Fall 2023 – CS 201 Data Structures and Algorithms
Homework-1

Maximum points: 200. Individual Work Only.

Due Date: October 3, 2023 before 11:59pm (**late submissions get a score of zero**)

Objectives

- Implement heap sort and quick sort in C++ using iterators.
- Compare the performance of heap sort and quick sort.
- Compare the performance of heap sort and quick sort with the sort method provided by the C++ algorithms library

Problem Description

1. Implement heap sort and quick sort using the following function definitions:

   ```
   template< class RandomIt >
   constexpr void heapsort ( RandomIt first, RandomIt last )
   template< class RandomIt >
   constexpr void quicksort ( RandomIt first, RandomIt last)
   ```

   where [first, last) is the range of the elements to be sorted.

2. Use the drive program (functest.cpp) provided to test your implementation of heap sort and quick sort using iterators and make sure that your program sorts the data correctly with different types of containers and data types.

3. Execute your program for different problem sizes using heap sort, quick sort, and the sort method provided by the C++ algorithms library and complete the table below. You can use the driver program (perftest.cpp) provided to compute the execution time for different problem sizes. If you execute the driver program as is, it will display the time taken by the std::sort method, you can add calls to your implementation of heap sort and quick sort methods to compute the time taken and complete the table below and include the completed table in your report.

Execution time for different problem sizes									
Problem Size	std::sort			Heap Sort			Quick Sort		
	Case #1	Case #2	Case #3	Case #1	Case #2	Case #3	Case #1	Case #2	Case #3
10	1e-06	1.1e-07	1e-06	2.6e-06	1e-06	3.3e-07	2.3e-06	6.6e-07	3.3e-07
100	5.6e-06	2.3e-06	2e-06	1.2e-05	1.1e-05	3.3e-07	8e-06	5e-06	5e-06
1,000	7.4e-05	2.6e-05	2.2e-05	0.000173	0.00016	0.0015	8.73e-05	3.16e-05	4.56e-05
10,000	0.0009	0.0004	0.00034	0.00297	0.00203	0.00201	0.00169	0.00072	0.00786
100,000	0.01231	0.005152	0.00545	0.031885	0.028084	0.02492	0.016416	0.00730	0.00905
1,000,000	0.12949	0.049772	0.052346	0.40041	0.30965	0.32511	0.175389	0.10177	0.103344
10,000,000	1.37529	0.585826	0.611438	5.9506	3.36084	3.57077	4.13808	3.52951	3.50706
100,000,000	15.2949	6.97731	7.14934	77.0844	37.2247	40.5666	296.551	287.478	291.386

4. Based on the execution time in the table above, answer the following questions and include them in the report along with the table above:
    a. For each of the three sort methods, what pattern do you observe in the execution time as the problem size is increased for different test cases (in other words, if you look at each column in the table above, do you see any patterns in execution time as related to the problem size)? Did you expect to see this pattern? If so, why? If not, then what did you expect to see?
    b. How does the execution time of heap sort compare with execution time of quick sort for different problem sizes and different test cases? Explain the difference in execution time.
    c. How does the performance of your heap sort and quick sort implementations compare with the performance of the std:sort? Explain any similarities or differences you observe. What could be the potential reasons for these similarities and differences?

Program Documentation and Testing

1. Use appropriate variables names and indentation in your source code.
2. Include meaningful comments to indicate various operations performed by the program.
3. Programs must include the following header information within comments:

```
/*
  Name:
  Email:
  Course Section: Fall 2023 CS 201
  Homework #:
  Instructions to compile the program:
  Instructions to execute the program:
*/
```

Submission

Upload only the source files (.h or .cpp or .cc files) and include the table and answer to the questions above in a separate document (Microsoft Word or PDF file) and upload it to Blackboard in the assignment submission section for this homework. Do not upload zip/tar files to Blackboard, upload individual source files and Word/PDF file.

Grading Rubrics

The following grading policy will be used for grading this homework:

Description	Points
1. Correct implementation of heap sort using iterators	75
2. Correct implementation of quick sort using iterators	75
3. Execution of all three versions and completing the table	25
4. Answer to short answer questions	25

NOTE: A correct implementation must implement the algorithms using iterators.