CS 211 – Data Structures
Assignment #2

Deadlines

This assignment is due on **Sunday, October 13 @ 11:59 PM.**

How to submit your work on the assignment:

Assignments will be accepted via Canvas. Use the Multiple File submission tool in Canvas to submit all files. Submit all the files in a single submission, and do NOT use ZIP.

It is expected AND REQUIRED that any code you create and submit (or modify and submit) is your work and your work alone. Copying of any code between students is strictly prohibited!

You may use the code in the supplied **assn2Test.cpp** file to test these problems. You do **NOT** need to turn in the **assn2Test.cpp** file with your assignment. Uncomment lines in **main()** to run tests.

Problem 1 – 30 points

Write a **recursive** function **addCommas** that expects a non-negative integer (the data type to use is **unsigned long long int**, which can hold any integer up to about 20 digits long), returns nothing, and outputs to the screen the number written with commas. For example, **addCommas(9008007006)** will output to the screen **9,008,007,006**. Getting full credit means handling ALL possible input values so that they output correctly – no leading zeroes at the very front, but exactly three digits between commas after that. (NOTE: do NOT try to implement tail recursion for this; it'll only make it more complicated :-).

HINTS: A division operation that stores its result in an **int** will truncate the result.
For example, the statement **int n = 3754 / 1000;** will result in **n** being set to the value **3**.
Also, remember that the **%** operator modulo gives you the remainder of a division. For example, the statement **int r = 3754 % 1000;** will result in **r** being set to the value **754**.

The line **#include <iomanip>** allows use of the following library functions to print a number **n** including leading zeroes when **n** has less than three digits. This code has been tested in CS50 – your IDE may vary!

```
cout << setfill('0') << setw(3) << n;
```

Problem 2 - 70 points

Examine the definition of the provided C++ classes **DoublyLinkedList** and **DLLNode.**

Create and thoroughly test new methods named **addToDLLHead(double el)** and **double deleteFromDLLHead()** that insert and delete from the doubly-linked list at the head, similar to how the existing methods **addToDLLTail(double el)** and **double deleteFromDLLTail()** do at the DLL's tail.

Test the methods thoroughly to make sure they perform correctly in all cases (on empty lists, on lists with only one item, and on lists with multiple items)!

Submit the modified files **assn2.h** and **assn2.cpp** containing the code added for both problems. There is no need to submit the testing file.