CSE446/598 Project 4 (100 Points)

Spring 2019

Assignment 7 Due: April 6, 2019, by 11:59pm (Arizona Time), plus a one-day grace period

Assignment 8 Due: April 13, 2019, by 11:59pm (Arizona Time), plus a one-day grace period

---

Introduction

The aim of this assignment is to make sure that you have read the slides and the Text Chapter 10 and have understood the concepts covered in the lectures and in the text, including Ado .Net that interfaces SOA software with relational databases, XML databases, and LINQ that deals with different types of databases. By the end of the assignment, you should have a good understanding of the concepts and have applied these concepts in developing operational programs.

Practice Exercises (No submission required)

1. Reading: Textbook Chapter 10.

2. Answer the multiple choice questions 1.1 through 1.15 of the text section 10.6. Study the material covered in these questions can help you to prepare for the class exercises, quizzes, and the exam.

3. Study for the questions 2 through 8 in text section 10.6. Make sure that you understand these questions and can briefly answer these questions. Studying the material covered in these questions can help you to prepare for the exam and understand the homework assignment.

4. Follow the tutorial in text section 10.3.4 to create a LINQ to SQL database, and use LINQ to query the database. Depending on the database that you have installed on your computer and the version of the Visual Studio that you use, the tutorial need to be adjusted based on the features of your software.

5. Read and test the following code (Source: https://social.msdn.microsoft.com/). The program processes a .csv (Comma-Separated Values) file, which can be read as a text file. This example can help you in doing your assignment question.

```
int entriesFound = 0;
using (var textReader = new StreamReader("fileName")) {
    string line = textReader.ReadLine();
    int skipCount = 0;
    while (line != null && skipCount < 1) {
        line = textReader.ReadLine();
        skipCount++;
    }
    while (line != null) {
        string[] columns = line.Split(_Delimiter);
        //perform your logic
        entriesFound++;
        line = textReader.ReadLine();
```

```
    }
}
```

Explain what this piece of code does.

Assignment 7 LINQ to Objects Query (Submission Required, 50 points)

Part 1 Due: April 6, 2019, by 11:59pm (Arizona Time)

1. As an example, you are given an Excel file Courses.csv in the "comma separated values" format, which is copied from the ASU course list. The list is modified for the assignment. Do not use the list for schedule you class. You can must add at least 10 classes into the list so that the data are sufficient to show all the features that are required. You will write a program to create an in-memory array from the data source and then query the data source using LINQ to Objects classes. Use comments to indicate what part of code answers which question. You can use a console application or other .Net application template to implement this assignment. You can put the Courses.csv into an App_Data folder in your application.

1.1 Define a Course class consisting of at least the following members: CourseId, Subject, CourseCode, Title, Location, and Instructor. Write a program to read the Courses.csv file and create an in-memory **array** of objects of the Course class. [10]
Hint: Start from the program given in Question 5 in Practice Exercises part.

1.2 Create the following LINQ queries to extract data from your in-memory data source.

a. Retrieve the list of IEE courses with a course number of 300 or higher. Deliver the result set in the type of IEnumerable<T>, where each entity contains course title and course instructor. The result set must be sorted by instructor and in ascending order. Print the results. [5]

b. Retrieve and deliver the courses in groups and subgroups (in two levels): Use the course subject, (e.g., CPI, CSE, and IEE), as the first level key, and use the course code (e.g., 240, 310, and 494) as the second level key. Print the groups that have at least two courses in the second level group. Hint: Read the example in text 10.3.3 and lecture slides 4-2. [10]

1.3 Create a new file called Instructors.csv. The schema of the file is: instructor name, office number, and email address. Choose at least 30 instructors from Courses.csv list and add them into your Instructors file. You can find the information fromCIDSE faculty directory at : https://cidse.engineering.asu.edu/facultyandresearch/directory/faculty/
You can make up an office number and the email address if you cannot find the information from the site. Each email address must be unique. You can create the file manually using Excel. [5]

1.4 Define the Instructor class consisting of the following members: InstructorName, OfficeNumber, and EmailAddress. Write a program to read the Instructors.csv file and create an in-memory **list** of the objects of the Instructor class. [10]

1.5 Use the Courses array and Instructors list as two data sources. Write a query to find the course instructor's email address for each course. Deliver the result set in the type **var**, where each entity

contains Subject CourseCode (e.g., CSE 310) and the course's instructor email address. The result set must be sorted by CourseCode in ascending order. Print the results for all 200 level courses. [10]

Assignment 8 LINQ Application (Submission Required, 50 points)

Part 2 Due: April 13, 2019, by 11:59pm (Arizona Time)

2. For the same Excel file Courses.csv file, in this part of the assignment, you will create an XML document from the data source and query the XML data source using LINQ to XML classes. Use comments to indicate what part of code answers which question. You can use a console application or other .Net application template to implement this assignment. You can put the Courses.csv into the App_Data folder in your application. [50 points]

2.1 Define the Course class consisting of at least the following members: CourseId, Subject, CourseCode, Location, and Instructor. Write a program to read the Courses.csv file and create an in-memory **XML** file of the objects of the Course class. [10]

2.2 Save the XML document into the App_Data folder in your application. [10]

2.3 Create the following LINQ queries to extract data from your in-memory XML source.

a. Retrieve the list of CPI courses with a course number of 200 or higher. Deliver the result set in the type of IEnumerable<XElement>, where each entity contains course title and course instructor. The result set must be sorted by instructor in ascending order. Print the results. [5]

b. Retrieve and deliver the courses in groups in two levels: Use the course subject, (e.g., CPI, CSE, and IEE), as the first level key, and use the course code (e.g., 240, 310, and 494) as the second level key. Print the groups that have at least two courses in the second level group. [10]

2.4 Use the Course XML document and Instructors list (created Part 1) as data sources. Write a query to find each course and the course instructor's email address. Deliver the result set in the type IEnumerable<XElement>, where each entity contains Subject and CourseCode (e.g., CSE 310) and course instructor's email address. The result set must be sorted by CourseCode in ascending order. Print the results for all 200 level courses. [15]

Grading

We will grade your programs following these steps:

(1) We will read your program and comments to your code. We give points based on the points allocated to each component, the readability of your code (organization of the code and comments), logic, inclusion of the required functions, and correctness of the implementations of each function.

(2) Compile the code. If it does not compile, 40% of the points given in (1) will be deducted. For example, if you are given 20 points in step (1), your points will become 12 if the program fails to compile.

(3) If the code passes the compilation, we will execute and test the code. If, for any reason, the program gives an incorrect output or crashes for any input, 20% of the points given in (1) will be deducted.

Please notice that we will not debug your program to figure out how big or how small the error is. You may lose 40% or 20% of your points for a small error such missing a comma or a space!

General Submission Requirement

All submissions must be zipped into a single file electronically submitted to the blackboard assignment folder. All files must be zipped into a single file for submission.

For programming assignments, the entire solution folder with all the files must be included. To make sure that you have all the files in the zip file, unzip the file on a different machine or in a different directory, and test if you can run the project in a different location.

Submission notice: A programming assignment typically consists of multiple distributed parts. They may be stored in different locations when you create them. You must copy these projects into a single folder for blackboard submission. To make sure that you have all the files included in the zip file and they work together, download your own submission from the blackboard. Unzip the file on a different machine, and test it and see if you can run the solution in a different location because the TA will test your application on a different machine.

Late submission deduction policy

For each part of the submission

- No penalty for late submissions that are received within 24 hours (grace period) of the given deadline;
- 1% grade deduction for every hour after the grace period!
- No submission will be accepted 48 hours after the grace period.