|  |
| --- |
|  |

**Delasi Dake**

**51692024**

**Intermediate Computer Programming**

**Cohort A**

**Dr. Robert Sowah**

**November 29th, 2022**

**Reflection**

This project was one that got me thinking more than needed due to many problems we encountered from the four stages of the work, the reading and parsing the input files, querying the data for the specified information, search Algorithm and writing to an output file.

Since a map data structure can contain key-value pairs, we were able to arrange the CSV data in a way that made it simple to retrieve particular pieces of information. In general, they kept track of airport, airline, and route objects in a String Key and Vector. The data for the object is extracted line by line from the CSV file and divided into several pieces that are supplied as parameters to the airport, airline, or route builders. The start and destination are read from a text file into a string.

We queried our data using varied start and destination cities and this generated the appropriate information. We used the chosen Airports to search for the flights options which connected the selected airports. To make sure a legitimate flight path is taken when the search is finished, a query is run again.

Within the creation and use of our search algorithm, the name and ID of the airport were extracted and utilized to build the node of a graph. The adjacency matrix of the graph is used to represent it as an undirected graph. A modified breadth-first search that iteratively checked all nearby nodes and returned a path List was the one that was employed. All potential pathways are returned by the search algorithm. Following the retrieval of the paths, I calculated the shortest distance and included it to the file. I may have returned numerous pathways since I did not account for the weight on the edges (distance derived from longitude and latitude).

With writing to the output file the output file with the provided name is formed when the data from the Breadth First Search has been written into it.

I have concluded that some improvements I recommend is that I must understand memory management and code optimization in order to shorten runtime since I frequently run out of memory during implementation.

I believe that the code could be made more efficient by eliminating pointless assignments, making it more compact by recycling some results from running the queries rather than writing a new one each time, and using a search algorithm that takes the separation between airports into account when calculating the shortest path.

For subsequent projects, I'll research ways to engineer the Dijkstra's algorithm to better meet the project's demands.