Purpose: Practice the use of generic methods and classes, extension methods, custom data structures such as Linked Lists, Stack and Queues.

Exercise 01: Generic Methods *[7.5 marks]*

Create a console app and write a generic method, **Search** [int Search<T>(T [ ] dataArray, T searchKey)] that searches an array using linear search algorithm.

a) Method **Search** should compare the search key with each element in the array until the search key is found or until the end of the array is reached. *[2 marks]*
b) If the search key is found, return/display its location in the array (i.e. its index value); otherwise return -1. *[2 marks]*
c) You need to populate the array with random values. (int values – between 10 and 49, double values between 50 and 99, char values between a and z). The size of array as 10. *[2 marks]*
d) Test this method by passing two types of arrays to it. (an integer array and a string array) Display the generated values so that the user knows what values he/she can search for. *[1.5 marks]*

[Hint: use (T: IComparable) in the where clause for method Search so that you can use method CompareTo() to compare the search key to the elements in the array]

Exercise 02: Extension Methods *[7.5 marks]*

a) Create a console app and implement an extension method – **CountWords()** for built-in class **StringBuilder** to count the number of words contained in a StringBuilder object. *[5 marks]*

[You need to add a separate ***StringBuilderExtensions.cs*** file containing class – **StringBuilderExtensions,** and adding **CountWords** method in there.]

b) In the class file, containing Main() method, you will be testing your method. *[2.5 marks]*

For example, if a StringBuilder object strobj = ”This is to test whether the extension method count can return a right answer or not”, the number of words contained in strobj is 16.

Or if strobj = “You can define extension methods for user defined types as well as predefined types”, then it will return the result - 14

Exercise 03: Generic Classes *[15 marks]*

A) You need to enhance your generic **LinkedListLibrary** (.dll file) class library project (which has been completed in the class/Lab) , by adding the following methods apart from the existing methods.

(Note: Create a new solution, and to that solution add a new class library project. Enhance it as per requirements.:)

1. T Minimum() method which will return the smallest item/node value *[5 marks]*
2. T GetLastNode() which will return the last element in the linked list *[5 marks]*

After that add another project- LinkedListLibraryTest to the above solution where you can test the above library by adding a reference to test project. Take out the build of the above library in the Release Mode.]

Test this library by using it in the project – **LinkedListLibraryTest** by creating two linked lists of integers and doubles (containing at least 5 elements each) and calling the methods **Minimum**() and **GetLastNode**()

B) You need to enhance class library project **QueueInheritanceLibrary** (generic version), which is derived from **LinkedListLibrary**, by adding following method apart from the existing ones:
- T GetLast() which will just return the last element in the queue and not delete it. *[2.5 marks]*

C) Test this library by using it in the project – **QueueInheritanceLibraryTest** by creating two linked lists based queue objects of integers and doubles and calling the method GetLast(). *[2.5 marks]*