CIS 044:   
Introduction to Data Structures Using Java

Lab 6

**Instructor**

Dr. Maher Mneimneh  
Email:mahernm@gmail.com

**Guidelines**

**Please follow the guidelines below:**

**Submit the homework and lab solutions in the drop box. For a programming question, submit a .java file (for source code) and a .txt file for program output. If the problem involves other questions, submit a separate .txt file to answer the question.**

**P1 (20 points)**

Implement a binary search of an array iteratively using the method

**public static <T extends Comparable<? super T>> boolean inArrayIterativeSorted(T[] anArray, T anEntry)**

**P2 (30 points)**

Consider an array data of n numerical values in sorted order and a list of numerical target values (target values are not necessarily sorted). Your goal is to compute the smallest range of array indices that contains all of the target values. If a target value is smaller than data[0], the range should start with -1. If a target value is larger than data[n - 1], the range should end with n.



For example, given the array [ 5 8 10 13 15 20 22 26] and the target values (8, 2, 9, 17), the range is -1 to 5.

1. Devise an efficient algorithm that solves this problem and implement it in

**public static <T extends Comparable<? super T>>**

**Interval findInterval(T[] sortedData, List<T> targetValues)**

where Interval is a class that provides two public methods getLower() and getUpper() to return the lower and upper values of an Interval object. Implement the Interval class.

1. If you have n data values in the array and m target values in the list, what is the Big Oh performance of your algorithm?

**P3 (20 points)**

Write the java code for the method

**pubic static <T extends Comparable<? super T>> boolean isSorted(T[ ] a)**

which returns true if the array a is in sorted in ascending order. The code must run in O(n) time.

**P4 (30 points)**

Consider a revised selection sort algorithm so that on each pass it finds both the largest and smallest values in the unsorted portion of the array. The sort then moves each of these values into its correct location by swapping array entries.

1. Implement the modified selection sort using the method

**public static <T extends Comparable<? super T>> void modifiedSelectionSort(T[] a, int n)**

1. How many comparisons are necessary to sort n values?