Lab Assignment #1

Due Date: **On or before Second Class of Week 3** **Marks/Weightage: 30/10%**

Purpose: The purpose of this Lab assignment is to:

- Practice the use of delegates, lambdas, and event handling and observer pattern.

References: Read the lecture notes/ppts and code examples. This material provides the necessary information that you need to complete the exercises.

Instructions: Be sure to read the following general instructions carefully:

This lab should be completed individually by all the students. You will have to demonstrate your solution in a scheduled lab session and submitting the assignment **through drop box link on e-Centennial**.

>> At the start, you must name your **Visual Studio 2019 solution name** according to the following rule:

FirstName-LastName_SectionNumber_COMP212_Labnumber

For Example: ***John-Smith_Sec003_COMP212_Lab01 (say if your section number is 003)***

>> And after that your **project name** should be as follows:

FirstName-LastName_SectionNumber _Labnumber

For Example: ***John-Smith_Sec003_Lab01***

>>Each exercise should be placed in a separate package named as firstname_last-name_*exercise1*, firstname_last-name_*exercise2* etc.

>> After you complete, exit eclipse and go to workspace folder, zip it up and you will get the following zip file.

FirstName_LastName_SectionNumber_COMP212_Labnumber.zip

Example: **John_Smith_Sec003_COMP212_Lab01.zip** *(if your section is 003..)*

>> Apply the naming conventions for variables, methods, classes, and packages:

- *variable names* start with a *lowercase* character for the first word and uppercase for every other word
- *classes* start with an *uppercase* character of every word
- **namespace** use only *lowercase* characters
- *methods* start with a *uppercase* character for the first word and uppercase for every other word

Note: Late submissions are accepted until up to three days past due date with 25% deductions. After that no submission will be considered.

Exercise 01:

Create a C# app in which you are required to create the following methods using built-in delegates predicate and lambdas.

- string **Minimum**(string1, string2, string3) which returns the smallest of three string values. To test this method, you need to use built-in **Func**<> delegate predicate
- void **AvgGrade**(value1, value2, value3) which displays the average of three grades. To test this method, you need to use built-in **Action**<> delegate predicate

Exercise 02:

Define a student's grade array having at least 10 double values between 10 and 100. Define a delegate – GradesPredicate which takes one input of type double and returns a bool.

Define a method – GradesFilter which displays only those grades values which are greater than or equal to 50. This method should take an array of type doubles and second argument is a variable of GradesPredicate. This method should not return any type.
We should call this method like this:

GradesFilter(gradesArray, lambda-Expression);

Lambda-Expression should filter values which are greater than or equal to 50.