### *A – Filling the array)*

Create an array of characters of size 51. This is going to be a fixed size array.

Initialize the array to contain all ‘\0’ characters.

Write a function to fill the array with random lower-case letters.

Prototype example: void fillArray(char letters[], int size);

### *B – Menu)*

Give the user a menu of options to process the array:

1. Check frequency of a letter
2. Remove a letter
3. Sort

### *C – Processing)*

#### Menu Option 1 –

Get a character from the user. Either safety-check it, or force it to be lower-case.

Create a function that returns an integer. This function should take the array, the letter and the size.

Prototype: int getFrequencyOf(char letters[], int size, char item);

You will loop through the array and count each matching instance of the character the user inputted. Return the final count.

Output the result and ask if they want to do another process.

#### Menu Option 2 –

Get a character from the user. Either safety-check it, or force it to be lower-case.

Create a function that will remove ALL instances of the specified character from your array.

Prototype: void removeCharacter(char letters[], int size, char item);

This should not leave gaps in your array. Make sure you ‘shuffle down’ the contents of the array to fill in the gaps. Make sure you properly place the ‘\0’

After processing the array, output the array with a printf(“%s”) statement to prove you processed it correctly.

Ask if they want to do another process.

### Menu Option 3 –

Write a function to sort the array alphabetically.

**Extra Credit opportunity create an O(n log n) algorithm +3 points.**

Prototype: void sortArray(char letters[], int size);

NOTE – Think about when this might be called. Has the user removed any letters?

## Part 2 – 2D Processing

Part 2 will have you doing the single dimensional operations to a 2D array of strings.

Write a void function to drive Part 2.

Part 2 starts by asking the user to input 10 strings of size no larger than 15 characters.

Each of these string will be put in a row of their own in a 2D character array. Remember to end each row with a '\0'.

After you have received the inputs – output the strings to confirm they were inputted correctly.

Give the user a menu of options to process the array:

1. Check frequency of a letter
2. Remove a letter

Apply the operation the user selects to ALL of the strings in the 2D array.

Each time an operation finishes output the results for each row.

NOTE – if you coded part 1 correctly, this section can and will make use of the exact same functions you coded earlier!

## Sample Output:

```
Welcome to Assignment 3!
Menu:
1 – Single Dimension Array Processing
2 – Two-D Processing
Choose an option:
```

### User choose 1:

```
Choose an option: 1

Filling the array!
phqghumeaylnlfdxfircvscxggbwkfnqduxwfnfozvsrtkjpre

What would you like to do?
1.     Check frequency of a letter
2.     Remove a letter
3.     Sort
>>?

User chooses 1:
>>? 1

What letter? g
There are 3 of the letter g
(take the user back to the menu)

User chooses 2:
>>? 2

What letter? g
phqhumeaylnlfdxfircvscxbwkfnqduxwfnfozvsrtkjpre
g has been removed
(take the user back to the menu)

User chooses 3:
>>? 3

Sorting!
abccddeefffffggghhijkkllmnnnoppqqrrrsstuuvvwwxxxyz
(take the user back to the menu)
```

## From Main Menu User chooses 2:

```
Choose an option: 2

Enter string  1: testing
Enter string  2: apple
Enter string  3: document
Enter string  4: computer
Enter string  5: science
Enter string  6: banana
Enter string  7: orange
Enter string  8: sugar
Enter string  9: classes
Enter string 10: fancy

What would you like to do?
1.     Check frequency of a letter
2.     Remove a letter
>>?

User chooses 1:
>>? 1

What letter? a
testing - 0
apple - 1
document - 0
computer - 0
science - 0
banana - 3
orange - 1
sugar - 1
classes - 1
fancy - 1

User chooses 2:
>>? 2
What letter? a
testing
pple
document
computer
science
bnn
ornge
sugr
clsses
fncy
```

