**CSC 228 Data Structures and Algorithms, Fall 2017**
**Instructor: Dr. Natarajan Meghanathan**

# Project 2: Implementation of Recursive Print and Merge List Functions for Singly Linked List

**Due:** September 27, 1 PM

Note: You should use the C++/Java source code attached to the project description for answering these questions.

Question 1 - SinglyLinkedListRecursivePrint_Project2_Question_1.cpp
SinglyLinkedListRecursivePrint_Project2_Question_1.java

Questions 2 and 3 - SinglyLinkedList_MergeLists_Project2_Question_2_3.cpp
SinglyLinkedList_MergeLists_Project2_Question_2_3.java

**Q1 - 40 pts)** Consider the implementation of the List ADT using Singly Linked List. Add a member function (to the List class) called *recursivePrintForwardReverseOrders* that prints the contents of the list in a recursive fashion in both the forward order and reverse order.

For example, if the contents of the List are: 10 --> 4 --> 8 --> 12 --> 9, the recursive member function should print the List as follows:
10 4 8 12 9
9 12 8 4 10

Note that both the forward and reverse orders should be printed through an invocation of the *recursivePrintForwardReverseOrders* member function on the List object called from the main function. You are free to choose the parameter(s) that need to be passed to the *recursivePrintForwardReverseOrders* function. But, you are not supposed to pass more than three parameter(s). A suggestion for the parameter to pass is given in the main function of the code posted for Question 1.

To test your code (and take screenshot), create a List of at least 5 elements and then call the *recursivePrintForwardReverseOrders* function on this List object by passing a pointer to the first node in the Linked List as an argument, as shown in the main function of the Singly Linked List code for Question 1.

**Q2 - 25 pts)** Add a member function *mergeList* to the Singly Linked List-based implementation of the List ADT. The member function takes as input a List object (representing the list of smaller size) as parameter and appends it to the List object (representing the list of larger size) on which the function will be called.

Consider largerIntegerList and smallerIntegerList to be the two List objects in the main function. The main function should call largerIntegerList.mergeList(smallerIntegerList) and then print the contents of the largerIntegerList using the IterativePrint( ) member function by calling largerIntegerList.IterativePrint( ) from main. The mergeList member function will append the elements of the smallerIntegerList to the end of the largerIntegerList. Use the Singly Linked List code for Question 2 attached in this project description.

For example:
If largerIntegerList is 78 --> 14 --> 52 --> 96 --> 52 --> 63 and
the smallerIntegerList is 44 --> 22 --> 11, then the contents of the largerIntegerList after the merger should be: 78 --> 14 --> 52 --> 96 --> 52 --> 63 --> 44 --> 22 --> 11.

To test your code (and take a screenshot), let the size of the largerIntegerList be any value from 6 to 10 and the size of the smallerIntegerList be any value from 3 to 5.

**Q3 - 35 pts)** Modify the mergeList member function added to the Singly Linked List-based List class for Question 2 in such a way that you append only elements (from the smaller link list) that are not already in the larger link list. Feel free to create additional member functions to facilitate this.

For example: if the largerIntegerList is 11 --> 22 --> 44 --> 55 --> 78 --> 89 and the smallerIntegerList is 22 --> 56 --> 89 --> 77, then the contents of the largerIntegerList (after the merger) should be : 11 --> 22 --> 44 --> 55 --> 78 --> 89 --> 56 --> 77.

Test your code as before by running the main function with a call to the mergeList function on the largerIntegerList object (and by passing the smallerIntegerList object as a parameter) such that only the unique elements are added to the list and printed.

**Submission (in Canvas):**
Submit a singly report that has separate C++/Java code (the complete code comprising of the Node class, List class and the main function) for each of Questions 1, 2 and 3. Also, include screenshots of the execution of the main function for all the three questions, as instructed above.