CS 3100/5100
Programming Assignment 1

Using the singly linked list structure available in the project1 dropbox and also in the public/project1 directory, add member functions to the List class to solve the following problems:

1. Write a public member function called **bool hasCycle();** to detect a cycle in a linked list using Floyd's Cycle Finding Algorithm (i.e. Tortoise and the Hare Algorithm). To test your solution add a public member function called **void makeCycle(int K);** that will alter the last node in a linked list so it points to the Kth node. If the list contains [10, 20, 30, 40, 50], calling makeCycle(3) will make the node containing 50 point to the node containing 20. If you attempted to print the list after called makeCycle(3), the list it will display an infinite list containing [10, 20, 30, 40, 50, 30, 40, 50, 30, 40, 50, ...]. A call to hasCycle() after a call to makeCycle returns True.

2. Write a public member function called **void unique();** to remove all duplicate items from a list. If the list contains the items [1, 3, 3, 2, 4, 6, 5, 5, 5, 7, 1, 2, 8, 9, 8], this function modifies the list to produce [1, 3, 2, 4, 6, 5, 7, 8, 9].

3. Write a public member function called v**oid removeDuplicates();** to remove all consecutive duplicate items in the list. If the list contains the items [1, 2, 2, 3, 4, 5, 5, 5, 6, 7, 1, 2, 8, 9, 8], this function modifies the list to produce [1, 2, 3, 4, 5, 6, 7, 1, 2, 8, 9, 8].

4. Write a public member function called **void reverse();** to reverse the list. Your solution should NOT be recursive and should NOT allocate any new nodes. Given the list, [1, 2, 3, 4, 5], a call to reverse() modifies the list to produce [5, 4, 3, 2, 1].

5. Write a public member function called **void recursiveReverse ();** that reverses the original list. Your solution MUST be recursive and must NOT allocate any new nodes. Given the list, [1, 2, 3, 4, 5], a call to recursiveReverse() returns a List object containing [5, 4, 3, 2, 1].

6. Write a public member function called **T getKth(int K);** to return the value in the node at the kth position in the list. If K > 0 return the value of the node counting from the beginning of the list. If K < 0 return the value of the node counting from the end of the list. Given the list [10, 15, 30, 12, 27, 17, 1, 19], a call to getKth(3) returns 30, a call to getKth(-1) returns 19 and a call to getKth(-4) returns 27. If the Kth position is before the first node or after the last node, return the value in the first or last node respectively. Do not assume you have direct access to the length of the list (i.e. the length is not stored).

7. Write a public member function called **void isSorted();** to determine whether or not the list is sorted in ascending order. Given the list, [12, 25, 31, 90, 99], a call to isSorted() returns True.

8. Write a public member function called **void sort();** to sort the list in ascending order. Given the list, [25, 90, 99, 31, 12], a call to sort() produces the list [12, 25, 31, 90, 99].

9. Write a public member function called **void mergeSort(List<T>& L2);** to merge two sorted list. Given the first list contains [1, 3, 4, 5, 6] and L2 contains [2, 4, 7, 12], a call to mergeSort() produces the list [1, 2, 3, 4, 4, 5, 6, 7, 12]. After calling mergeSort, L2 should be empty (i.e. move the node from L2 to the other list).

Points will be awarded for correctness (70%), quality of the code (20%) and documentation (10%). Your solution should be efficient with respect to execution time and storage space. To enforce this requirement, the points awarded for correctness will be further divided into two categories. You will receive 70% of the available correctness points for producing the desired behavior and 30% for efficiency in terms of time and space.

You may not alter the prototypes of the public member functions described in this assignment, but you may add additional public or private member functions to the List class. Do not alter the Node class or add any class level data members to List or Node.