# Lab # 1
## Object Oriented Programming
### BS SE F17 Afternoon

**Instructions:**

• **Attempt the following tasks exactly in the given order.**

• Indent your code properly.

• Use meaningful variable and function names. Follow the naming conventions.

• Use meaningful prompt lines and labels for all input/output.

• Make sure that there are **NO dangling pointers** or **memory leaks** in your program.

**Task 1**

Write a program that takes two strings (character arrays) from user and concatenates (combines) them. Size of strings can be fixed. For this you are required to define following functions.

Concatenate(char * string1, char* string2);

In main(), declare 2 character arrays with size not more than 10, take input in them and place a null character at the end of both arrays. Pass them to function, declare a third array with size 20 to hold concatenated data and display it in function.

**Example:**
**Enter first string:** abcde
**Enter second string:** fghij
**Output:** abcdefghij

**Task 2**

Implement a function **specialSearch** which takes a one-dimensional array of integers, its size, and an integer *key* as arguments. This function will determine the number of elements less than, and number of elements greater than *key* in the given array. The prototype of your function
should be:

**void specialSearch (int *arr, int n, int key,**
**int& numLess, int& numGreater)**

In the above function prototype: **arr** is an array which contains **n** integers in *unsorted* order, **key** is the value based upon which the searching will be performed, **numLess** and **numGreater** are reference parameters which will be used to return the counts of the number of elements less/greater than the **key**.

**Task 3**

Write a function that accepts an int array and the array's size as arguments. The function should create a new array that is twice the size of the argument array. The function should copy the contents of the argument array to the new array, and initialize the unused elements of the second array with 0. The function should return a pointer to the new array.

## Task 4

**2D Array Operations**

Write a program that creates a two-dimensional array initialized with test data. Use any data type you wish. The program should have the following functions:

• **getTotal**. This function should accept a two-dimensional array as its argument and return the total of all the values in the array.

• **getAverage**. This function should accept a two-dimensional array as its argument and return the average of all the values in the array.

• **getRowTotal**. This function should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The function should return the total of the values in the specified row.

• **getColumnTotal**. This function should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a column in the array. The function should return the total of the values in the specified column.

• **getHighestInRow**. This function should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The function should return the highest value in the specified row of the array.

• **getLowestInRow**. This function should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The function should return the lowest value in the specified row of the array.

Demonstrate each of the functions in this program.