Lab 10

Due Date: Nov 13 , 2019
Total Points: 15 points

The purpose of this lab is to practice working with Linked Lists

---

In this lab, you are going to design a linked list to keep track of a telephone directory. You need to design two classes:

- Class *Node* has the following **private** data:

```
string name
string phoneNumber;
```

- Class *LL* has the following **private** data:

```
Node * head; // head pointer points to beginning of linked list
```

Class *LL* should be made friend of class *Node* so public functions of *LL* can access a node's private attributes without having to write getters and setters for class *Node*.

You also need to implement the following public methods for class *LL*:

- `LL::LL()`
The default constructor sets the head to nullptr

- `void LL::append(string pName, string phone)`
This function creates a node, sets its data to the values sent as arguments, and inserts the new node at the end of the list (after the last node)

- `void LL::print()`
This function traverses the list printing the data in each node

- `void LL::insertAtBegin(string pName, string phone)`
This function creates a node, sets its data to the values sent as arguments, and inserts the new node at the beginning of the linked list (before the first node)

- `void LL::searchByName(string pName)`
This function searches for a particular person's record, whose name is pName in the list and prints the person's corresponding phone number if it exists. If that person does not exist, the function prints an error message

- `LL::~LL()`

The destructor destroys (deletes) all nodes in the list and sets head back to nullptr. Have the destructor call another private function destroy that does the deleting (you need to implement that function too:

```
void LL:destroy()
```

- `LL::LL(const LL& source)`

The copy constructor performs a deep copy of the list source

- `bool LL::operator == (const LL & L1)`

The overloaded operator == returns true if two lists are the same, false otherwise. In other words, if both lists have the same number of nodes and the same node values (in the same order)

When done, write a main program to test your functions. In your main, you need to:

- create an empty list, list1
- call the function insertAtBegin to add a node with the values: "Nancy Garcia", "617-227-5454"
- call the function append to add a node with the values: "Adam James", "202-872-1010"
- call the function append to add a node with the values: "Jim Meyer", "337-465-2345"
- call the function insertAtBegin to add a node with the values: "Joe Didier", "352-654-1983"
- call the function append to add a node with the values: "Mayssaa Najjar", "878-635-1234"
- call the function print to display all nodes in list1
- call the function searchByName 3 times to search for the following persons: "Nancy Garcia", "Mayssaa Najjar", "Jamie Garcia"
- instantiate another list, list2, using the copy constructor so that list2 is a copy of list1
- use the overloaded == operator to check if list1 and list2 hold the same set of data. If they do, print a message that they are the same and call the function print to display the data of either one of them. Otherwise, display a message that they are different and print both lists
- modify list1 by adding an entry at the beginning with the values: "Michel Delgado", "327-165-2345"
- check if list1 and list2 hold are the same. If they are, print a message that they are the same. Otherwise, display a message that they are different and print both lists