EECS 560 Lab 12: Sorting

Due date
11/22/2020 Sunday, 11:59 pm

Objective
Get familiar with quick sort with C++.

Requirements to complete the template

1. Create a header file **“mySorting”** containing all following **template-based functions**.
2. Implement the quick sort driver **void quickSort(std::vector<Comparable> & a, bool reverse = false)** according to Figure 7.15. Modify the code so that the algorithm will sort **a** into non-descending order when **reverse** is **false** (by default), and into non-ascending order when **reverse** is **true**.
3. Implement the main quick sort routine **void quickSort(std::vector<Comparable> & a, int left, int right, bool reverse)** according to Figure 7.17. **Use cout to print the pivot element.** Modify the code to support different sorting order.
4. Implement the function **const Comparable & median3(std::vector<Comparable> & a, int left, int right, bool reverse)** that returns the pivot element of **a** from index **left** to **right** (include both end). Refer to Figure 7.16 and modify the code to support different sorting order.
5. Implement the function **void insertionSort(std::vector<Comparable> & a , int left, int right, bool reverse)**. Refer to Figure 7.2 and modify the code to support different sorting order.
6. Any other functions you need to implement the previous functions.

Testing and Grading
We will test your implementation using a tester main function, on a number of instances that are randomly generated. We will release the tester main function, several instances (will be different from the grading instances but will have the same format), and expected output together with the lab instruction via Blackboard. You code will be compiled under Ubuntu 20.04 LTS using g++ version 9.3.0 (default) with C++11 standard.
The command line we are going to use for compiling your code is: “g++ -std=c++11 main.cpp” (note that main.cpp will try to include the .hpp file you submit, and your .hpp file needs to be property implemented to compile successfully).

Your final score will be the percentage your program passes the grading instances. Note that if your code does not compile (together with our tester main function), you will receive 0. Therefore, it is very important that you ensure your implementation can be successfully compiled before submission.

Submission and Deadline
Please submit your implementation as one .hpp file, with file name “mySorting_[YourKUID].hpp”. For example, if my KU ID is c123z456, my submission will be a single file named “mySorting_c124z456.hpp”. Submissions that do not comply with the naming specification will not be graded. Please submit through Blackboard.