Department of Computer Science, IBA

Data Structures (3+1) Quratulain

Practice Questions

Note: Consider an implementation in doubly and singly LinkedList.

1. 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. The prototype of the function is public node getNode(Linklist L, index).

2. void addAll(List l) appends the list l to the last element of the current list, if the current list is nonempty, or lets the head of the current list point to the first element of l if the current list is empty.

3. 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 listThe function 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 a special case code to deal with the (length <2) cases.

4. Add a method itemInTheMiddle() to the LinkedList class that returns the item associated with the middle node of the list, but does so in a single iteration of the list and without knowing the size of the list ahead of time. In the case of a list with 2k items for some integer k, return the (k+1)th item.

5. Write a SortedMerge(Linklist L1, Linklist L2) function that takes two doubly 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. Note L1 and L2 remain unchanged.

6. 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.

7. 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.

8. 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.

9. Find the middle node of a doubly linked list in one iteration. Assume no other information is given except the head pointer.