CS163 Lab Session #5 – Recursion

Please complete this to become familiar with Recursion (as review).
Submit code to the CS199 D2L dropbox. Limit the time invested to 1 hour and 50 minutes maximum.

Coding: *With this lab we will be working with an existing linear linked list of integers. In this situation, a class is not being used, so the head pointer will be sent in as an argument. Your job will be to implement functions to experience manipulating the lists using* ***recursion! You will be working with code supplied on D2L from the D2L online "locker".***

_____Step 1. **Check to find the requested data (sent in as an argument) is in the list:**

a. Prototype: **bool find(node * head, int match);** //return true if there is a match
b. Plan out the code before writing it using these questions:
    i. What is the simple (base) case? (Also known as the stopping condition)
    ii. What is the increment step to get to the next smaller sub-problem
    iii. What needs to get done before going to that next smaller sub-problem?
    iv. What needs to get done after returning from that smaller sub-problem?
c. Download the .h and .o files from D2L's online locker
d. Compile: g++ *.cpp *.o
e. Run: ./a.out

_____Step 2. **Insert number 9 after each number 2 in the list**

a. Prototype: **void insert_9(node * & head);**
b. Plan out the code before writing it using these questions:
    i. What is the simple (base) case? (Also known as the stopping condition)
    ii. What is the increment step to get to the next smaller sub-problem
    iii. What needs to get done before going to that next smaller sub-problem?
    iv. What needs to get done after returning from that smaller sub-problem?
c. Compile: g++ *.cpp *.o
d. Run: ./a.out

_____Step 3. **Challenge: Display the last two items in the list**

a. Prototype: **void display_last_two(node * head);**
b. Plan out the code before writing it using these questions:
    i. What is the simple (base) case? (Also known as the stopping condition)
    ii. What is the increment step to get to the next smaller sub-problem
    iii. What needs to get done before going to that next smaller sub-problem?
    iv. What needs to get done after returning from that smaller sub-problem?
c. Compile: g++ *.cpp *.o
d. Run: ./a.out

_____Step 4. Write a recursive function to determine if two linear linked lists are of equal length – only traverse as far as necessary (ie., do not traverse through past the length of the smaller of the two!). *(Note: such a function could take a long time to execute using iteration if one list is very long even if the other is very short – because it would first determine the length of both individually).*

a. Prototype: **bool same_length(node * head1, node * head2);**
b. Plan out the code before writing it using these questions:
    i. What is the simple (base) case? (Also known as the stopping condition). Think about the situation where there is just one item.
    ii. Now, think about how you would use a loop for this problem and describe what you would use the loop for *(be specific):*
    iii. Imagine how this loop could be replaced with a recursive call. What would be the base case or stopping condition? What needs to get done before going to that next smaller sub-problem?
    iv. What needs to get done after returning from that smaller sub-problem?
c. Compile: g++ *.cpp *.o
d. Run: ./a.out

_____Step 5. **Develop the test plan for one of the solutions in this lab: (Fill out the shaded boxes)** ***Hint…what are the special cases?***

Test Case(s)	Expected Result
Compare the original two lists	They are the same
Display the last two	display something and then a two
Insert 9 to the first list	Insert 9
Display the last two	Display 2 then a 9
Compare the original two lists	They are different sizes

Verify correctness: Using the above test plan, create a test program that tests the interactions of all functions together.

Self-Assessment: *What could you do to improve for next time?*