CS/SE 1337 – Homework 3 – Sorting & Searching

Dr. Doug DeGroot's C++ Class

Due: Saturday, March 14, 2020, by midnight

How to submit: Upload source code and a copy of your program's output. Be sure to include a copy of ***all*** output produced by your program.

Use a file name such as HW3-CS1337-<firstname>-<lastname>.cpp.

(Do not name it just main.cpp when submitting it.)

Maximum Number of Points: 100

Objective:

Write a program that performs the following tasks:

1. Generate some number of random numbers, perhaps 30 or 40-ish, or more. Ask the user how many numbers to generate, though. Then ask the user how large the random numbers should be, but don't accept a number larger than 20. (That will help ensure that there are multiple numbers of the same value generated by the random number generator. If you don't see multiples of several numbers, restart the program and either use more numbers or a smaller spread of random numbers.)
2. As you generate the random numbers, store them all in a fresh, new, empty vector. Let's say this vector is called vec1.
3. Next, show the vector's contents, size, and capacity. (The vector will be unsorted.)
4. Make a copy of your vector, simply (not by manually copying it element by element). Let's say you name the second vector something like vec2 or vecCopy, (Feel free to use more creative names.)
5. Sort the numbers within the first vector you created using the sort algorithm in <algorithm> (see pages 1087-1090 of our text.) Try something as simple as this:

   sort(vec2.begin(),vec2.end());
6. Show the vector's contents again. They should be sorted, and there *should* be duplicates.
7. Now write a routine that sorts vec2 using the bubble sort or the insertion sort algorithm (your choice).
8. Show the contents of vec2 to prove that it is also sorted.

We're through with vecCopy at this point and in the following instructions.

9. Now we're going to search for some numbers in the original vector. Create a routine that asks the user to enter some number. Search the vector (vec1) for this number and report to the user whether the number was found, and if so, how many times it can be found in the vector. Do this for at least three to five numbers, asking the user for a number each time. **Then be sure to search for at least 2 additional numbers that don't appear in your vector.**
10. If the number sought by the user is found, report how many times that number appears in the vector. **Use the binary search method** to find the first occurrence of the number, if it's in there. Then look both before that number's position and after the number for duplicates of that number. Keep a count of the number of times you find that number in the vector. **Do not use any STL functions to count these duplicates, and do NOT start at the beginning of the vector and search sequentially.** (If you want to use the STL routines, do so after you search before and after the found element, and report the count both times.)

11. All output must be adequately labeled and user-friendly. Use informative header lines for your output.
12. Use good program decomposition techniques. Try to stick to one major control construct per routine.
13. Make a copy of your output and include it with your homework submission.
14. Once you've asked the user for 2 or 3 number choices to search for (steps 9 and 10), **delete** all duplicates from the sorted vector. Don't use a second vector to copy into, as that would be too simple, but delete the duplicates from the original vector itself. Report the results. Do this manually, not with STL.
15. **Be certain to take a screen shot of all your output and upload it with your homework submission.** Make it easy for the graders to follow along with what you program does/did. That is, have your program talk to you/the user and the grader.

Annotations for the code:

1. The main function can be at either the beginning or the end of the program. I don't care which.
2. Add comments at the top of your main.cpp file to include your name, the name of the program, and notes on how your design works when executed. Point out any special features or techniques you added using a comment saying "**// Special Features**". Keep a //changelog and a //notes section.
3. Comment your code effectively, as we discussed in class. Use descriptive variable names everywhere so that the code becomes as self-documenting as possible. Use additional commentary to improve readability and comprehensibility by other people.
4. You absolutely MUST use consistent indentation and coding styles throughout the program. Failure to do so will result in a loss of three points.
5. If the program does not work at all, or works incorrectly, 10 points will be deducted.
6. No late submissions will be accepted. Please meet the deadline. Ten points will be deducted for every day your submission is late.