CSCI 2270 – Data Structures and Algorithms
Instructor: Hoenigman
Lab 5
Due: Tuesday, July 15 by 7pm

Build your own communications network - continued

In this assignment, you're going to add additional functionality to the communications network from lab 4. In that lab, you set up a singly linked list to send a message from Los Angeles to Boston, passing through several other cities along the way.

In today's lab, you will add functionality to remove a city from the network and add a city to the network. After adding and/or removing cities, you still need to be able to send the message from beginning to end of the network in either direction.

Include the following cities in your network:
Los Angeles
Phoenix
Denver
Dallas
St. Louis
Chicago
Atlanta
Washington, D.C.
New York
Boston

Implement each city as a class. If you didn't get yesterday's lab working correctly, you are welcome to use the solution posted on Moodle as the starting point for this lab. You could also create a LinkedList class and a City class and set up your LinkedList to include cities.

Your cities need to have a private name variable, and have a forward **and a reverse** path connecting each of them with the next and previous city in the network, and a place to store the message being sent. (You can assume the message is a string.)

There is a file on Moodle called *messageIn.txt* that contains one line of text with words separated by spaces. Your program needs to transmit each word one at a time from Los Angeles to Boston, going through each city in the linked list. When a word is received in Boston, it should be printed to the terminal and stored in another file, a linked list, or an array. Once all words have been received in Boston, turn around and send the message back to Los Angeles. When all the words have been received in Los Angeles, print the entire message to the terminal, along with **the number of operation** needed to send the message back and forth.

Sending the message back and forth will verify that you have your cities set up correctly before beginning this next step.

A snowstorm smashes Denver
It's winter and a mighty upslope storm has smashed into Denver, knocking out power for miles. All communication needs to be re-routed to bypass Denver. Your program needs to search the list, find the city of Denver, reset the pointers before and after it, and then delete the city.

Add a new city
Next, ask the user to add a city. They will need to provide the name of the city and which city in the current network should be before the new city. Add the new city to the network and reset your pointers accordingly. Resend the entire message, back and forth, making sure to print at the end the message as transmitted and the number of operations needed.

Perform garbage collection and end the program.

Implementation details
Transmitting the message between cities needs to be done in a separate function called *transmitMsg*. If you are using classes, *transmitMsg* should be part of your class. If you are using structs, it will be a function. When you read a word from the file, call the *transmitMsg* function and assign the word to the first city. You then need to then call the *transmitMsg* function each time you send the word to the next city from the city that currently holds it. Keep in mind that transmitting a message means that only one city can hold the message/word at one time. As soon as the message is transmitted to the next city, the message should be removed from the sender city.

Actions to remove and insert a city need to be handled in a function. Resetting the previous and next pointers should be part of the class.

In this Lab you need to (once again) count the **number of operations.** To make it simpler, every time you modify one of the attributes of your City should count as +1 operations.

You need to create a class and a header file for your city and include the header file in your program. To submit your work, zip all three files together and submit the zip file. All files should include a comment block at the top of the file with your name, instructor's name, and lab number.