For my final project, I attempted to use my dataset "2018_Flights.csv" to create a minimal spanning tree in order to find the best and cheapest way to travel to all of the locations. A minimum spanning tree (MST) is a subset of the edges of a connected, weighted- undirected graph that connects all the vertices together, with the minimum total edge weight. The origins and the destinations of the areas traveled to and from had an integer associated with them to determine which location is which. For example, Philadephia would be categorized as 23 under OriginWac. I first made a struct for my Dataframe that included all of the columns in order to read them and create a graph. My next step was to create a forward and backward hashtable after reading my CSV into my program. My hashtable is composed of an i64, and a Linked List that contains access to the rest of the columns. The i64 of my HashTable was the origin and then I iterated around each row to push the other columns: Origin string, Destination String, etc into an empty LinkedList. My edges consisted of three categories: The origin of the flight, the Destination of the Flight, and lastly the weight which would be the PricePerTicket divided by Miles. The goal was to use these edges in order to create an undirected graph and then use KruskelMST to receive a Vector of Edges. Then I would've used the hashtable in order to find the values associated with the edges and be able to locate the string of the location to the OriginWac. However, after creating a Hashtable with an int and a linkedlist I was not able to fully perform an executable minimal spanning tree. For example, I should have received only one origin and destination with the lowest priceperticket per mile. Since my dataset consisted of over 1000 rows, there are many rows that have the same origin to destination with a different price associated with it. My hashtable was made to remove those repeated depending on the priceper ticket, due to the fact that "match" was implemented in it. For my KruskelMST I tried to find the parent in order to be able to find where the graph is connected first by. I then attempted to merge them together as long as the next edge connected was not the parent node since it would already be used for the first edge in the tree.