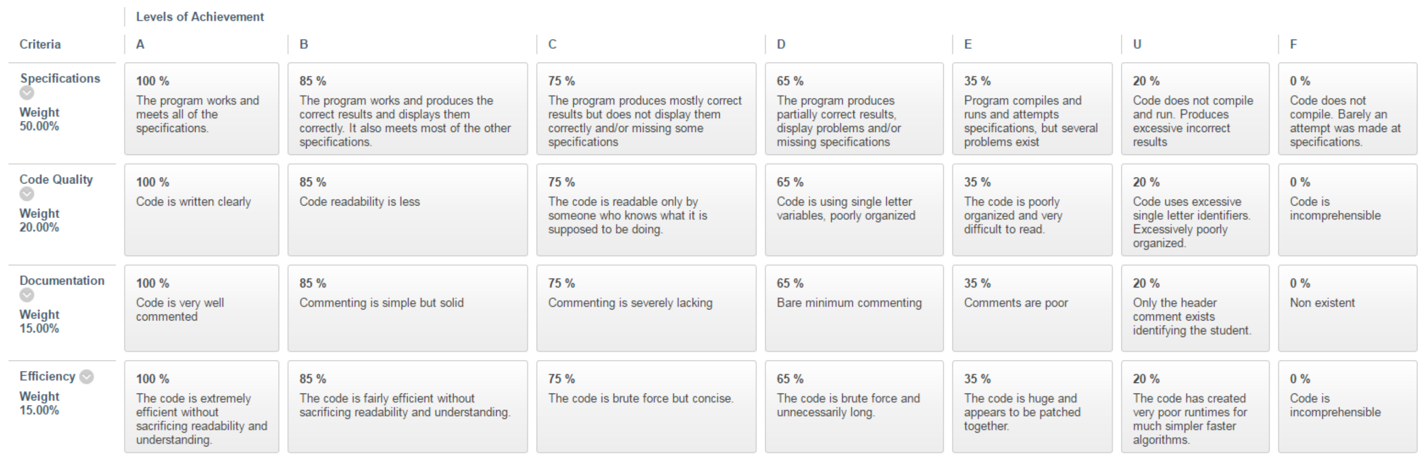
## Grading of Programming Assignment

The TA will grade your program following these steps:

(1) Compile the code. If it does not compile, 20% of the points given will be deducted. For example, if there are 20 points possible, you will earn 16 points if the program fails to compile.

(2) The TA will read your program and give points based on the points allocated to each component, the readability of your code (organization of the code and comments), logic, inclusion of the required functions, and correctness of the implementations of each function.

### Rubric:



| | Levels of Achievement | | | | | | |
|---|---|---|---|---|---|---|---|
| Criteria | A | B | C | D | E | U | F |
| Specifications<br>Weight 50.00% | 100 %<br>The program works and meets all of the specifications. | 85 %<br>The program works and produces the correct results and displays them correctly. It also meets most of the other specifications. | 75 %<br>The program produces mostly correct results but does not display them correctly and/or missing some specifications | 65 %<br>The program produces partially correct results, display problems and/or missing specifications | 35 %<br>Program compiles and runs and attempts specifications, but several problems exist | 20 %<br>Code does not compile and run. Produces excessive incorrect results | 0 %<br>Code does not compile. Barely an attempt was made at specifications. |
| Code Quality<br>Weight 20.00% | 100 %<br>Code is written clearly | 85 %<br>Code readability is less | 75 %<br>The code is readable only by someone who knows what it is supposed to be doing. | 65 %<br>Code is using single letter variables, poorly organized | 35 %<br>The code is poorly organized and very difficult to read. | 20 %<br>Code uses excessive single letter identifiers. Excessively poorly organized. | 0 %<br>Code is incomprehensible |
| Documentation<br>Weight 15.00% | 100 %<br>Code is very well commented | 85 %<br>Commenting is simple but solid | 75 %<br>Commenting is severely lacking | 65 %<br>Bare minimum commenting | 35 %<br>Comments are poor | 20 %<br>Only the header comment exists identifying the student. | 0 %<br>Non existent |
| Efficiency<br>Weight 15.00% | 100 %<br>The code is extremely efficient without sacrificing readability and understanding. | 85 %<br>The code is fairly efficient without sacrificing readability and understanding. | 75 %<br>The code is brute force but concise. | 65 %<br>The code is brute force and unnecessarily long. | 35 %<br>The code is huge and appears to be patched together. | 20 %<br>The code has created very poor runtimes for much simpler faster algorithms. | 0 %<br>Code is incomprehensible |

## What to Submit?

You are required to submit your solutions in a compressed format (.zip). Zip all files into a single zip file. Make sure your compressed file is labeled correctly - lastname_firstname3.zip.

The compressed file MUST contain the following:

    <lastname>_<firstname>_hw3.c

No other files should be in the compressed folder.

If multiple submissions are made, the most recent submission will be graded, even if the assignment is submitted late.

### Where to Submit?

All submissions must be electronically submitted to the respected homework link in the course web page where you downloaded the assignment.

## *Academic Integrity and Honor Code.*

*You are encouraged to cooperate in study group on learning the course materials. However, you may not cooperate on preparing the individual assignments. Anything that you turn in must be your own work: You must write up your own solution with your own understanding. If you use an idea that is found in a book or from other sources, or that was developed by someone else or jointly with some group, make sure you acknowledge the source and/or the names of the persons in the write-up for each problem. When you help your peers, you should never show your work to them. All assignment questions must be asked in the course discussion board. Asking assignment questions or making your assignment available in the public websites before the assignment due will be considered cheating.*

*The instructor and the TA will* ***CAREFULLY*** *check any possible proliferation or plagiarism. We will use the document/program comparison tools like MOSS (Measure Of Software Similarity: http://moss.stanford.edu/) to check any assignment that you submitted for grading. The Ira A. Fulton Schools of Engineering expect all students to adhere to ASU's policy on Academic Dishonesty. These policies can be found in the Code of Student Conduct:*

*http://www.asu.edu/studentaffairs/studentlife/judicial/academic_integrity.htm*

*ALL cases of cheating or plagiarism will be handed to the Dean's office. Penalties include a failing grade in the class, a note on your official transcript that shows you were punished for cheating, suspension, expulsion and revocation of already awarded degrees.*