# EECE 1080/CS1 - Summer 2019 – Laboratory 6

***The Tesla, The Yugo, Elon Musk, and the bar chart***

***Topics covered: Arrays, strings, and Functions***

**Due**:

Sunday June 30, 2019 by 11:59PM on Blackboard

**Objective**:

The objective of this lab is to gain experience creating and using classes, and 2-dimensional C style arrays. Also, it will reinforce the concepts of loop structures, functions, and text formatting.

Please read this over carefully before beginning to work on this laboratory. It is strongly suggested that you create an outline (possibly pseudo code or even a flowchart) on solving these problems before attempting to code the solution.

**Collaboration:**

As with most laboratory assignments in this course this laboratory assignment is to performed by an individual student. You can help each other learn by reviewing assignment materials, describing to each other how you are approaching the problem, and helping each other with syntax errors. You can get help from tutors, teaching assistants, and instructors. Having similar code for some assignments is expected but most assignments have multiple different solutions. Your code is expected to be different.

Please document any help you receive from other students, teaching assistants, or instructors. Just add the names to the top of your source file.

Having someone else code for you, sharing code with other students, or copy-pasting code from the internet or previous terms , is cheating. A helper should "teach you to fish, not feed you the fish". Laboratory assignments prepare you for exams so be smart.

**Specification**:

**Task 1:** Review material used in assignment

Review strings, functions, arrays, and read the problem specification before continuing.

**Task 2:** Program Development

To receive full credit for part A, please demonstrate this section to a TA or your instructor. Note: You will to still receive credit for attending the laboratory session.

# EECE 1080/CS1 - Summer 2019 – Laboratory 6

*The Tesla, The Yugo, Elon Musk, and the bar chart*

*Topics covered: Arrays, strings, and Functions*

## Part A: Array Statistics and Random Numbers

Filename: array_stats.cpp

The program will generate a series of random numbers that will all fall within a user specified upper limit and zero. The program will count the number of times each number occurs in the randomized input data.

**You will NOT need to store the randomized input data or perform any analysis on this data.**

You will want to create an array to store these counts. ***Remember each random input value can be used as an index to your array so you can update the count directly. For example,***

```
int randomsample = randnumber(); // random number 0 to 100
v[randomsample] += 1;
```

***would update the count for the number associated with randomsample.***

The user will enter from the keyboard the number of random numbers to be generated (at least 10000 samples).

The user will also be able to specify the upper limit of these random values. Please allow a maximum upper limit of 100 for these random values.

Please create functions to complete the bulleted list below. Each bulleted item could become a function. **You need at least 5 to 7 functions. The more functions the better.** You are free to create these functions with any name and number of parameters. All statistics are in terms of the frequency array and not the data samples.

- Generate a random number based on a specified start and end number. This function should be able to generate random integer numbers for any range up to RAND_MAX

- Create a frequency count array with counts of random numbers from a range of values from 0 to the user specified limit (no more than 100). You will need to generate the user supplied sample count number of values.
  - For a range of 0 to 100 you will have to use an array of a size equal to 101.
  - You are required to call the random number function above.
  - ***No need to save or store the random samples.***

- Print array statistics including sum, mean, max, and min for the frequency array.

  - You are strongly encouraged to write functions for sum, mean, max, and min. They are all very simple functions.

- Print the array in a formatted manner. You can (if you want to) combine this with the barchart display below.

- Display a horizontal barchart scaled to fit the page. If the count goes over 60 to 70 (the approximate width of the terminal) then you should scale it to fit (make one '*' represent two occurrences, five occurrences, etc. instead of just the one of each number in the series). You should use an asterisk '*' to represent the "bar" portion of the bar graph.

    - To scale the barchart you will need the maximum value stored in the frequency array and the screen size.

      The scaling factor equals (max_value/max_bar_size).

      Max_bar_size is usually 60 or lower.

      For sample below, the scale equals 956/60 = 16 stars per

      The ciel() function might be helpful.

  See Sample Output below:

```
Enter End Range:
10
Enter number of Samples:
10000

Index     Value Bar
    0       907 *********************************************************
    1       911 *********************************************************
    2       867 *******************************************************
    3       915 **********************************************************
    4       947 ************************************************************
    5       873 *******************************************************
    6       896 ********************************************************
    7       925 **********************************************************
    8       956 ************************************************************
    9       890 ********************************************************
   10       913 **********************************************************

Scale: 16 per *

Range: 0 to 10
Sample Count: 10000
Min Value: 867
Max Value: 956
Sum Value: 10000
Mean Value: 909.091
```

Remember functions can call other functions!

**EECE 1080/CS1 - Summer 2019 – Laboratory 6**

***The Tesla, The Yugo, Elon Musk, and the bar chart***

***Topics covered: Arrays, strings, and Functions***

**Part B:** Saving Elon Musk's Roadster from going into space

filename: tesla.cpp.

In this task, you will be developing a game to allow the user to “save” Elon Musk’s prized Roadster from being sent into space.

*The Story*:

You find your way to SpaceX the night before the Roadster is mounted atop the rocket which is to send it into space.  The building is dark and the roadster is hidden somewhere in large open dark room (15x15). In addition to the roadster, two other cars (a Yugo and ford pinto) are being stored in this facility as well.

BTW: Elon Musk is sleeping somewhere in the same room and you have to avoid waking him to find your way to the roadster. Once awoken he will randomly move through the building looking for you. If he lands next to you, you can get away from him only once. The second time you will be the “Starman” and blasted into space with his roadster. Once he is a awake your only chance to avoid being “Starman” is to make your escape in his roadster. If you get into either of the other two cars you will be caught and become the “Starman”.

*Game Play:*

The goal for the user is to find the roadster before becoming the “Star man”.

As a programmer you are free to use any method to display game information including (but not limited to) a descriptive menu, a board graphic, a combination of both, etc. If you are showing the room to the user graphically, do not show the location of the cars, or elon musk unless they are in the immediate vicinity (within the 3 by 3 grid area with the user at the center).

Note: You will need to randomly generate the locations of the sleeping elon musk, the yugo, the pinto, and the roadster. You will probably want to store these directly in your room as special symbols if you are displaying the board to the user.

- The user should start in a random unoccupied location.

- The user should be able to enter w (forward/north), a (left/west), s (backward/south), d (right/east) for movement directions.
- The user can quit at anytime.
- Your goal is to guide the user to the proper location of the roadster by issuing hints. An example, of a hint might be using the phrases “you are getting warmer” or “you are getting colder” or “I think you should turn around you are going the wrong way”, etc.. Try to avoid telling the user to go a specific direction.
- You can decide how Elon Musk wakes up.
- Once the game is over you will ask the user to play again and if so you will generate a new random location for the game elements and go through the same process as before.
- You will need to track statistics about the number of turns it took to find the roadster, the number of times the roadster was found, and the number of times the user become “Starman”.

**<u>Task 3:</u>** Submit to blackboard

Submit your completed project to blackboard

- ***array_stats.cpp***
- ***tesla.cpp***

by the due date.

**<u>Assignment Grading Rubric:</u>**

1. Part A: 40-Points
2. Part B: 60-Percent

All source code needs to compile. Source code should be properly styled and commented as well.