Assignment 2: Data Structures Lab. Div. A Batch: B2

Sr. No	Problem statement	Roll Nos	
1	Given a 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 FrontBackSplit() on the list {2, 3, 5, 7, 11} should yield the two lists {2, 3, 5} and {7, 11}. 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.	2	7
2	WAP to perform addition of two polynomials using singly linked list.	8	
3	Write an iterative Reverse () function that reverses a list by rearranging all the .next pointers and the head pointer. Ideally, Reverse () should only need to make one pass of the list.	55	
4	Write an Append () function that takes two lists, 'a' and 'b', appends 'b' onto the end of 'a', and then sets 'b' to NULL (since it is now trailing off the end of 'a').	58	
5	WAP to perform Multiplication of two polynomials using singly linked list.	59	
6	Consider a CopyList() function that takes a list and returns a complete copy of that list. One pointer can iterate over the original list in the usual way. Two other pointers can keep track of the new list: one head pointer, and one tail pointer which always points to the last node in the new list.	61	
7	WAP to store at most 10 digit integer in a Singly linked list and perform arithmetic operations on it.	68	
8	WAP to create doubly linked list and perform following operations on it. A) Insert (all cases) 2. Delete (all cases).	69	
9	WAP to store at most 10 digit integer in a Doubly linked list and perform arithmetic operations on it.	70	
10	WAP to merge two sorted Doubly linked lists and display ther result.	72	

11	WAP to append one doubly linked list to a) the start of the Second list b) the end of the second list.	62	
12	Implement Push and POP operations of STACK on Doubly linked lists	51	
13	Implement ADD and DELETE operations of QUEUE on Doubly linked lists	15	
14	Implement Insertion sort using Singly Linked List	66	
15	Implement Bubble sort using Doubly Linked List	63	
16	Write a RemoveDuplicates() function which takes a list sorted in increasing order and deletes any duplicate nodes from the list. Ideally, the list should only be traversed once.	64	
17	Write a SortedInsert() function which given a list that is sorted in increasing order, and a single node, inserts the node into the correct sorted position in the list. While Push() allocates a new node to add to the list, SortedInsert() takes an existing node, and just rearranges pointers to insert it into the list.	65	
18	Given a 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 FrontBackSplit() on the list {2, 3, 5, 7, 11} should yield the two lists {2, 3, 5} and {7, 11}. 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.	67	
19.	WAP to perform addition of two polynomials using singly linked list.	12	