# 

# Department of Computing

**CS 250: Data Structures and Algorithms**

**Lab 05: Mergesort Quicksort Algorithms**

**Date: July 27, 2023**

# 

# Instructor: Dr. Syed Imran Ali

**Lab 05: Quick Sort**

**Introduction**

In this lab, you will implement merge and quick sort algorithms that have already been discussed in the class.

**Objectives**

Objective of this lab is to implement Quick sort.

**Tools/Software Requirement**

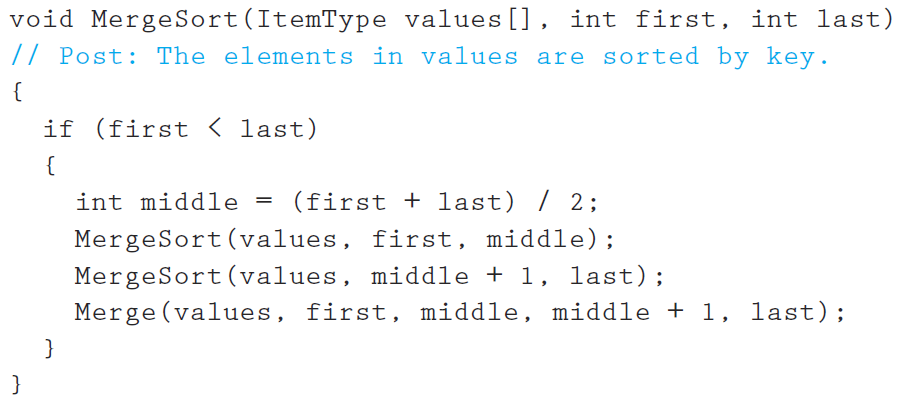
Visual Studio C++

**Description**

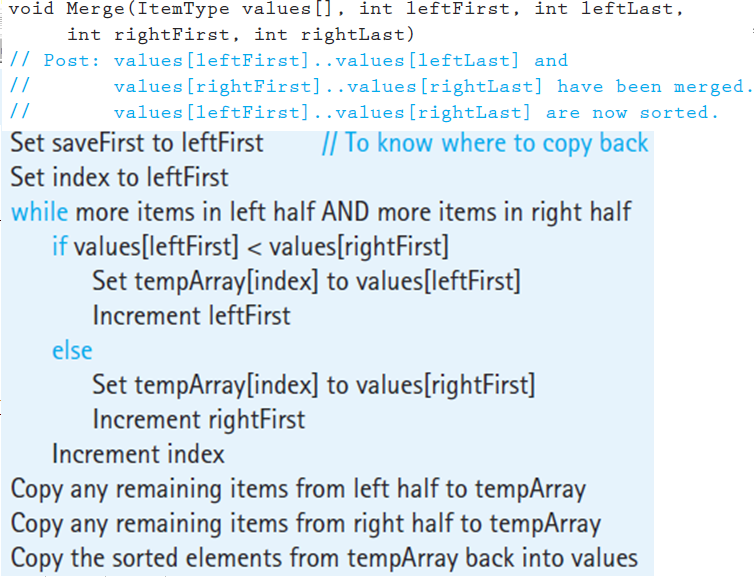
You will implement a version of the pseudo codes of merge and quick sort algorithms given in the book "Algorithms and Data Structures using C++" by Nell Dale.

**Merge Sort:**

Its best and the worst case time complexities are of order nlogn. Unlike insertion sort, it is not an in-place sorting algorithm as it requires a temporary array of size n to sort values.. The pseudo code for merge sort is shown below:



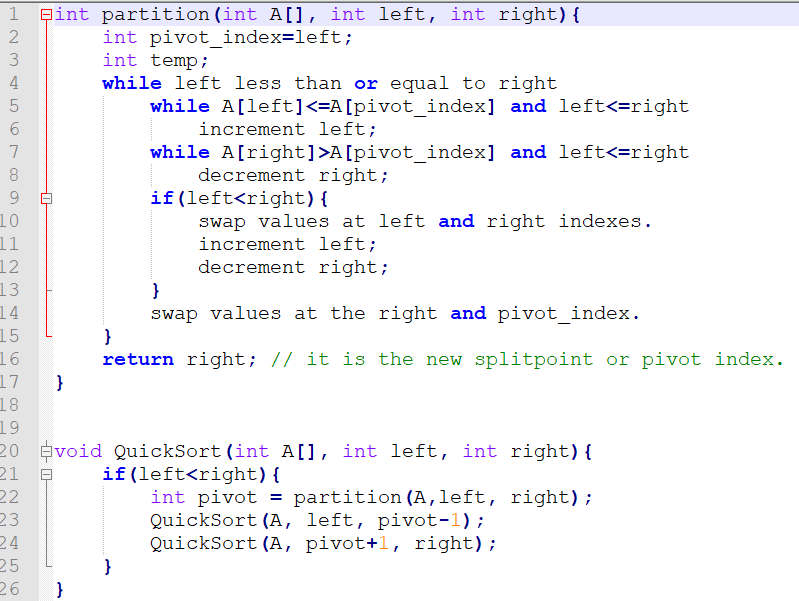
Make sure you print the updated array after a call to the Merge function terminates.

