CSCI 2270 – Data Structures

Instructors: Maciej Zagrodzki, Christopher Godley

Assignment 4 - Linked Lists

Communication Between Towers

OBJECTIVES

1. **Delete, reverse, and rotate in a linked list**
2. **Get practice implementing classes**

This assignment is an extension of assignment 3.

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 country and you need to be able to send a message between nodes from one side of the world to the other.

Building your own communications network

You will be implementing a class to simulate a linear communication network between countries. There are three files in Moodle containing a code skeleton to get you started. *Do not modify the header file or your code won't work in Moodle!* You will have to complete both the class implementation in CountryNetwork.cpp and the driver file main.cpp.

The linked-list itself will be implemented using the following struct (already included in the header file):

```
struct Country
{
    string name;          // name of the country
    string message;       // message this country has received
    int numberMessages;   // no. of messages passed through this country
    Country *next;        // pointer to the next country
};
```

CSCI 2270 – Data Structures

Instructors: Maciej Zagrodzki, Christopher Godley

Class Specifications

The **CountryNetwork** class definition is provided in the file *CountryNetwork.hpp* in Moodle. *Do not modify this file or your code won't work on Moodle!* Fill in the file *CountryNetwork.cpp* according to the following specifications.

Country* head;

- ➔ Points to the first node in the linked list

CountryNetwork();

- ➔ Class constructor; set the head pointer to NULL

bool isEmpty();

- ➔ Return true if the head is NULL, false otherwise

void insertCountry(Country* previous, string countryName); *// Beware of edge cases*

- ➔ Insert a new country with name **countryName** in the linked list after the country pointed to by **previous**. If **previous** is NULL, then add the new country to the beginning of the list. Print the name of the country you added according to the following format:

```
// If you are adding at the beginning use this:
cout << "adding: " << countryName << " (HEAD)" << endl;

// Otherwise use this:
cout << "adding: " << countryName << " (prev: " << previous->name <<
")" << endl;
```

void deleteCountry(string countryName); *// Beware of edge cases*

- ➔ Traverse the list to find the node with name **countryName**, then delete it. If there is no node with name **countryName**, print *"Country does not exist."*

void loadDefaultSetup();

- ➔ First, delete whatever is in the linked list using the member function **deleteEntireNetwork**. Then add the following six countries, in order, to the network with **insertCountry**: "United States", "Canada", "Brazil", "India", "China", "Australia"

Country* searchNetwork(string countryName);

- ➔ Return a pointer to the node with name **countryName**. If **countryName** cannot be found, return NULL

void deleteEntireNetwork();

- ➔ If the list is empty, do nothing and return. Otherwise, delete every node in the linked list and set **head** to NULL. Print the name of each node as you are deleting it according to the following format:

```
cout << "deleting: " << node->name << endl;
```

After the entire linked list is deleted, print:

```
cout << "Deleted network" << endl;
```

void rotateNetwork(int n);

- ➔ Manipulate ***next** pointers to rotate the linked list. Here, ***n*** is the number of nodes from the beginning of linked list that you need to move to the end of the linked list. For example, if you have linked list like this: **"A -> B -> C -> D -> NULL"**, and ***n=3***, then the linked list after rotateNetwork should be "**D -> A -> B -> C -> NULL**". Here, **"D"** is the new head which points to "A".
  If you have linked list like this: **"A -> B -> C -> D -> NULL"**, and ***n=1***, then the linked list after rotateNetwork should be "**B -> C -> D -> A -> NULL**". Here, "B" is the new head which points to "C".
- ➔ If the linked list is empty, print "*Linked List is Empty*".
- ➔ If ***n*** is bigger than the number of nodes in the linked list or smaller than ***1***, then print "*Rotate not possible*".
- ➔ Otherwise print "*Rotate Complete*".
- ➔ Note: Rotation is anti-clockwise
  [NOTE: Change the order of the "node", not the "value of the node"]

void reverseEntireNetwork();

