Lab Assignment #01 – Using Generics Methods Collections – Part B

Student: _____

Due Date: **Week02/03**

Marks/Weightage: 20/8%

Purpose: The purpose of this Lab assignment is to:
- Practice the use of Generic Methods and Collections

References: Read the course's textbook chapter 5, notes and class code examples (You can also refer previous chapters if you need to.) This material provides the necessary information 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.
- The solution **must be named** using the first name and last name followed by Lab assignment number and section number. For the student name - John Smith, the solution name should be *John-Smith_Lab-1B_Sec001*, and project(s) name should be **John-Smith_Lab-1B_Ex01** for the first exercise, **John-Smith_Lab-1B_Ex02** for the second exercise, and so on**.**
- You will have to demonstrate your solution in a scheduled lab session and submitting the zipped solution/projects through the **Dropbox** link on **eCentennial**.
- You are required to follow the variable/control naming guidelines and must also implement exception handling in all the exercises.

Note (Very Important): *Late submission past due date is NOT allowed/accepted.*

Exercise #1:

(Generic Linear Search Method) Write a generic method, *Search*, that searches an array using the linear-search algorithm. Method *Search* should compare the search key with each element in its array parameter until the search key is found or until the end of the array is reached. If the search key is found, return its location in the array; otherwise return -1. Write a win form test app that inputs and searches an *int* array and a *double* array. Provide buttons that the user can click to randomly generate *int* and *double* values. Display the generated values in a *TextBox*, so the user knows what values they can search for [*Hint:* Use(*T : IComparable <T>)* 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.]

[5 marks]

Exercise #2:

(Overloading a Generic Method) Overload generic method *DisplayArray* of the class/lab example so that it takes two additional *int* arguments: *lowIndex* and *highIndex*. A call to this method displays only the designated portion of the array. Validate *lowIndex* and *highIndex*. If either is out of range, or if *highIndex* is less than or equal to *lowIndex*, the overloaded *displayArray* method should throw an

ArgumentException; otherwise, *DisplayArray* should return the number of elements displayed. Then modify *Main* to exercise both versions of *DisplayArray* on arrays *intArray*, *doubleArray* and *charArray*. Test all capabilities of both versions of *DisplayArray.*

[5 marks]

Exercise # 3:

Write a console app which makes use of generic **LinkedList** data structure (Refer code examples covered during the lecture) to maintain the list of students (add at least 5, you need to create a student class (Student.cs, with properties – ID and Name and appropriate constructors and overriding of ToString() method.) and demonstrate the use of following operations on the student linked list.

a) Add a student to the list (You need to define a method – **AddLinkedListItem(LinkedList** name **, Student** object**)** and inside that you can use built-in method of LinkedList class – AddFirst or AddLast
b) Remove a student from the list (You need to define a method – **RemoveLinkedListItem(LinkedList** name **, Student** object**)** and inside that you can use built-in method of LinkedList class – Remove)
c) Displaying the items of the linked list. (You need to define a method – **PrintLinkedList(LinkedList** name**) ,** and inside that you can use built-in method of LinkedList class**)**
d) Search a student (You need to define a method – **SearchLinkedListItem(LinkedList** name**, Student** object**) ,** and inside that you can use built-in method of LinkedList class**)**
e) Clearing the linked list. (You need to define a method – **RemoveAllLinkedListItems(LinkedList** name**),** and inside that you can use built-in method of LinkedList class)

[5 marks]

Exercise #4:

Write a console app which makes use of generic **SortedDictionary** data structure (Refer code examples covered during the Lab) to maintain the list of employees (add at least 5, you need to create a Employee class (Employee.cs, with properties – Name, Salary (type – double) and appropriate constructors and overriding of ToString() method.) and demonstrate the use of following operations on the above employee Sorted Dictionary list:

a) Add an employee to the dictionary (You need to define a method – **AddDictionaryItem(SortedDictionary<**int**,** Employee **>** var**,** Employee emp**)**
b) Remove an employee from the list (You need to define a method – **RemoveDictionaryItem(SortedDictionary<**int**,** Employee **>** var, int key**)**
c) Displaying the items of the SortedDictionary. (You need to define a method – **PrintDictionary(SortedDictionary<**int**,** Employee **>** var**)**
d) Search an employee (You need to define a method – **SearchDictionaryItem(SortedDictionary<**int**,** Employee **>** var, int key**)**
e) Finding an employee with highest salary (You need to define a method – **MaxDictionaryItem(SortedDictionary<**int**,** Employee **>** var**)**

[5 marks]