Kunio Comey

CS202

Analysis 1

Effectiveness of Design:

The effectiveness of my design and classes was nice from the client perspective. In the client, all I needed to have was one object that had all the other classes it derived from in it as well. Having a class hierarchy allowed the client to call a large array of functions that were in the hierarchy. I also decided to create a string class to automatically deal with the allocation and deallocation of memory for a dynamically allocated character array. This helped a lot since the design required a lot of arrays of characters. This drastically cut down the complexity of the coding experience.

Validity of Approach:

Overall this approach was valid. All the legwork for the program was done by individual classes that were connected to each other. Each class had their own functions to manipulate the data in their own class so that the client only needed to call the function rather that manipulate class data themselves. This helped keep all of the code organized and easier to implement since each class had their own job.

Analysis in terms of OOP:

Like I said above in the validity of approach section, most of the jobs were separated by the objects dealing with their own portion of data. Having the class hierarchy allowed the client to have easy access to the large range of functions within that hierarchy. For the data structures, they all had containing relationships with the other classes so that it was easy to duplicate the classes in the data structure classes. None of the clients of a given class changed the data members inside of the class.

Major Changes to Design:

The biggest change to the design was throwing out the idea of a user class holding the data structures. I felt this added another layer of complexity that was not necessary when the main function could easily simulate a user. If this program was bigger than the assignment guidelines, then maybe it would be necessary to have user classes. Another big change was having a string class in the program. I made this change because the problem required using a lot of dynamically allocated character arrays. Implementing a string class made it much easier to create new objects with character arrays as data members since I could reuse the code for the string class to deal with memory management.

Efficiency of Approach and Efficiency of Resulting Code:

The approach made the code more organized and easier to debug. The most inefficient part of this program was the class handling the array of linear linked lists. If it used more pointer arithmetic rather than using the subscript operator, it could have been more efficient. Also the display function for that class wasn’t too easy to read from the client perspective. Overall though the approach and the code itself allowed the solution to be solved in organized chunks.