Purpose: Practice the use of LINQ, Asynchronous Programming, Functional Programming (Filter Map Reduce) and PLINQ

Exercise 01: Based on LINQ extension methods. *[10 marks]*

Create an Invoice class which includes four properties – a PartNumber (type int), a PartDescription (type string), a Quantity of item being purchased (type int) and a Price(type decimal).

Use the following sample data for Invoice class objects:

Part Number	Part Description	Quantity	Price
87	Electric Sander	7	57.98
24	Power Saw	18	99.99
7	Sledge Hammer	11	21.50
77	Hammer	76	11.99
39	Lawn Mower	3	79.50
68	Screw Driver	106	6.99
56	Jig saw	21	11.00

Perform the following queries on the array of Invoice objects and display the results:

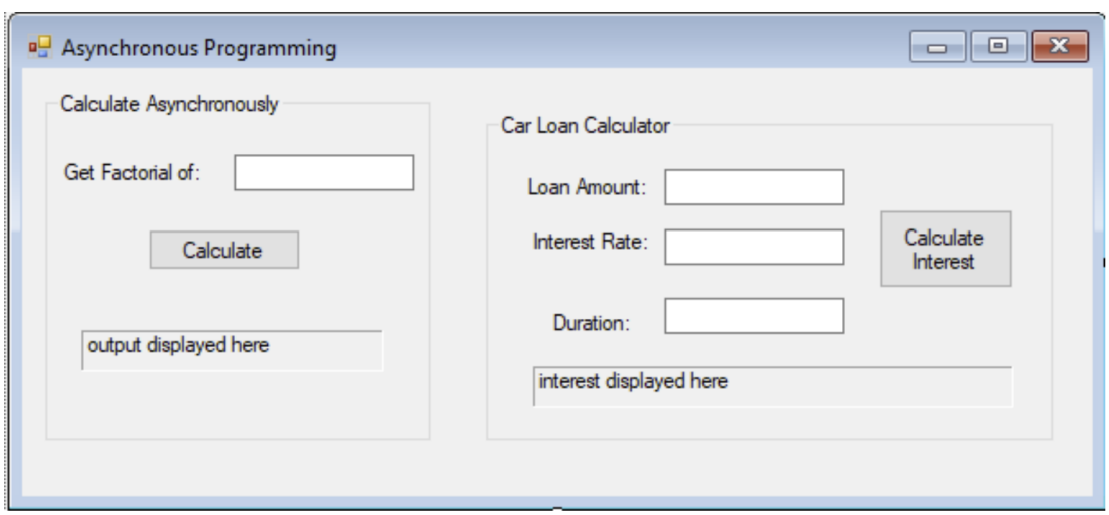
a) Use LINQ to select from each Invoice the PartDescription and value of the Invoice (i.e. Quantity * Price). Name the calculated column as InvoiceTotal. Order the results by invoice value in ascending order. *[Hint: use let]*

b) Part description of the part who has highest quantity.

c) Average price of the parts.

Exercise 02: Build the following app using Win Forms. *[5 marks]*

Factorial implementation should be asynchronous (use of async and await; and Task.Run() method) . Car Loan Calculator is synchronous implementation.



To calculate the factorial of bigger numbers, say 100, long int type will not be sufficient, you need to use **BigInteger**. Research on that and it is strongly recommended to use it.

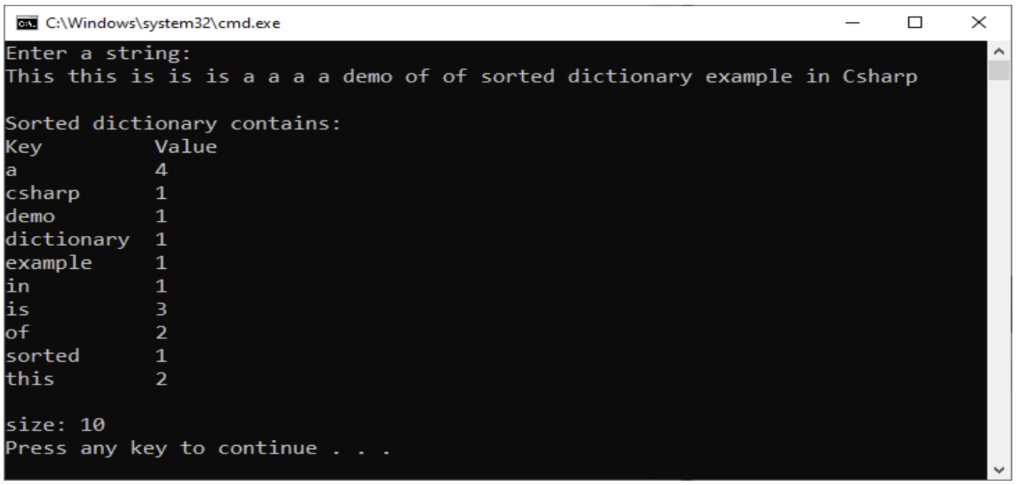
Exercise 03: This is based on Functional programming with LINQ extension methods and lambdas. *[5 marks]*

Create a console app (.Net Framework) and do the following:

Create a list (use built-in list<> class. And add 10 values (between 1 and 100, use random function) to it. Display the values in the list. After that find all the values > 50 and add 10 to each values, sort them and display them. (refer the code example – FilterMapReduce)

Exercise 04: Create a console app (.Net Framework) and do the following: *[5 marks]*

This is based on the Sorted Dictionary Example (covered in the class).



You need enhance the above example to find and display the duplicate words i.e. those repeated more than once. So output here should be – **a, is, of, this**.

Exercise 05: This is based on Parallel LINQ (PLINQ) *[5 marks]*

Create a console app (.Net Framework) and do the following:

Populate an integer array with 10 million elements, value of those elements should between 1 and 500. (use Enumerable.Range and random function).

Perform the following operation in a normal LINQ way and using PLINQ and compare their times taken:

a) Sum of all the elements in the array.
b) Counting the distinct elements in the array.