**Lab Exercises 8: Sorting**

**LEARNING OUTCOMES**

These lab exercises is to

* Understand selection sort, insertion sort, bubble sort, quick sort and merge sort.
* Able to implement selection sort, insertion sort, quick sort and merge sort in coding.
* Able to implement Bubble Sort in coding Understand the Quick Sort and Merge Sort

### PROBLEM 1: SELECTION & INSERTION SORT

**LAB EXERCISE 8.1**

Estimate Time**: 2 *Hours***

**Tasks:**

Modify Program 8.1.1, “Selection Sort”, to count some criteria in measuring the algorithm’s efficiencies to order an array of 20 000 random numbers:

1. Display number of data compares.
2. Display number of data moves. A data move is a movement of an element of data from one position in the array to another, to hold area, or from a hold area back to array. Display the array before and after the sort. At the end, display the total moves needed to sort the array.
3. Display the running time in milliseconds and seconds.

**LAB EXERCISE 8.2**

Estimate Time**: 30 *Minutes***

**Tasks:**

Repeat Exercise 8.1 using the Insertion Sort (see Program 8.2.1)

**PROBLEM 2: BUBBLE SORT**

**LAB EXERCISE 8.3**

Estimate Time**: 30 *Minutes***

**Tasks:**

Repeat Exercise 8.1 using the Bubble Sort (see Program 8.3.1)

**LAB EXERCISE 8.4**

Estimate Time**: 2 *Hours***

**Tasks:**

Write a program to sort a string “THISISATEST” in descending order by using a **bubble sort**. The program must be done using the following rules:

* 1. String “THISISATEST” must be initialized into an array during variable declaration.
  2. This program must consists of two functions describes in Figure 8.1.
  3. Output of the program **must be same** as Figure 8.2.

void bubleSort(char array[ ], int arraySize)

{

// array will receive an address of QUESTION’s array

// arraySize will receive the number of characters is QUESTION

// this function will sort receive array in descending order using

// bubble sort

…..

}

void exchange(char array[ ], int index1, int index2)

{

// array will receive an address of QUESTION’s array

// index1, index2 : character to be exchange

// this function will exchange value between two index received

…..

}

Figure 8.1: Functions in the Program

Descending unSorted : THISISATEST

Sorted : TTHISISATES

Sorted : TTTHISISASE

Sorted : TTTSHISISAE

Sorted : TTTSSHISIEA

Sorted : TTTSSSHIIEA

Sorted : TTTSSSIHIEA

Sorted : TTTSSSIIHEA

Sorted : TTTSSSIIHEA

Sorted : TTTSSSIIHEA

Descending Sorted : TTTSSSIIHEA

Figure 8.2: Program Output

### PROBLEM 3: QUICK SORT & MERGE SORT

**LAB EXERCISE 8.5**

Estimate Time**: 30 *Minutes***

**Tasks:**

Repeat Exercise 8.1 using the Quick Sort (see Program 8.5.1)

**LAB EXERCISE 8.6**

Estimate Time**: 30 *Minutes***

**Tasks:**

Repeat Exercise 8.1 using the Merge Sort (see Program 8.6.1)

REFERENCES

[1] Keong, W. C., *Data Structures With C*, Sejana Publishing, Petaling Jaya, Malaysia. 2000. ISBN 983-2017-20-3. Pages 1-1, 4-5 ,4-6, 4-7,9-3,9-6,9-9,9-10,8-7, 8-9, 8-13

[2] Malik, D. S., *Data Structures Using C++*, Thomson Learning, 2003. ISBN 0-619-15907-3. Page 555.

[3] Sellappan, P., *Programming in C*, Sejana Publishing, Petaling Jaya, Malaysia. 1999. ISBN 983-2017-10-6. Page 11-8.