CSCI 2270 – Data Structures - Section 100

Instructor: Shayon Gupta

Assignment 3, Sep 2018

Linked List

Communication Between Towers

OBJECTIVES

1. Create, traverse, add, and delete from a linked list.

Background

In the Lord of the Rings trilogy, there is a scene where a warning beacon is lit in the towers of Minas Tirith, which is seen by a second beacon, prompting them to light their own fire which a third beacon sees, and so forth. This was a rapid means of communication in the days before telegraphs were invented. In this assignment, you're going to simulate a communications network using a linked list. Each node in the list will represent a city and you need to be able to send a message between nodes from one side of the country to the other. Your program also needs to allow cities to be added to the list.

Building your own communications network

You will be building off some starter code. This code is available on Moodle. It implements the general structure of your code, although you will have to fill in many of the functions. Your list will be implemented using the following struct:

```
struct city {
  string name; // name of the city
  city *next; // pointer to the next city
  int numberMessages; // how many messages passed through this city
  string message; // message we are sending accross
}
```

CSCI 2270 – Data Structures - Section 100

Instructor: Shayon Gupta

Assignment 3, Sep 2018

Specifics

- Use the starter code provided. Don't change sections marked "DO NOT MODIFY".
- You shouldn't need to `#include` any additional code, or write any additional functions.
- Your code should be readable, efficient, and accomplish the task provided.

Your program will start by displaying a menu. The user will select an option from the menu to decide what the program will do, after which the menu will be displayed again. This is built in to the starter code, so you won't have to add it yourself. The specifics of each menu option are displayed below. Many of the options call multiple functions and use code written for other options, so the order that you implement them may make it easier to test your answers. You may want to implement just a part of one option to make it easier to implement other options.

Option 1: Build Network

This option calls the `loadDefaultSetup` function, which you must write yourself. This function should clear the existing list if it exists using the `deleteEntireNetwork` function (see option 6), then builds a new linked list using six default cities in this order:

1. Los Angeles
2. Phoenix
3. Denver
4. Dallas
5. Atlanta
6. New York

It should create these cities by using the `addCity` function (see option 4). After creating all six linked list nodes, the program will print out the list by calling the `printPath` function - this function call is built into the starter code already, so all you have to do is write `printPath` (see option 2).

Option 2: Print Network Path

This option calls the `printPath` function, which you will need to write. It should print out the name of each city in the linked list in order from the head to the tail, finishing with "NULL". This function will be your primary debugging tool to make sure you can insert cities to the list correctly. Each city should be separated by an arrow. For example, the output when printing the default list created by option (1) should be:

```
== CURRENT PATH ==
Los Angeles -> Phoenix -> Denver -> Dallas -> Atlanta -> New York -> NULL
===
```

CSCI 2270 – Data Structures - Section 100

Instructor: Shayon Gupta

Assignment 3, Sep 2018

Option 3: Transmit Message

This option reads in two inputs from the user: a message to send, and a city to receive the message. It then passes that information to the `transmitMsg` function, which you will write. That function should 'transmit' the message starting at the beginning of the network to the specified city by printing out each city that receives the message. For each of those cities, it should also increase the number of messages that city has received by 1 and print that as well.

The format of this printing should be:

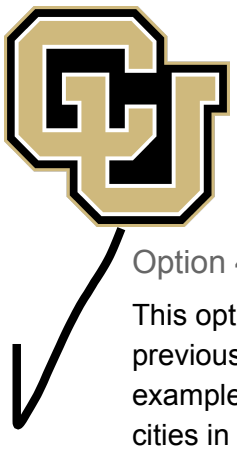
```
<CITY NAME> [# messages passed: <NUMBER OF MESSAGES>] received: <MESSAGE>
```

For example, the following should be the output if the linked list contains the default cities from option (1) and the message "The Broncos game is a blast!" is sent to "Denver":

```
Los Angeles [# messages passed: 1] received: The Broncos game is a blast!
Phoenix [# messages passed: 1] received: The Broncos game is a blast!
Denver [# messages passed: 1] received: The Broncos game is a blast!
```

If the user then sends the message "Moving to the east coast." to "New York", the output will be:

```
Los Angeles [# messages passed: 2] received: Moving to the east coast.
Phoenix [# messages passed: 2] received: Moving to the east coast.
Denver [# messages passed: 2] received: Moving to the east coast.
Dallas [# messages passed: 1] received: Moving to the east coast.
Atlanta [# messages passed: 1] received: Moving to the east coast.
New York [# messages passed: 1] received: Moving to the east coast.
```



CSCI 2270 – Data Structures - Section 100

Instructor: Shayon Gupta

Assignment 3, Sep 2018

Option 4: Add City

This option prompts the user for the name of a new city to add to the network, as well as a previous city name - that is, the name of whichever city should come before the new one. For example, if the user wants to add Tucson after Phoenix in the default network, then the first four cities in the network would be:

```
Los Angeles -> Phoenix -> Tucson -> Denver…
```

If the user wants to add the new city to the head of the network then they should enter 'First' instead of a previous city name. The starter code will get a pointer to the previous city by passing that city's name to the `searchNetwork` function, which you will have to write. That function should search through the whole network for a given name, then return a pointer to the city with that name. The starter code will then pass that pointer to the `addCity` function, which you will also write. This should just add the city to the list. If the previous city is not 'first', it should also print out a message to the user in the form:

```
prev: <PREVIOUS CITY NAME> new: <NEW CITY NAME>
```

For example, if the user wants to add Washington D.C. after Atlanta it should print:

```
prev: Atlanta new: Washington D.C.
```

Note: if the previous city was 'First', it will pass NULL to `addCity` rather than a pointer to an existing city. It also shouldn't print anything.

Finally, this option will call `printPath` (see option 2).

Option 5: Delete City

This option prompts the user for a city name, then passes that name to the `deleteCity` function which you will write. This function should delete the city while keeping the linked list intact, no matter where the deleted city is. You will have to handle all possible locations where a city could be deleted. The `deleteCity` function shouldn't print anything, but after it runs the starter code will display the new path by calling `printPath` (see option 2).

CSCI 2270 – Data Structures - Section 100

Instructor: Shayon Gupta

Assignment 3, Sep 2018

Option 6: Clear network

This option calls the `deleteEntireNetwork` function, which should delete all cities in the network. It should print out in the following format:

```
deleting: <1st CITY NAME>
deleting: <2nd CITY NAME>
…
deleting: <LAST CITY NAME>
Deleted network
```

For example, deleting the default network should print:

```
deleting: Los Angeles
deleting: Phoenix
deleting: Denver
deleting: Dallas
deleting: Atlanta
deleting: New York
Deleted network
```

Option 7: Quit

This option exits the program.

Submitting your code:

Submit your code on Moodle by following the directions on Assignment 3 Submit. You must submit an answer to be eligible for interview grading.