CSE 20312 Data Structures
Spring 2017

Lab 3 handout for Week of February 6

Objectives

1. Use a sequence container data structure to add list-based integers
2. Introduce yourself to templated classes and self-referential objects (linked list nodes)
3. Have fun!

High-level problem description

Note we will continue to take attendance at the start of lab... please be on time!

As you may already know, integer values in C and C++ are limited by the number of bits used to represent that data. For instance, a 64-bit unsigned integer has the maximum value of $2^64 - 1$ or 18446744073709551615.

To represent integers of arbitrary length, we can use a singly linked list data structure such that a node in the linked list corresponds to a single digit in the integer.

For instance, the number 123 can be stored as linked-list that looks like the following:

```
```

Note that the first or head node in this list contains the least significant digit (in this case 3), while the last node contains the most significant digit (1).

For this problem, you are to read in pairs of arbitrary length integers into singly linked lists, then use the lists to add the two numbers, and finally output the results.

Input/Output

You will be given a series of integers from standard input in the follow format:

```
integer1 integer2
integer1 integer2
...
integer1 integer2
```

You are to add each pair of integers and then output the resulting total as a single number:

Example

Given the following input:

```
1 1
123 123
1 12
```

Your program should output the following:

```
2
246
13
```
Part I: Creating your data structures – Node and List

Note: Place all your code in a file called LinkedList.cpp. You do not have to create separate files or write your own Makefile for this assignment.

1. Create Node struct or class. Note that in a struct, all members are public by default.
   a. A variable to hold value, for part I this can be int. In Part II, this will be of type T.
   b. A Node* called next, to point to the next node in the linked list.
2. Create List struct (or class) to represent a linked list.
   a. List only needs a Node* called head.
   b. only has Dynamic memory allocation.
3. pushFront() function that adds a number to the front, so that when we add 1, then 2 then 3. It creates linked lists like 3-> 2-> 1-> NULL.
4. print() – prints the contents of the linked list
5. Constructor – initializes head to nullptr
6. Deconstructor – ensure each Node is deleted. You may want to use two different Node* pointers to aid you in walking the list so that you can delete nodes without losing access to the next node in the list.
7. Create a driver main() function to test that when you add things to the linked list, they get added properly. Note: After Part I, remove the code that tests linked list creation and printing. Part II code describes what main should do.

Part II: Templating and I/O cycle

1. Use templating - use : template <typename T>
2. In main() – read in two numbers(as strings) of input from user (eg. cin >> numString1 >> numString2); use each numString to create a linked list, print the contents of linked list, then read in next line... and so on till the input ends (ctrl+d)
   a. HINT: You may wish to read the integers in as std::strings, that way you can splice each digit from the number.
   b. HINT: You can declare the new linked lists inside the while loop for input, that way they get destroyed and created anew in each run of the while loop.

Part III: Adding numbers stored in linked lists

1. Create a function called addNumberLists which accepts two lists by reference and returns a list. The reverse representation of the numbers will help you in adding them together, as you start adding with the lowest place digit.
2. Keep in mind that the input numbers may not be the same size.
3. Remember to calculate the carry
4. Hint: You can use modulo(%) and division(/).
5. Your function signature: List<T> addNumberLists(List<T> &l1, List<T> &l2)
6. Use the addNumberLists() function you just created to add the two numbers read from the terminal and print the output.
7. Test your code: The following files are provided to allow you to test your code – Makefile, input, output and measure.c
   a. make – to make your code.
   b. make test – to run output, memory and timing tests. The memory test should be run on linux machines only. The mac os will give an incorrect output of failed test if the memory test is run there.
Resource files

You will have to create your own LinkedList.cpp file.

The Makefile, input, output and measure.c files are provided to you. To download these resource files, navigate to the directory where you are working on your homework and run the following wget commands. These will download the files into your local directory.

```
```

Rubric

+60  Program implemented as intended
+10  Program uses templating for both Node and List
+10  List should initialize Node* head to nullptr (in constructor or when declaring private member variable)
+5   On receiving ctrl+d, the program does not expect more input.
+10  For each pair or input, program prints some output
+15  addNumberLists() handles unequal length lists
+5   Program prints correct sum for equal length numbers
+5   Good program structure and code style

+20  Program passes memory test
+15  List Deconstructor deletes all nodes OR Node/List each delete their data - no memory leaks

+5   Program passes timing test
+15  Includes the standard lab report