- ➔ Manipulate ***next** pointers to reverse the order of the linked list. For example, reversing the following list with **"A"** at the head: **"A -> B -> C -> D -> NULL"**, should yield **"D -> C -> B -> A -> NULL"** with **"D"** as the new head
  [NOTE: Change the order of the "node", not the "value of the node"]

void printPath();

- ➔ Print the names of each node in the linked list. Below is an example of correct output using the default setup. (Note that you will **cout << "NULL"** at the end of the path)

```
== CURRENT PATH ==
United States -> Canada -> Brazil -> India -> China -> Australia -> NULL
===
```

➔ If the network is empty then print *"nothing in path"*

Main driver file

Your program will start by displaying a menu by calling the **displayMenu** function included in main.cpp. The user will select an option from the menu to decide what the program will do, after which, the menu will be displayed again. The specifics of each menu option are described below.

Option 1: Build Network

This option calls the **loadDefaultSetup** function, then calls the **printPath** function. You should get the following output:

```
adding: United States (HEAD)
adding: Canada (prev: United States)
adding: Brazil (prev: Canada)
adding: India (prev: Brazil)
adding: China (prev: India)
adding: Australia (prev: China)
== CURRENT PATH ==
United States -> Canada -> Brazil -> India -> China -> Australia -> NULL
===
```

Option 2: Print Network Path

Calls the **printPath** function. Output should be in the format below:

```
// Output for the default setup
== CURRENT PATH ==
United States -> Canada -> Brazil -> India -> China -> Australia -> NULL
===

// Output when the linked list is empty
== CURRENT PATH ==
nothing in path
===
```

CSCI 2270 – Data Structures

Instructors: Maciej Zagrodzki, Christopher Godley

Option 3: Add Country

Prompt the user for two inputs: the name of a new country to add to the network, **newCountry**, and the name of a country already in the network, **previous**, which will precede the new country. Use the member functions **searchNetwork** and **insertCountry** to insert **newCountry** into the linked-list right after **previous**.

- If the user wants to add the new country to the head of the network then they should enter “First” instead of a previous country name.
- If the user enters an invalid previous city (not present in the linked list), then you need to prompt the user with the following error message and collect input again until they enter a valid previous country name or “First”:

```
cout << "INVALID country...Please enter a VALID previous country
name:" << endl;
```

- Once a valid previous country name is passed and the new country is added, call the function **printPath** to demonstrate the new linked-list.

For example, the following should be the output if the linked-list contains the default setup from option (1) and the user wants to add Colombia after Brazil:

```
Enter a new country name:
Colombia
Enter the previous country name (or First):
Brazil

adding: Colombia (prev: Brazil)
== CURRENT PATH ==
United States -> Canada -> Brazil -> Colombia -> India -> China -> Australia ->
NULL
===
```

Option 4: Delete Country

Prompt the user for a country name, then pass that name to the **deleteCountry** function and call **printPath** to demonstrate the new linked-list.

For example, the following should be the output if the linked-list contains the default setup from option (1) and the user wants to delete Canada:

```
Enter a country name:
```

```
Canada
== CURRENT PATH ==
United States -> Brazil -> India -> China -> Australia -> NULL
===
```

Option 5: Reverse network

Call the **reverseEntireNetwork** function then the **printPath** function.

For example, the following should be the output if the linked-list contains the default setup from option (1):

```
== CURRENT PATH ==
Australia -> China -> India -> Brazil -> Canada -> United States -> NULL
===
```

Option 6: Rotate Network

Call the **rotateNetwork** function with **n=3** then the **printPath** function.

