Class Name: MapQuestEmulator

Class Function: Simulates a ‘MapQuest’ program, where the user can input/remove cities, connected cities, and distance values. The user may also calculate the minimum distance from one city to another.

Class Description:

1. Cities can be added and removed by String city name
2. Connections between cities can be created by String city name, String connection name, and floating point mileage value
3. A city can be checked to see if it has a connection to another specified city
4. The shortest path from one arbitrary city to another can be calculated

Methods:

* public MapQuestEmulator()

An empty MapQuest instance is created.

* public boolean addCity(String cityName)

Creates and adds a new city with the specified name to the MapQuestEmulator, if it already does not exists, and returns true. If the city already exists, false is returned.

* public boolean removeCity(String cityName)

Attempts to remove the city with the specified cityName from the MapQuestEmulator. If successful, true is returned. If not successful, false is returned.

* public boolean addConnectionToCity(String city, String connection, float distance)

Adds a connection from one city name to another city name with a floating point distance value. If the add was successful, true is returned, otherwise false is returned.

* public boolean delConnectionFromCity(String city, String connection)

Removes a connection from one city to another, both ways. If the deletion was successful, true is returned, else false is returned.

* public boolean cityHasConnection(String city, String connection)

Checks if the first parameter city has a connection to the second parameter city. Returns true if it does, false if not.

* public String[][] getMinimumDistanceBetweenCities(String city1, String city2)

Calculates the shortest path from city1 to city2. The resulting minimum distance route is stored in a 2-dimensional array as a list of tuples, stored by city, connection, distance.

Each row, or tuple, shows each city traveled to along the path and the distance traveled from the previous city.

The final row of the array stores city1’s name, city2’s name, and the total minimum distance.

* public CityList getCityList()

Returns a list of cities that are loaded into the MapQuestEmulator.

Test Case:

1. Instantiate a new weighted city graph
2. Check to make sure that it is empty
3. Load an arbitrary number of cities and connections into the graph via a CSV file, while saving a representation of it into an array, for comparison purposes
4. For each city in the graph, compare each one to the array entry, to ensure that each city was added correctly
5. Using the getCityList() method, output each city name
6. Verify that all of the city connections were added correctly with cityHasConnection()
7. Test the getMinimumDistanceBetweenCities() method with known shortest routes
8. Delete all connections from all cities in the graph using the delConnectionFromCity() method
9. Delete all cities from the graph using the removeCity() method
10. Verify that the graph is now empty again