Eduardo Robles

CS-163

Efficiency Review Program #1 Write-Up

For my program I designed a linked list of subjects and each subject contained another linked list of ways to get assistance for that particular subject. I split the program into multiple classes to allow flexibility and easier debugging. The task of the assignment was to build some form of list that contain subjects. Each subject was to have another type of list to store help information. The linked list data structure was a good choice for this type of task. A link list is not fixed and can always be expanded and thing can be added at the front or the back. Adding a subject was simple movement of pointers and comparisons. The link list also has the concept of nodes. A node can contain anything you want you want inside from objects to functions. This made it really it easy to store a subject and its list of ways to get help. In my program my node had a spot for a name, list object and a pointer to the next node. This made things easier to keep track of and problem solve. Overall I think the linked list performed well and it was able to handle all the tasks given.

I think the linked list data structure is a simple data structure that works pretty well for the given task. All were doing is holding information and adding to the list. That’s what the linked list is designed to do. Because linked list uses dynamic memory, we can keep on adding as we please and don’t have to worry about copying data over anywhere if we were to get to full. This data structure allows for easy maintenance and flexibility. A doubly linked list could have also been chosen as the data structure. This would allow for easy traversal back and forth in the list. Although it would be way more operations and slower than linked list.

This program required a linked list of linked list. There was to be a list of subjects and also that subject were to have list of ways to get help. For my design I created 3 separate classes to store each component of the linked list. I had an Assistance class that just stored all the information about how to get help for a subject. This made is easy to keep all the information in one place. The next two classes I created were AssistanceList and List. The AssistantList job was to create a list of assistances. It added or modified assistances contained in its list. This is made it easy to modify any data without interacting with the subjects directly. The list of subjects was its own class and primarily just focused on adding or displays subjects. This made it really efficient to do any changes with the list. I didn’t have to worry about having to insert assistances because I already have a class that does that for me. I designed this program so that each component was its own class. This allows for quick changes to specific things and allows for better debugging.

Some of the downfalls with my design I think primarily have to do with memory and use of recursion. For instance, in my List class I have a node structure that contains a AssistanceList object to manage adding or removing from a subject. I think instead of an object I could have also had a pointer to a node of the assistance type. Instead of creating a whole object and building a list, I think the use of pointers could be faster. For certain functions I think I could of used iteration for more efficiency and clarity. I think for the most part the design was good and allowed for the worked to be split up instead of having just one class. If I had more time I would try and find ways to break certain tasks into smaller helper functions. Some of my functions have a lot of lines that could be re-used and put into a function.