Project 3 Design Document

EclipseR – The main class of this project reads in an initial input file, performing checks on bad eclipse data and keeping the good. Once the file is read, the user can pick from numerous options C,O,H, and L. The main function idiot checks the user and loops until good input is input. Once the user checks a valid answer, the main function outputs specially for user, sorts specially for user, finds objects specially for user, purges information specially for user, or merges information specially for user.

ResizableArray – A templated array for storing any kind of variable. I used the array to store primitive data and to sort and search the eclipses of the data file and I also made the data mostly encapsulated so data cannot be inversely manipulated. I also have many functions that I threw in for later, just in case I need them.

DoublyLinkedList – A templated linked list that can store any variable, but I created the linked list to store all Eclipses. I made my linked list doubly so I can iterate from either side based on size and the specified position. It cuts on time complexity. I also have many functions that I threw in for later, just in case I need them.

HashTable(**HashNode, HashMap, LinkedHashMap**)

> **HashNode** – A templated Hash Node that creates a struct like node that is templated so the key or the value can be objects of any type.
>
> **HashMap** – A templated Hash Map that creates a array and uses Separate chaining as a means of collision resolution. It also uses a remainder of table size to hash the key. This is a simple hash, but since we are using **integers that are mostly between 0 and Table Size,** this is a perfect hash to keep thing close and have least collisions.
>
> **LinkedHashMap** – A non-templated Link for the hash map and the doubly linked list. This is the biggest design choice I had. I am able to manipulate the eclipse lists easier and can show if the linked hash map is not syncopated to the linked list it is linked to. This class holds insertion, removal, and selection of eclipses.

Eclipse – This Class contains the functions for the Eclipse object. I didn't get a good description on this, so I will also mention that I made every variable copy accessible through accessor functions. I used the constructor to process the data line so main can be clear of clutter and I. I also used 18 different variables to store the data to make it easier debugging. Each eclipse has an error line associated with it. The error line shows that it is an incomplete or incompatable eclipse with the other eclipses that are to be stored.

Search – I kept my search linear. The array can be unsorted this way, so O(n) is the worst no matter what happens. The actual search takes place in a small portion of this class, but the column check is the bulk of the class. Unless the specified column is 1, then this search is implemented. If column 1, then hash table is used.

`Sort` – I have a stepped quicksorting algorithm for sorting any column I want to. The actual sort takes place in a small portion of this class, but the column check is the bulk of the class. This sort only occurs if the column is not 1. Column 1 is already present in the doubly linked list.

`HelperFunctions` – This class has many functions that keep clutter from the main function. This is the only reason I have it. A few example functions in this class are header printing, integer checking, and double checking.