# 

# Department of Computing

**CS 250: Data Structures and Algorithms**

**Lab 05: Sorting Algorithms**

**Date: July 27, 2023**

# Instructor: Dr. Syed Imran Ali

**Lab 05: Sorting Algorithms**

**Introduction**

In this lab, you will implement Selection sort, Bubble Sort, Short Bubble sort and Merge sort 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.

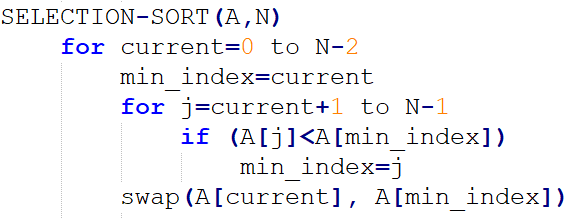
**Tools/Software Requirement**

Visual Studio C++

**Description**

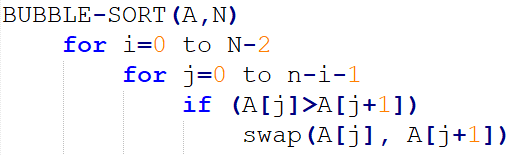
**Selection Sort:**

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



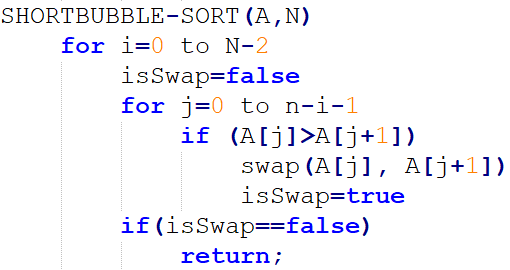
**Bubble Sort:**

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



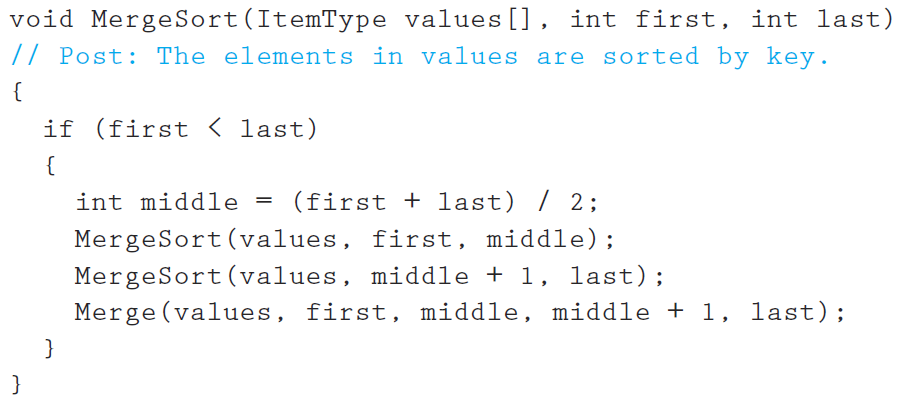
**Short Bubble Sort:**

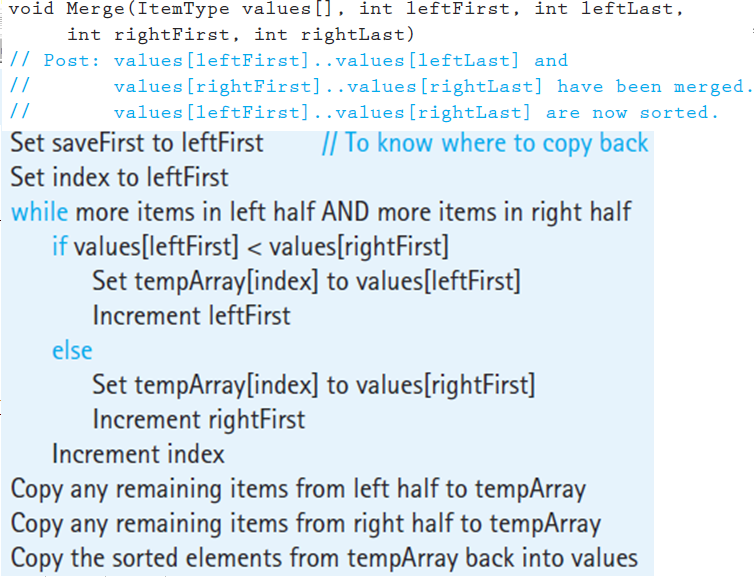
It is a variant of Bubbles sort the best-case complexity of which is Ω (n). Note that the best-case and worst-case complexities of Bubble sort are of order n2.



**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:



****

**Lab Tasks**

**Task 1:**

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

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

The next step is to compare the running time of algorithms. Generate arrays of random numbers in the range 1 to 1000 with sizes 100, 1000, and 5000. Compare the running times of the four 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 the four 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.

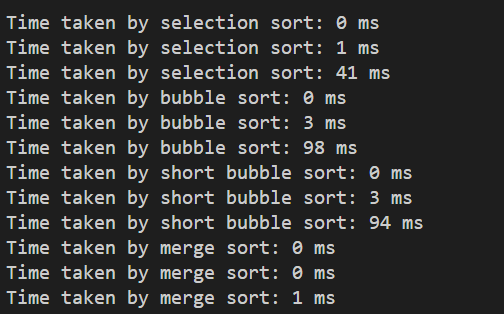
**Deliverables**

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#**

# QUESTION NO: 2

Output snap:



I have run the program multiple times to find the average running time for all the four algorithms. The algorithms are sorting arrays of sizes 100, 1000, and 5000.

It is evident that Bubble sort takes the most time to sort the random arrays. And merge sort is the most efficient sorting algorithm. We can deduce from the results that the running times are of the order:  
  
**Bubble sort > Short Bubble sort > Selection sort > Merge sort**

A screenshot of a computer screen

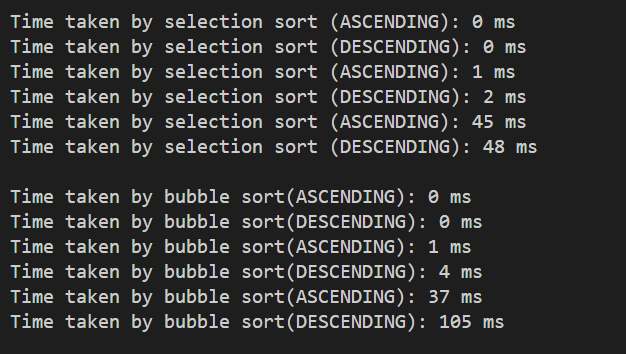
Description automatically generated

A screenshot of a computer screen

Description automatically generated

**QUESTION NO: 3**

Output snap:



I used the built-in C++ function sort() to sort the randomly generated arrays in ascending and descending order as asked in the task. Then, I gave these sorted arrays as inputs to the four algorithms. As evident from the output snaps of my program, short-bubble sort and bubble sort show the most variations in their running time respective to their inputs.

# A screen shot of a computer Description automatically generated