More on Linked List

Objective

The objective of this lab is to practice more on linked list implementation.

In Lab Tasks

1. Find the middle node of a linked list. Assume no other information is given except head pointer.

2. Write a SortedMerge(Linklist L1, Linklist L2) function that takes two singly lists, each of which is sorted in increasing order, and merges the two together into one list which is in increasing order. SortedMerge() should return the new list. Ideally, sortedMerge() should only make one pass through each list.

Post Lab Tasks

3. Write a function that takes singly linked list and index as input and return the value stored in the node at that index position. For example, the node pointed by head has index=0, next to head node has index=1 and so on. You do not need to store index values. Prototype of function is given below:
   Public node getNode(Linklist L, index) { …. code here ……}

4. Write function InsertNth(Linklist L, index) which can insert a new node at index within a singly linked list. Function can pass any index in the range [0..length], and the new node should be inserted at that index.

5. Write an Append(Linklist ListA, Linklist ListB) function that takes two doubly linked lists, ListA and ListB, appends ListB onto the end of ListA, and then sets ListB to NULL.

6. Given a doubly linked list, split it into two sublists, one for the front half, and one for the back half. If the number of elements is odd, the extra element should go in the front list. So for example, public node[ ] ListSplit(Linklist L) on the list {2, 3, 5, 7, 11} should return the two lists {2, 3, 5} and {7, 11}.
   Note: Getting this right for all the cases is harder than it looks. You should check your solution against a few cases (length = 2, length = 3, length=4) to make sure that the list gets split correctly near the short-list boundary conditions. If it works right for length=4, it probably works right for length=1000. You will probably need special case code to deal with the (length <2) cases.

7. Write a RemoveDuplicates(Linklist L) function which takes a doubly list sorted in increasing order and deletes any duplicate nodes from the list. Ideally, the list should only be traversed once.