# 

# Department of Software Engineering

**CS 250: Data Structures and Algorithms**

**Class: BESE-7AB**

**Lab 8: Sorting Algorithms & Their Asympototic Complexity**

**CLO4: Investigate and evaluate various algorithms based on accuracy, time complexity, and memory requirements.**

**Date: November 17th, 2017**

**Time: 9:00 am -12:00pm, 2:00pm – 5:00pm**

# Instructor: Dr. Muhammad Shahzad

**Lab 8: Sorting Algorithms**

**Introduction**

In this lab, you will implement three sorting algorithms and compare them.

**Objectives**

Objective of this lab is to implement insertion sort and merge sort and compare the running times for both sorting algorithms.

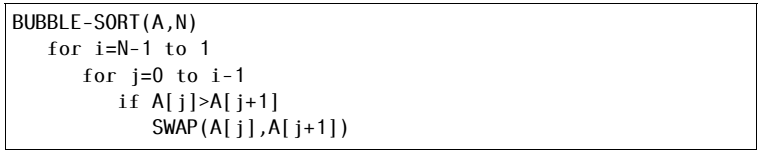
**Tools/Software Requirement**

Visual Studio C++

**Description**

**Bubble Sort:**

Insertion sort is a popular sorting algorithm, which is quite simple to implement. The pseudo code is as follows:



**Selection Sort:**

Selection sort is a popular sorting algorithm, which is quite simple to implement. The pseudo code is as follows:



**Insertion Sort:**

Insertion sort is a popular sorting algorithm, which is quite simple to implement. The pseudo code is as follows:



**Merge Sort:**

Merge sort is another important sorting algorithm that we have seen. Unlike insertion sort, it is not an in-place sorting algorithm. The pseudo code for merge sort is shown below:



Merge (Arr, n1, mid, n2)

a=n1, b=mid, c=n1 ,B;

while a <= mid and b<=n2

if Arr[a]<Arr[b]

B[c++]=Arr[a++];

else

B[c++]=Arr[b++];

while a<mid

B[c++]=Arr[a++];

while b<n2

B[c++]=Arr[b++];

for a=n1; a<n2; a++

Arr[a]=B[a];

**Lab Tasks**

**Task 1:**

Implement Bubble sort, Selection sort, Insertion sort and Merge sort algorithms in C++.

**Task 2 (average case complexity):**

The next step is to compare the two algorithms. Generate arrays of random numbers in the range 1 to 100 with sizes 100, 1000, 10000, 100000, and 1000000. Compare the running times of the three algorithms on each array. How do they compare? Are the results what you expected, and why? Answer the questions in at the end of the word file.

**Task 3 (best and worst case complexity):**

Now sort the arrays using stl::sort, once in ascending order and then in descending order. Given both sorted arrays as inputs to all three algorithms and compute their running time. The running time of which algorithm shows most variations based on the structure of the input and why? Answer the questions in at the end of the word file.

**Deliverables**

Students are required to upload the lab on LMS before deadline.

**Note:** Use proper indentation and comments. Lack of comments and indentation will result in deduction of marks. You will submit your workingcodes in **word document** (do **NOT** take screenshot of code, just copy your code and paste it). The name of word document should follow this format. i.e. **YOUR\_NAME\_Lab#**

# 