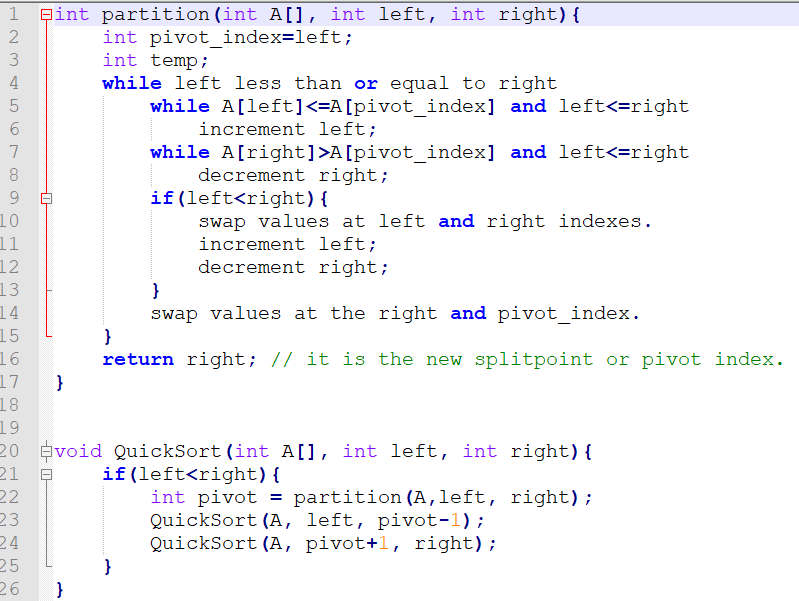
****

**Quick Sort:** It is also a divide-conquer based algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quickSort that pick pivot in different ways.

* Always pick first element as pivot.
* Always pick last element as pivot (implemented below)
* Pick a random element as pivot.
* Pick median as pivot.

The key process in quickSort is partition(). Target of partitions is, given an array and an element x of array as pivot, put x at its correct position in sorted array and put all smaller elements (smaller than x) before x, and put all greater elements (greater than x) after x. All this should be done in linear time. Once, done quicksort the left partition and quicksort the right partition.

****

****

**Lab Tasks**

1. You will run the algorithms on data that we used in the class to verify the results.
2. Compare how many partitions happened in each case.
3. To understand which calls to the recursive mergesort and the merge functions are made, include print statements in the first line and before the closing parenthesis of the merge sort function. Do the same for the merge function as well. Moreover, print the updated array-list after a call to the merge() function in the merge sort function.
4. To understand which calls to the recursive QuickSort and the Partition functions are made, include print statements in the first line and before the closing parenthesis of the Quicksort function. Do the same for the partition function as well. Print the updated array after line 22 i.e. when a new split has been created after sorting the pivot value.
5. Choose a different pivot value and see if number of partitions decreases. The new pivot value can be chosen as the median of the first, middle and last elements of the array.

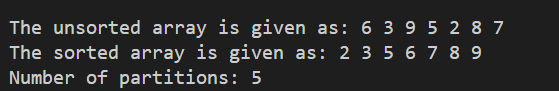
**Deliverables**

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

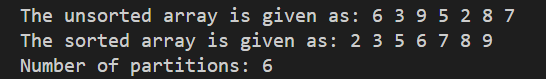
# Note: Use proper indentation and comments. Lack of comments and indentation will result in deduction of marks. You will submit your working codes in word document (do NOT take screenshot of code, just copy your code and paste it). The name of word document should follow this format. i.e. YOUR\_NAME\_L

**Question no: 2**

**QUICK SORT:**

****

**MERGE SORT:**

****

**Question no: 5**

when pivot = first element

**A black background with white text

Description automatically generated**

when pivot = last element

A black background with white text

Description automatically generated