Ryan Filgas
4-23-2021
CS202
Karla Fant

Program 1 Efficiency Write-up (See debugger write-up on P2)

The effectiveness of my design was influenced by a lot of factors, one of them of course being time. Starting this project I had a vision that all of my data going in and out would be sorted, and all of the data structures would be implemented in a way that makes sense for their applications. This was partially true. I found that a linear linked list of arrays is actually not a horrible structure to implement for small amounts of emails. It may not be fast access, but it allows for easy detaching of nodes, so they can be shifted to other categories. My LLL worked fine as a priority queue, however I didn't have the time to implement it as I would have liked. For example if I'm going through the trouble of ordering the data I should have a function that can pop the data off of the list. For my CLL, I did a list of 7 days, but I didn't use a CLL for its best purpose, traversing rear for each day. It would have been good if the user could cycle through each day, rather than choosing a day to display. It also would have been good if the data in the CLL was organized by time, and the arrays grew in size if they were too small.

Implementation wise, I feel I did ok for what was there. I really focused on breaking down problems so I didn't have to do work twice, although I didn't accomplish this everywhere. Using a retrieve function to help in the implementation of my other functions was a huge time saver. Using recursion where I did helped break down problems as well.

As far as OOP methodologies go, I stayed within this, but for one exception. In implementing my copy constructor for the LLL I used my add function to make the new list, which was perhaps a mistake. Since the list was a priority sorted list, I needed that data member, so I called a function to return an int.

I made one major change in the implementation of my design, and it was a huge mistake that cost me time. I allocated an object * instead of an object ** for my cll class, not realizing that I couldn't use 'new' on an individual pointer since it's already allocated. I had implemented several functions by the time I realized it wasn't going to work. At the same time as this I was trying to retrieve an appointment object from a node through a return statement, and it was causing too many errors. I spoke with a TCSS about them, and before this I was under the impression that nodes shouldn't manage data. I was assured that they should manage their own data if they are a class, just not their next pointers; otherwise I'm violating the purpose of a class. Having lost a lot of time, I reluctantly deleted half a day's work, and went back to redesign. Efficiency wise, I would have liked to use pointers more than I did. At the point I had the choice, I was having enough syntax errors that I thought it better to keep it simple given the deadline was coming.

Ryan Filgas
4-23-2021
CS202
Karla Fant

Debugger Write-Up Program 1

Using gdb and valgrind helped me immensely with this program. Gdb allowed me to step through each line of the program, and was instrumental for finding seg faults, and in some cases memory leaks. I had one particular leak caused by a function that was supposed to be returning a value upon success, but instead it returned failure 100% of the time. I used the debugger to analyze my data structure traversals as well. Using the display function in gdb, I was able to check the address of individual nodes and pointers to make sure everything was being allocated correctly. I learned a lot as well. I worked a lot in the –tui visual mode, learned how to use checkpoints to get back to a specific place, using bt to back trace where seg faults show up. I also learned that I could preload file inputs in a .txt file, and run them in gdb so I could focus on debugging, and not putting the same input in over and over. I do really want to spend some more time running gdb through the creation of hierarchical relationships, as it relates to constructors. I spent most of my time debugging, and not following curiosity. I do look forward to using it as an intermediate test to see what my program is doing early, rather than after I've programmed in input, traversal, display, etc. It's going to make it easier to build from the ground up if I use it wisely.