**ASP.NET Core Workshop**

**LINQ**

Learning Outcome

To practice LINQ with C# Collections and Models

Exercise 1

You are given a list of students.

List<Student> students = new List<Student>() {

new Student() { Name = "John", Age = 13 } ,

new Student() { Name = "Joey", Age = 21 } ,

new Student() { Name = "Bill", Age = 18 } ,

new Student() { Name = "Alex" , Age = 20} ,

new Student() { Name = "Ron" , Age = 15 }

};

Write a single LINQ statement to get all the names of students with age between 12 and 20.

Exercise 2

You are given an array

int[] numbers = { 20, 12, 6, 10, 0, 3, 1 };

Write a single LINQ statement sort the array ascendingly.

Exercise 3

You are given a list of students.

List<Student> students = new List<Student>() {

new Student() { Name = "John", ExamMark1 = 75, ExamMark2 = 80 } ,

new Student() { Name = "Joey", ExamMark1 = 95, ExamMark2 = 50 } ,

new Student() { Name = "Bill", ExamMark1 = 40, ExamMark2 = 50 } ,

new Student() { Name = "Alex" , ExamMark1 = 20, ExamMark2 = 75 } ,

new Student() { Name = "Ron" , ExamMark1 = 70, ExamMark2 = 80 }

};

1. Use a LINQ statement to retrieve the names of students, sorted by their ExamMark1 in descending order.
2. Use a LINQ statement to retrieve the names of students, sorted by their average exam mark in descending order.
3. Modify the LINQ statement in b. to also include the respective average score for each student name.

Exercise 4

You are given two integer arrays.

int[] numbers1 = { 0, 1, 2, 3, 4 };

int[] numbers2 = { 5, 6, 7, 8, 9 };

Write a single LINQ statement that generate pairs such that the multiplication of the two elements in a pair is greater than 20. The first element of a pair must come from numbers1, while the second from numbers2. Your results should look like this:

3,7

3,8

3,9

4,6

4,7

4,8

4,9

Exercise 5 (\*)

You are given a list of students, for example

List<Student> students = new List<Student>() {

new Student() { Name = "John", ExamMark1 = 75, ExamMark2 = 80 } ,

new Student() { Name = "Joey", ExamMark1 = 95, ExamMark2 = 50 } ,

new Student() { Name = "Bill", ExamMark1 = 40, ExamMark2 = 50 } ,

new Student() { Name = "Alex" , ExamMark1 = 20, ExamMark2 = 75 } ,

new Student() { Name = "Ron" , ExamMark1 = 70, ExamMark2 = 80 }

};

Write a C# program using LINQ Order and Group to display the category for students’ average marks (0x, 1x, 2x, 3x…) and the respective students in each category.

The result for the sample above input should be:

(7x): 3 students

John, average 77.5

Ron, average 75

Joey, average 72.5

(4x): 2 students

Alex, average 47.5

Bill, average 45

In the sample output above, John’s average mark is 77.5, Joey 72.5 and Ron 75, so they are put into category 7x. In the same manner, Alex’s average is 45 and Bill 45, so they are put into category 4x.

For each group, they are also displayed in descending order, based on their average marks.

Hint: you may need to make some changes to the class Student.