For example, the following should be the output if the linked-list contains the default setup from option (1):

```
== CURRENT PATH ==
India -> China -> Australia -> United States -> Canada -> Brazil -> NULL
===
```

Option 7: Clear network

Call the **deleteEntireNetwork** function. For example, deleting the default network should print:

```
Network before deletion
== CURRENT PATH ==
United States -> Canada -> Brazil -> India -> China -> Australia -> NULL
===
deleting: United States
deleting: Canada
deleting: Brazil
deleting: India
deleting: China
deleting: Australia
Deleted network
```

```
Network after deletion
== CURRENT PATH ==
nothing in path
===
```

Option 8: Quit

Print the following message:

```
cout << "Quitting... cleaning up path: " << endl;
```

Then call **printPath**, followed by **deleteEntireNetwork**. Now, check if the network is empty using **isEmpty**. If it is, print:

```
cout << "Path cleaned" << endl
```

Otherwise, print:

```
cout << "Error: Path NOT cleaned" << endl;
```

Finally, print the following before exiting the program:

```
cout << "Goodbye!" << endl;
```

Example run

```
Select a numerical option:
+=====Main Menu=========+
 1. Build Network
 2. Print Network Path
 3. Add Country
 4. Delete Country
 5. Reverse Network
 6. Rotate Network
 7. Clear Network
 8. Quit
+----------------------+
```

```
#> 1
adding: United States (HEAD)
adding: Canada (prev: United States)
adding: Brazil (prev: Canada)
adding: India (prev: Brazil)
adding: China (prev: India)
adding: Australia (prev: China)
== CURRENT PATH ==
United States -> Canada -> Brazil -> India -> China -> Australia -> NULL
===
Select a numerical option:
+=====Main Menu=========+
 1. Build Network
 2. Print Network Path
 3. Add Country
 4. Delete Country
 5. Reverse Network
 6. Rotate Network
 7. Clear Network
 8. Quit
+----------------------+
#> 3
Enter a new country name:
Iran
Enter the previous country name (or First):
Canada

adding: Iran (prev: Canada)
== CURRENT PATH ==
United States -> Canada -> Iran -> Brazil -> India -> China -> Australia ->
NULL
===

Select a numerical option:
+=====Main Menu=========+
 1. Build Network
 2. Print Network Path
 3. Add Country
 4. Delete Country
 5. Reverse Network
```

```
 6. Rotate Network
 7. Clear Network
 8. Quit
+----------------------+
#> 4
Enter a country name:
Australia
== CURRENT PATH ==
United States -> Canada -> Iran -> Brazil -> India -> China -> NULL
===

Select a numerical option:
+=====Main Menu=========+
 1. Build Network
 2. Print Network Path
 3. Add Country
 4. Delete Country
 5. Reverse Network
 6. Rotate Network
 7. Clear Network
 8. Quit
+----------------------+
#> 5
== CURRENT PATH ==
China -> India -> Brazil -> Iran -> Canada -> United States -> NULL
===

Select a numerical option:
+=====Main Menu=========+
 1. Build Network
 2. Print Network Path
 3. Add Country
 4. Delete Country
 5. Reverse Network
 6. Rotate Network
 7. Clear Network
 8. Quit
+----------------------+

#> 6
```

```
Enter the count of values to be rotated:
2
Rotate Complete
== CURRENT PATH ==
Brazil -> Iran -> Canada -> United States -> China -> India -> NULL
===

Select a numerical option:
+=====Main Menu=========+
 1. Build Network
 2. Print Network Path
 3. Add Country
 4. Delete Country
 5. Reverse Network
 6. Rotate Network
 7. Clear Network
 8. Quit
+-----------------------+

#> 7
Network before deletion
== CURRENT PATH ==
Brazil -> Iran -> Canada -> United States -> China -> India -> NULL
===
deleting: China
deleting: India
deleting: Brazil
deleting: Iran
deleting: Canada
deleting: United States
Deleted network
Network after deletion
== CURRENT PATH ==
nothing in path
===

Select a numerical option:
+=====Main Menu=========+
 1. Build Network
 2. Print Network Path
```

```
 3. Add Country
 4. Delete Country
 5. Reverse Network
 6. Rotate Network
 7. Clear Network
 8. Quit
+----------------------+
#> 8
Quitting... cleaning up path:
== CURRENT PATH ==
nothing in path
===
Path cleaned
Goodbye!
```