LAB ASSIGNMENT #5 – FILE IO

Due Date: **Week 12** **Marks/Weightage: 20/10%**

Purpose: The purpose of this Lab Assignment is to:

- Practice the use of FILE IO

References: Read the course's text book **chapter 14 FILE IO** and the lecture notes/ppts. 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 project **through drop box link on e-Centennial**.

You must name your solution according to the following rule:
FirstName-LastName_SectionNumber_COMP123_Labnumber
For Example: **Joh-Smith_Sec001_COMP123_Lab05**

Each exercise should be placed in a separate namespace named firstname-last-name_*exercise1*, firstname-last-name_*exercise2* etc.

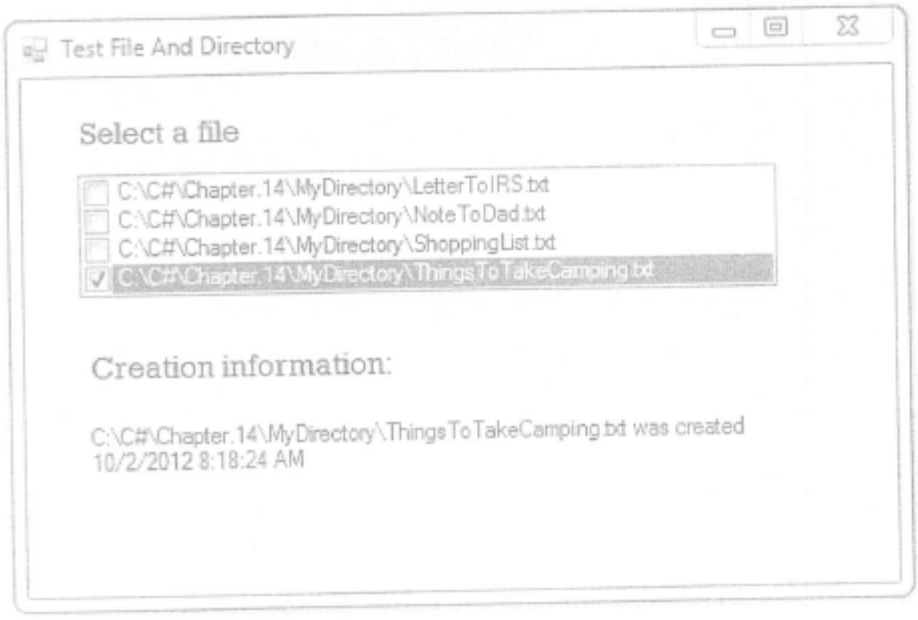
Submit your assignment in a **zip file** that is named according to the following rule:
FirstName-LastName_SectionNumber_COMP123_Labnumber.zip
Example: **Joh-Smith_Sec001_COMP123_Lab05.zip** *(if your section is 001..)*

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
- namespaces use only *lowercase* characters
- *methods* start with a *uppercase* character for the first word and uppercase for every other word

Exercise #1:

[10 marks]

Using Visual Studio, create a `Form` like the one shown in Figure 14-38. Specify a directory on your system, and when the `Form` loads, list the files the directory contains in a `CheckedListBox`. (You first saw an example of a `CheckedListBox` in Chapter 12.) Allow the user to click a file's corresponding check box and display the file's creation date and time. (Each time the user checks a new filename, display its creation date in place of the original selection.) Save the project as **TestFileAndDirectory2**. Create as many files as necessary to test your program.



Exercise #2:

[10 marks]

4. a. Create a program named **WriteInventoryRecords** that allows you to enter data for items you sell at an online auction site and saves the data to a file. Create an `Inventory` class that contains fields for item number, description, and asking price.

   b. Create a program named **ReadInventoryRecords** that reads the file created in Exercise 4a and displays each item's data on the screen.

   c. Create a program named **FindInventoryRecords** that prompts the user for an item number, reads the file created in Exercise 4a, and displays data for the specified record.

   d. Create a program named **FindInventoryRecords2** that prompts the user for a minimum selling price, reads the file created in Exercise 4a, and displays all the records containing a price greater than or equal to the entered price.

a. Create a program named **CustomizeAForm** that includes a `Form` for which a user can select options for the background color, size, and title. The `Form` should look like the one shown in Figure 14-39. Change each feature of the `Form` as the user makes selections. After the user clicks the "Save form settings" `Button`, save the color, size, and title as strings to a file and disable the button.

Evaluation:

Functionality	
Correct implementation of classes (instance variable declarations, constructors, getter and setter methods etc.)	70%
Correct implementation of driver classes (declaring and creating objects, calling their methods, interacting with user, displaying results)	20%
Comments, correct naming of variables, methods, classes, etc.	5%
Friendly input/output	5%
Total	100%