The objective of this lab is to:

1. Revise basic and some advance concepts of programming fundamentals.
2. Reinforce the concepts of functions, pointers, dynamic memory, linear arrays and 2-D arrays with a focus on problem-solving.
3. Practice good coding conventions e.g commenting, meaningful variable and function names, properly indented and modular code.

Instructions!

1. Please follow dress code before coming to the lab. Keep your student identity cards with you.
2. This is an individual lab, you are strictly **NOT** allowed to discuss your solutions with your fellow colleagues, even not allowed asking how is he/she is doing, it may result in negative marking. You can **ONLY** discuss with your TAs or with me.
4. Strictly follow good coding conventions (commenting, meaningful variable and functions names, properly indented and modular code.
5. Save your work frequently. Make a habit of pressing **CTRL+S** after every line of code you write.
6. Beware of **memory leaks** and **dangling pointers**.

Task 00: [10 Minutes]

Read and understand the README.pdf document provided as part of this lab.

Task 01: [10 Marks]

1. Write a program that should create an array of user-defined size and then expand it. To expand the array, implement a function *resizeArray* that will take an array and its size as input. It should return a new array of double size, the first half of the array should contain the elements of the original array, and the second half should contain two times the values of the input array. Your function should have the following prototype:

```
int* resizeArray (int inArray[], const int size);
```

Sample Execution:

Enter size of the array: **4**
Enter elements of array: **1 2 3 4**

Resized array: 1, 2, 3, 4, 2, 4, 6, 8

Task 02: [5 Marks]

Write and program that should create a square matrix of user-defined size, populate the matrix with user given values, and determine whether the matrix is a diagonal matrix or not. Where, a diagonal matrix is a matrix in which the entries outside the main diagonal are all zero. The diagonal entries themselves may or may not be zero.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & -2 \end{bmatrix}$$

Task 03: [10 Marks]

1. Write a program that declares a char array of size 30 then takes a string from the user and stores it in a character array. Display a substring of *count* letters from left, where the count is given by the user. Print appropriate message if the user enters out of range count.

Sample Execution:

Enter a string: **Equal matrices have equal dimensions**

Enter count: **5**
Left substring: Equal

Implement the following two functions for this task:

1. `char* left(const char* target, const int count );`
   This function should return a string that contains a copy of the specified range of characters. Where,
   target
   The target string from which this function will extract characters.
   count
   The number of characters to extract from the *'target'* string from left side.

2. `void Destructor(char *p);`
   This function should De-allocate the dynamically allocated memory pointed by *'p'*.

Task 04: [15 Marks]

Recently we have hired a new TA, he doesn't like manual record keeping and calculating statistics by hand. You can help him by making a small program that he can be used to store the grades of all students enrolled in OOP class. The program should perform the following tasks:

a) Ask the number of students enrolled in our course. An array of integers with that many elements should then be dynamically allocated.
b) Populate the array by taking the marks of each student of the class.
c) Calculate and display the maximum, minimum, average, and median of the marks entered.

You should implement the following functions for this task:

a. `int findMin (int inArray[], const int size);`
b. `int findMax (int inArray[], const int size);`
c. `int computeAverage (int inArray[], const int size);`
d. `int computeMedian (int inArray[], const int size); // when a set of values is sorted in ascending or descending order, its median is the middle value. If the set contains an even number of values, the median is the average of the two middle values.`

Use appropriate prompts for input and output to make your program easy to use. Furthermore, make sure zero is the minimum score one can get (i.e. do not accept negative numbers for input).