Due Date: November 3, 2017

Part-I [20 points]

You are asked to implement a **Stack** using `LinkedList` data structure. The header file is called LinkedStack.h. A partial implementation of `LinkedStack` template class is given below, please use this template as it is, don't modify. You are asked to implement necessary functions required for a stack abstract data structure.

```
//LinkedStack.h

template <class T>
struct node{
    T data;
    node<T>* next;
};

template <class T>
class LinkedStack{
      Node<T> *top;
   public:
      //provide function prototypes for a Stack here



};
//provide function implementations here
```

Part-II [40 points]

Benny the Barber owns a one-chair shop. They told Benny that customers are not happy with waiting in lines and not get processed in order they arrived. As computer scientist, you are going to help Benny to write a C++ program to manage waiting lines and to process customers in the same order they arrive.

Here are the functional requirements of the program:

- The program shall allow to add a customer to waiting line
- The program will guarantee that customers will be processed in the same order they arrive
- When the Barber wants to serve to next customer, the program shall give the name of next customer in line

You are given the test application source file below, App.cpp, and you are **not allowed to modify** it. Use it as it is.

//App.cpp: test application
```
int main(){
	BarberShop shop;
	Customer customer1("MARK","KILGORE");
	Customer customer2("RICK","GRIMM");
	shop.addCustomer(customer1);
	shop.addCustomer(customer2);
	Customer nextCustomer = shop.nextCustomer();
	cout<<nextCustomer.getName()<<" is served next"<<endl;
	Customer customer3("JILL","WOLFF");
	shop.addCustomer(customer3);
	nextCustomer = shop.nextCustomer();
	cout<<nextCustomer.getName()<<" is served next"<<endl;
	nextCustomer = shop.nextCustomer();
	cout<<nextCustomer.getName()<<" is served next"<<endl;
}
```

- You are also given partial implementation of BarberShop class below. You are asked to use two stacks to implement your program, and you cannot use any other data structure. Also, you are not allowed to delete any statement from this file. You'll write your code in the lines indicated by "write your code here"
- You may create new source and header files as needed.
- You are asked to use the stack header file, `LinkedStack.h`, that you created in Part-I.

//BarberShop.cpp
```
#include <iostream>
#include "LinkedStack.h"
using namespace std;

class BarberShop{
     LinkedStack<Customer> s1;
     LinkedStack<Customer> s2;
     public:
          void addCustomer(Customer&);
          Customer nextCustomer();
};
void addCustomer(Customer& customer){
        //write your code here


}
Customer nextCustomer(){
        //write your code here

}
```

Part-III [30 points]

Please write a C++ program that reads two very large numbers from the user as two string inputs, and add these two large numbers. You can only use **Stack** data structure to perform addition. You are asked to use the `LinkedStack.h` implementation you have created in Part I.

```
Sample Run-1:
Enter a very large number: 18765432171212345356785343423
Enter a very large number: 82923245245235423412345234445
sum= 101688677416447768769130577868
```

Hint:

- You may need to create 3 stacks: first stack will hold the digits of the first number, second stack will hold the digits of the second number, and the third stack will store the digits of the sum.
- You can convert a character ch into an integer i as follows:
  `i = ch-'0' ;` *//can you have a guess why?*

Part-IV [10 points]

Write a C++ program which reads a positive number from the user and displays the decimal digits of an integer in reverse order. You are supposed to use **recursion**. The function prototype is given below.

```
void reverseDigits(int);
```

Sample Run:

```
Enter a number: 183459
954381
```

How to submit:

Please follow the following steps for each assignment:

1. Create a separate repository for each question in GitHub, named **assignment3_1, assignment3_2, assignment3_3 assignment3_4**
2. **Clone** each repository to your local computer.
3. Modify the files and **commit** changes to complete your solution in your local repository.
4. **Push**/sync the changes up to GitHub.
5. To turn in the assignment, send the link for each repository as an email to your instructor (fatma.serce@..)