**Colorado Mesa University**

**Computer Science and Engineering**

**CSCI112 Spring 2021**

**Assignment 4**

Due April 30 11:00PM

**Background:**

In this lab you will analyze and sorting algorithms.

**Task:**

In the lectures we discussed how a programmer can perform their own analysis of a sorting algorithm. One such approach involves counting the number of comparisons and plotting them against sample size. For this assignment you will implement 4 sort algorithms and see if you can identify their performance.

**You may not use arrays, you need to use STL containers and may use any STL algorithms, all must be meticulously commented – no clear commenting no grading.**

**The Algorithms:**

You will compare the relative performance of Quick Sort, Heap Sort, Shell Sort (research), and Merge Sort

**The Task**

The program Prac12main.cpp from practicum generates a sequence of random shuffles of a set of integers so that a sort function can be used and the number of comparisons plotted. If compiled with the -DPRINT option, the sorted data is printed instead of the plot data.

You will need to modify it to use vectors and any other STL container you deem of use. As it stands it requires a function with prototype void Sort(int[],int,int&) you will need to modify this to suit the container you use;

Sort sorts an array (modify to suit) of a specified length (size increase to 100,000), returning the number of comparisons made between data values for each of the 4 sorts, testing at 10k, 30k, 50k, 75k and 100k, a total of 20 experiments. Careful you do not mess up the shuffle of your original vector or alternatively reinitialize and reshuffle between experiments.

**Procedure:**

In the file sort.cpp, implement the function with prototype

void Sort( input args and reference args for returning data);

which uses the above sort to sort the data in container you chose, returning the number of comparisons per sort in the reference argument.

To test the sort algorithm, the file Prac12main.cpp contains a driver program to generate random shuffles of 100, 200, 300,…,10000 integers and call the sort function to sort them. You will need to make substantial modifications to this such that it generates the data needed for this task.

It generates a data file ready for a graph converter called converter2 (also in this task archive) ready for plotting but we are not concerned with this in this assessment.

If you want to experiment with plotting the source of this converter is also in this archive. The driver program can also be used to check if your sort is working by printing out the sorted data – remember there's a lot so use less to control it. To compile this version of the program use this **command line compile command**:

g++ -DPRINT L3main.cpp sort.cpp

For the assignment output a table containing your test result data for all 4 sorting algorithms and indicate which is most efficient given the input data vector size. Table will be 20 rows by 4 columns the efficient algorithm per row (cell) to be clearly indicated (use some iomanip or similar to make it stand out).

Submit your code files and in a simple text tile your **command line compile command and run command**. I will use these to test your program so make sure they work by testing on your *Ubuntu vm* **repeatedly**.

Grading is purely based on experimental run and code inspection, you may be required to zoom demo keep an eye on you CMU email daily for an appointment notice.