CSIEB0100 Data Structures, Fall 2014
Midterm Exam

ID: _____ Dept:_____ Name: _____

1. **(5%)** Given an algorithm that solves a problem in three phases. The first phase takes $O(100n)$ to input the data of size n. The second phase takes $O(n \log n)$ to process the data. The third phase takes $O(\log n)$ to output the data.

(a) What is the complexity of the algorithm?

(b) If the data size is 10, which phase is most likely to take the longest time to execute?

2. **(10%)** For each of the complexity expression below, determine its overall complexity. For example, given expression $2n + 3n$, the overall complexity should be $O(n)$.

(a) $2n^2 - 3n$

(b) $n! + 2^n$

(c) $5n^2 + n \log n$

(d) $n^{1.001} + n \log n$

(e) $5n^3 - 3n^2 \log n + 2n$

3. **(15%)** Given an array of integers **A[0,…,n-1]**, write C++ functions to compute the prefix sum of **A** in the following ways.
   (a) Output the prefix sum to a new array **B**. In other words, **B[0]=A[0]**, **B[1]=A[0]+A[1]**, **B[2]= A[0]+A[1]+A[2]**, etc.
   (b) Save the prefix sum in **A** itself.

4. **(15%)** A *triple-ended queue* is similar to an ordinary queue, except that it allows you to insert and delete on the **front**, **rear** and **middle**.
   (a) Design an ADT to represent a triple-ended queue.
   (b) Write a C++ class to implement the triple-ended queue ADT.

5. **(30%)** Extend the template class `List` discussed in the class with the following functions. You don't need to write out the entire class. Just the definitions of the new public functions.
   (a) `int length();` // Return the length (number of elements) of the list.
   (b) `void insertNth(int, Type);` // Insert an element at the nth position. Remember to check for valid n ($0 \leqq n \leqq$ length).
   (c) `void deleteAll(Type);` // Delete ALL occurrences of an element.
   (d) `void shift(Type, char);` // Shift all elements by one position to the right if char is 'R' or to the left if char is 'L'. Note that the shifting should be performed in a circular way.

6. **(15%)** Write a string function `int myStringCompare(const char* s1, const char* s2)` with the following behavior. Returns -1 if s1 is shorter than s2 or s1 comes before s2 in dictionary order when they are of the same length. Returns 0 when s1 and s2 are the same. Returns 1 otherwise. Note that the function is semantically different from the `strcmp` function.

7. **(10%)** Given a linked list, write a function to determine if the list is symmetric. A list $[l_1, l_2, \ldots, l_n]$ is symmetric if $l_1 = l_n$, $l_2 = l_{n-1}$, etc. Note that n must be an even number. (Hint: use a stack.)