CSCI 3005 – Programming Assignment 3 – Spring 2021

Your task is to use the breadth-first search algorithm studied in class to implement a solution to the problem of finding the best general route when traveling between states or countries. In developing your solution, you are to use the Graph.java class provided by the instructor.

Data for your program will be obtained from text files in which each line contains the names of two vertices corresponding to an edge in a general, unweighted graph (for instance, see contiguous-usa.dat for edges corresponding to adjacent states in the USA).

Your solution is to be implemented as a class named `RoadTrip` containing the following public methods:

`RoadTrip (String filename)`: a constructor to read in the data from a text file which contains a series of lines. Each line has the names of two graph vertices separated by a single space.

`String getPath(String source, String destination)`: returns information regarding the shortest road trip (involving the smallest number of vertices) from source to destination. The returned string will contain up to three lines of output, as shown below

```
There is a path from <source> to <destination>      // Required for C credit
The shortest path requires crossing <n> borders     // Required for B credit
Path: [source -> vertex -> vertex -> destination]   // Required for A credit
```

If it is not possible to travel from the source to the destination, the method should return a single line of the form:

```
There is no path from <source> to <destination>
```

The **RoadTripTest.java** program and sample text files are available for partial testing. Submit your RoadTrip.java (and any other .java files developed as part of your solution) to Mimir for testing. You should submit neither Graph.java nor any data files nor any .class files when testing your solution.

The Graph.java class provided implements a graph using an adjacency list (where sets are used to store adjacent vertices). For instance, the method calls below would be used to represent the graph in the figure.

```
Graph g = new Graph();
g.addVertex(25);
g.addVertex(10);
g.addVertex(50);
g.addEdge(25, 50);
g.addEdge(25, 10);
System.out.println(g);
```