This report will explain what each section or function does as it is called by the main function. It will further go on to explain the reasoning behind some of the decisions. Some of the decisions can be changed but most have valid reasons behind each of them and are required.

The main function first calls the function testFile. This calls up the file and checks each line for a minimum of two comma separated values. If two are not found, then it tells you and ends the program. The next function to be called is the mainMenu function. This is a function that will display the menu on the screen so that the person running the program can decide what to do. At this point the user can choose one of four options.

The first is loading the file. This just loads the file and creates a linked list with each of the two or three comma separated values. It also calls another function to convert and sort the class code into a numeric value and attaches that into the linked list as well. After this it returns to the menu again. The next option is to print the class list. This function goes through the linked list printing all but the numerical class code since that was only used for sorting. The third option is to print specific class information. The function for this asks for a class code and then searches the linked list for that code. If found it will print the information, otherwise it will say the class was not found. The final option is to exit the program. There is another built in option if something other than one of those options is chosen stating to try again.

This will be the explanation why some of the features or math were chosen. The first being the linked list. The decision to go with a linked list is due to the current size of the class list. Due to there being only eight classes offered, most sort and search methods would have similar times. If the class list were to grow, a binary tree would be the better option for this. There was no mention of this list growing so it was assumed that these are the only classes to be offered in the foreseeable future. The binary tree would have a search at best a search time of O(log n) as best case and worst case be O(n). The linked list would be a linear search would have the same O(n) that a worst-case scenario would have. (ZyBooks, 2021)

A picture containing text, line, plot, screenshot

Description automatically generated

According to my research, the above chart (ZyBooks, 2021) shows that O(log n) and O(n) have quite different computations as the number of options increases. In reality, the difference in time for a modern computer for only eight options is negligible. If the number of classes were to increase to 10 or more, then the binary search would be the better way to search the list.

The next thing to talk about is why the class code numerical conversion was chosen. This was simply a way to sort them numerically instead of using a string-based method. Even a string-based method would fundamentally do something similar behind the scenes. This also gave a better way to sort if more class were to be added the future. The only code that would need to be changed is the linked list to a binary. The one downside would be that there may be duplicate numbers. For example, AAKC has the same numerical value as DCEB. However, since most of the alphabet part of the class code is reasonable, in English, this is unlikely to occur. Also, they would need the same numerical portion as well. They are not just some random assortments of letters. The letters CSCI is 33 and MATH is 40. There is no conflict with any current classes.

The pseudocode turned out quite well using previous iterations and ideas to culminate in this. The idea of converting each to some numerical value happened during this final version of the code. It was done for if a binary search would be specifically requested.