CS225 Final Project Goals

1. DataSet:
    a. Open Flight route data set
        i. Routes.dat
        ii. format:

Airline	2-letter (IATA) or 3-letter (ICAO) code of the airline.
Airline ID	Unique OpenFlights identifier for airline (see Airline).
Source airport	3-letter (IATA) or 4-letter (ICAO) code of the source airport.
Source airport ID	Unique OpenFlights identifier for source airport (see Airport)
Destination airport	3-letter (IATA) or 4-letter (ICAO) code of the destination airport.
Destination airport ID	Unique OpenFlights identifier for destination airport (see Airport)
Codeshare	"Y" if this flight is a codeshare (that is, not operated by *Airline*, but another carrier), empty otherwise.
Stops	Number of stops on this flight ("0" for direct)
Equipment	3-letter codes for plane type(s) generally used on this flight, separated by spaces

The data is UTF-8 encoded. The special value **\N** is used for "NULL" to indicate that no value is available, and is understood automatically by MySQL if imported.

Notes:

- Routes are directional: if an airline operates services from A to B and from B to A, both A-B and B-A are listed separately.
- Routes where one carrier operates both its own and codeshare flights are listed only once.

Sample entries

```
BA,1355,SIN,3316,LHR,507,,0,744 777
BA,1355,SIN,3316,MEL,3339,Y,0,744
TOM,5013,ACE,1055,BFS,465,,0,320
```

        iii. https://openflights.org/data.html
2. Traversals:
    a. BFS
        i. https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/
3. Covered Algorithms:
    a. Dijkstra's Algorithm
        i. https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/
4. Complex or uncovered options:
    a. Astar Algorithm
        i. https://en.wikipedia.org/wiki/A*_search_algorithm

We are going to be using the Open Flights Route database to make a graph data structure with the given data. With the graph of the route database, we will be running a Breadth-First-Search traversal, Dijkstra's algorithm to find the shortest path between two nodes, and also running the Astar algorithm to also find the shortest and best path given a start and end node. Using our graph structure and a BFS traversal, we want to simulate the spread of a global pandemic using different nodes as starting points and output an array of the traversal. By doing this, we can compare and contrast the reach of a pandemic that started at different points. Using both Astar and Dijkstra's algorithm, we will find the shortest path between two nodes. This is convenient for someone looking for the shortest route between two airports. Overall, our main goals are to find the shortest route from destination to destination, and visualize how a pandemic would spread using BFS.