**ICP Individual Project 2 Reflection – C++ Implementation of Flight-Finding Algorithm**

This project aimed to find a flight or list of flights, given a start country and city and a destination country and city. The input to the program was to be a text file containing the location information and the output was to be a text file containing a list of flights to be taken from different airports between the start and the goal.

The first task was to represent the airports in each city. For each airport in the airports.csv file, an airport object was created to store the name, city, country, iata code, latitude, and longitude of that airport. Since the user input file would contain lines of a city and country, it seemed logical to be able to relate each city and country directly to their corresponding airport objects. To do this, a map was created that had string vectors as keys and airports as values. Each entry in the map was a vector containing the city, country, and iata code of a particular airport and the values were the airport objects associated with the key. The functions that created the airport objects and stored them in the map were all contained in an Airport Class

The next step was to obtain connected airports and the flights that connected them. The routes.csv file contained a destination airport that could be reached for each given source airport, along with the airline code of the flight that makes that route. A new map, named AirportRoutesMap, stored the iata codes for each source airport within the file as a key, and its values were a vector containing all possible destination airports that could be reached from that key. A third map, AirlineRoutesMap, was created to store the airlines that went from each source airport to a particular destination. This map had vector keys that stored a source and destination iata code, and the number of stops a flight makes between the two locations. Its values were the list of the airline codes of all flights that went between the airports in a particular key. The AirportsRoutesMap acted as an adjacency list that was used in the breadth-first searching algorithm that was implemented. Each key in the map was the parent to all of the elements in its values, its successors. The solution path returned by the search is defined by a parent-child map, where each key is a child, and its value is its parent. Each of these functions was defined in a Routes Class.

The ReadWrite class contains methods for reading the input file and writing to the output file. The user’s text file containing the start and destination country and city is passed by its file name into the main method. The associated iata code of all airports within the city and country specified are obtained from the Airports map and stored in vectors, and the search algorithm attempts to find a route between each start and destination within the vectors. The total distance covered by the path returned for each start and destination is calculated using the haversine formula. The associated path and its haversine distance are then stored into a temporary map. The path with the lowest haversine distance is then returned to the file writing function to be written into an output file.