UNIVERSITY OF SANTO TOMAS

COLLEGE OF INFORMATION AND COMPUTING SCIENCES

LABORATORY ACTIVITY 3

Year and Section: 2ITF

Name: Cassandra Eunice Cortez, Coleen De Guzman, Jan Vincent Dyogi, and Keila Santiago

Time Started: September 6 ; 5 PM

Time Finished: September 7 ; 6 PM

Topics Covered:

- Computational Complexity
- Sorting Algorithms
- Searching Algorithms

Objectives:

At the end of this activity, students should be able to:

- Design a searching algorithm.
- Design a sorting algorithm.
- Determine Big-O notation for a given sorting and searching algorithm.

Instructions:

- This is a group work.
- Submit your laboratory work as .pdf or .doc file through the “File Exchange” tool.
- Submission deadline: On or before September 8, 2022 (11:59 PM)

Tasks:

Answer the following and give the Time Complexity (Big O) for each item:

1. **Selection Sort – (25%)**
   a. How many swaps are performed when we apply selection sort to a list of N items?
      - n - 1 swaps are performed when the selection sort is applied to a list of N items.

   b. How many comparisons are performed when we apply selection sort to a list of N items?
      - n (n – 1)/2 comparisons are performed when the selection sort is applied to a list of N items.

      1. How many comparisons are performed to find the smallest element when the unsorted portion of the list has M items?
         - **m - 1** comparisons are performed when finding the smallest element of an unsort portion of a list of M items.

      2. Sum over all the values of M encountered when sorting the list of length N to find the total number of comparisons.
         - Summing m – 1 from 2 gives:
           1 + 2 + 3 …. + (n – 1) = (n – 1) * n/2

2. Write a Java class which implements **linear search**. It should take a list and an element as a parameter and return the position of the element in the list. If the element is not in the list, the program should raise an exception. If the element is in the list multiple times, the program should return the first position. **(25%)**

```
public class Linear {
    public static void main(String[] args){
        int[] array = {2, 3, 4, 5, 7, 6, 8, 2, 4, 5, 7};
        int element = 7;
        int pos = linearSearch(array, element);
        try {
            if (pos == -1) {
                  throw new Exception();
            } else {
                  System.out.println(element + " is found at index " +
            pos);
            }

        } catch (Exception e) {
```

```
                System.out.println("Element not found in the list.");
        }
    }
    static int linearSearch ( int[] array, int element){
        for (int i = 0; i < array.length; i++) {
            if (array[i] == element) {
                return i;
            }
        }
        return -1;
    }
}
```

3. Write a Java class that implements **merge sort**. It may help to write a separate class which performs merges and call it from within your merge sort implementation. **(50%)**

```
package com.ust;

import java.util.Scanner;

public class mergeSort {

    public static void merge(int[] lArr, int[] rArr, int[] arr, int
   lSize, int rSize) {

        int i = 0, l = 0, r = 0;

        while (l < lSize && r < rSize) {

            if (lArr[l] < rArr[r]) {

                arr[i++] = lArr[l++];

            }

            else {

                arr[i++] = rArr[r++];

            }

        }

        while (l < lSize) {

            arr[i++] = lArr[l++];

        }

        while (r < rSize) {
```

```
            arr[i++] = rArr[r++];
        }
    }
    public static void mergeSort(int[] arr, int length) {
        if (length < 2) {
            return;

        }
        int j = 0;
        int mid = length / 2;
        int[] lArr = new int[mid];
        int[] rArr = new int[length - mid];

        for (int i = 0; i < length; ++i) {
            if (i < mid) {
                lArr[i] = arr[i];
            }
            else {
                rArr[j] = arr[i];
                j = j + 1;
            }
        }
        mergeSort(lArr, mid);
        mergeSort(rArr, length - mid);
        merge(lArr, rArr, arr, mid, length - mid);
    }
    public static void printArray(int[] a) {
        for (int j : a) {
            System.out.print(j + "  ");
        }
    }
```

```
 public static void main(String[] args) {
     Scanner s = new Scanner(System.in);
     System.out.print("Enter number of elements in the array: ");
     int n = s.nextInt();
     int[] a = new int[n];
     System.out.print("Enter elements: ");

     for (int i = 0; i < a.length; ++i) {
         a[i] = s.nextInt();
     }
     System.out.println("----------------------------------------");
     System.out.println("Elements in array ");
     printArray(a);

     mergeSort(a, a.length);
     System.out.println("\nElements after sorting");
     printArray(a);
 }
}
```