SIT221 –DATA STRUCTURES AND ALGORITHMS

LAB3: CONTINUEING WITH SORTING AND SEARCH ALGORITHMS

LAB OBJECTIVE:

Implementing common sorting and searching algorithms

SUBMISSION INSTRUCTIONS

Please submit your work to Week03 assignment folder

PREPARATION

1. Revisiting common sorting algorithms including Bubble Sort, Selection Sort, Insertion Sort, and Quick Sort in Workbook (section 7.2, page 103).

2. Download the template project available on the weekly resources folder. The solution has two projects: **DataStructures_Algorithms & Runner** projects.

3. In the Runner project, there is a Data Folder for Week01 – Note that the dataset has three files (1H.txt, 1T.txt, 1M.txt) [1H = 100 points, 1T = 1000 points, 10T = 10,000 points]

LAB TASKS

1. IMPLEMENTING COMMON SORTING ALGORITHMS

In this task, we need to implement and compare common sorting algorithms including Bubble Sort, Insertion Sort, Selection Sort, and Quick Sort.

Please use the same project template provided in this unit to complete your assignment.

In the DataStructures_Algorithms/Week03, you are provided with

- An Enum called SortingAlgorithm [BUBBLE, INSERTION, SELECTION, MERGE, QUICK, MICROSOFTSORT]
- Add ISorter interface (ISorter.CS) that has one method: **public void Sort(Enum SortingAlgorithm)**
- Vector class (copied from Week01). This Vector class implements the ISorter interface – which means you need to implement the sort method above.

In the Runner project,

- You are given Runner03_Task1.cs where the Run method will be started.
- Run method of Runner03_Task1 has two command line arguments (right click on project, display properties, select debug properties, and specify the following):
  - Input filename,
  - Sorting Algorithm name
- Runner03_Task1 should read data from the input file - argument 1, create a vector object from the input file, and then call the Sort method using specified Algorithm name – argument 2
- Finally write your sorted data to a file called SortMethod_InputFilename – e.g. INSERTION_1H.txt.
- Execution time & memory will also be displayed.

2. BINARYSEARCH

In this task, we want to extend the Vector class to support binary search. Please implement the following:

a. public int BinarySearch(T element) – searches a sorted Vector for a given element.
b. Public int BinarySearch(T element, IComparer comparer) – searches a sorted Vector for a given element using the provided comparer object.
c. Adjust **Runner02_Task2**, and add a search for part with Id = 1444, using binarysearch with & without comparer.