# Department of Computing

# School of Electrical Engineering and Computer Science

**CS-250: Data Structure and Algorithms**

**Class: BEEE 13 (Grp1+2)**

# 

# Lab 11: Implementation of Sorting Algorithms and Complexity Analysis

**Date: 22nd April, 2024**

**Time: 10 am - 12 pm**

# Lab Instructor: Anum Asif

# 

# Lab 11: Implementation of Sorting Algorithms and Complexity Analysis

=

**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++

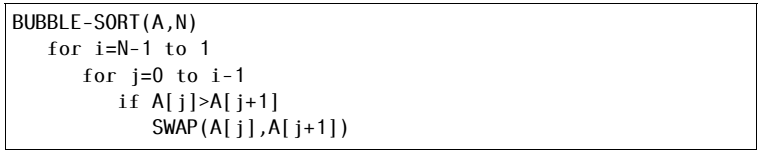
**Helping Material**

Lecture slides, text book

**Description:**

**Bubble Sort:**

Bubble 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 the solution section.

**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 the solution section.

**Important Note:** Practice your knowledge of OOP with C++ when creating a solution. Remember to comment your code properly. Inappropriate or no comment may result in deduction of marks.

**Solution:**

|  |
| --- |
| Solution |
| Task 1:  Task 2 :  Task 3:  Task 1 Code:  Task 2 Code:  Task 3 Code: |

### Deliverables

Compile a single word document by filling in the solution part and submit this Word file on LMS. Insert the solution/answer in this document. You must show the implementation of the tasks in the designing tool, along with your complete Word document to get your work graded.