CS313 HW_1

HW File: (ALL CODE MUST BE IN C++)

All of these programs should have an accompanying “white paper” explaining what you did. They should be commented fully as well. Please record a video presentation explaining what you did in a concise and clear manner.

1. Create your own vector and linked list. They should each have all the necessary operations such as insertion, deletion, search, etc. You should create an iterator for them as well. Test the following to see which is faster:
    a. Filling them with random numbers
    b. Filling them with random strings (use ascii)
    c. Doing the above with move semantics when filling with random strings
2. Test whether a recursive, iterative or linked-type binary search is faster by testing it on arrays of sizes 1 million and 10 million with arrays that are filled with random numbers. You will need to either fill it in a “sorted way” or sort it before doing the binary search.
3. Create a sort for 2d matrices. Do a bubble sort, insertion sort, and selection sort variation of this. Do NOT convert the matrix into a 1D array to sort. You must sort it as a 2D structure.
4. Compare the times it takes to sort an array filled with random numbers vs a linked list via bubble sort and insertion sort.
5. Create a multi-level sort. For instance, for all selections of n > 10 you do sort X and within sort X, when you have a situation with n < 10 you do sort Y. Be creative. Time your sort against two “reasonably comparable” sorts (you may use libraries for the “reasonably comparable sorts”).
6. We have received a secret message encoded with a Vigenere cipher (wiki). We know that the key is 32 digits long and a brute force attack would take us a long time (see here). We do not know any “tricks” how to break a Vigenere cipher, but we do know how it works. Instead we are going to test random possible solutions and use a feature function to evaluate the possible solution. We will collect solutions that pass a certain threshold and mutate those solutions with some low probability, hopefully getting closer to the ‘true solution”. We will also use recombination (see here) , with some low probability, continually increasing our threshold as we narrow in on a solution. Create a program that does this, paying special attention to the data structure you wish to manage your possible solutions.
7. C++ has several containers (https://www.cplusplus.com/reference/stl/). Pick three of them and compare similar operations by timing them on a large scale. Explain a hypothesis, your results, and your explanation.
8. Create a *templated class* that *effectively* finds all possibilities of a list of random numbers that adds to some s. In other words, create a class that asks the user for the size of an array and for some number s. Fill in the array with random numbers. Create function(s) to solve for every possible output that sums to the number s. For instance, if your random list were {1,2,4,6,9,9,7,11,16} and s = 18, then you should output (11,7) (9,9) (11,4,2,1) (16,2) (11,6,1) (9,7,2) (9,6,2,1) (7,6,4,1).

9. Create a random array of size k. Create a function to check if it is a deBruijn (https://en.wikipedia.org/wiki/De_Bruijn_sequence) sequence of B(n,k). If it is not, randomly mutate each spot in the array (from a 0 to 1 or 1 to 0) with a 5% probability, keep doing this until you have found a deBruijn sequence. Do this with an array structure and a linked structure. Do this 100 times for each, time it, and compare your results.
10. Create a function for a linked list that sorts the list so that first all odd numbers are listed and then all even numbers.
11. Create a class called Restaurant. Give it the necessary structure, variables, and functions, and create a pure virtual function called menu. Create a class called Italian_Restaurant, Greek_Restaurant, and Chinese_Restaurant as derived classes. Use the parent classes constructor as well as any other variables that need to be created. Create a templated class called Reader_Robot. It should take in *any* kind of restaurant and use polymorphism to read the menu (that you overrode). Show how this works in main.
12. Create a doubly Linked List that does insertion and deletion in the correct position. Alice is an office manager. She uses her doubly Linked List to insert appointments in the proper position. A new list is created each week to hold appointments from Monday to Friday, 9am-5pm. Each appointment can be anywhere from 30min to 2hrs. Alice schedules all appointments the week prior. Her rule is that if a user asks for an appointment slot that is taken, she will find you the closest available one (on her doubly Linked List - which may mean that a Friday appointment will be scheduled on a Monday). Show with *ample* explanation how Alice fills in the appointments for the week.
13. Bob has a doubly Linked List, where each spot corresponds to a job that must get done, aptly named job_1 through job_25. Bob has 25 workers on his team (worker_1 through worker_25), he has another doubly Linked List that has the name of each worker. He uses these two lists to assign workers to jobs.
    a. Show how Bob can do this, by creating and running a function that outputs every worker and their assigned job.
    b. If there is a problem with the assignment, Bob “turns” the list against each other, such that each worker will be assigned to a previous job. For instance, if worker_4 were assigned job_4, now worker_4 is assigned job_3, and worker_5 is assigned job_4. Show how Bob manages his workers to adjust for problems.