The C++ version of Project 1 is built on the Java version. What changed was the addition of the Path class which enables us to return a flight path once we find a valid route between the source and destination cities. The Path class has the path sequence (a vector of string AirportIds) and path cost (a double in km) as instance variables. It also has the create string method, which essentially forms the data to be written to the output file. It does this by looping through the airport IDs in the airport sequence, getting the airline code airport code of the next airport in the sequence, and creating a string out of all the data.

There is also the node class which keeps track of the airports during the search. The node class has a state (an airportId), a parent (the node object from where we arrive at this node), and a path cost (number of flights from the source airport to the current airport). The node class has the path method which when called on a node returns the sequence of airports from the source to the current node. This is to help backtrack the sequence of airports visited to get to the destination.

Finally, there is the algo class which defines the breadth-first search algorithm, unlike the uniform cost search in the Java project, to be used for the problem. The breadth-first search method defined takes the source airport Id, the destination airport Id, and the route map. It makes use of a deque for the open list (frontier) and an unordered set for the closed list (explored set). These two variables keep track of the airport we're yet to explore (openList) and the airports we have already explored (closed list). Each time we generate possible destination airports, we create node objects out of them, perform a goal test to check if it's the destination airport, and if not, we add it to the frontier. Else, we call the path method defined in the node class and return from the search function. This process is repeated until there are either no items on the open list to be explored or the destination airport is found.

The main class makes use of all the methods defined in the other classes to perform the search. It first uses the readFiles class to read the airport and route data and store them in vectors. We used vectors because they are stored in continuous memory, and the items in the vector contain only the values, unlike the list data structure, which stores both the value and a pointer to the next item in the list. We also used the readfiles class to read the user's input file. Then, we got all the possible source and destination airports in the city and country provided in the input file. These airports were then passed as arguments to the breadth-first search method to perform the search, and the data returned is constructed into a string by the Path class, to be written to the provided file.