Computer Engineering Department, S V N I T, Surat.
B Tech-II (CO) 3rd semester
Course: Data Structure and Algorithm (CO-203)
Tutorial – 1

1. Given a linked list whose typical node consists of an INFO and LINK field, formulate an algorithm which will count the number of nodes in list.

2. Formulate an algorithm that will change the INFO field of the kth node to the value given by Y.

3. Formulate an algorithm which concatenates a linear linked list to another linear linked list.

4. Given a simple linked list whose first node is denoted by the pointer variable FIRST, it is required to split this list into two simply linked lists. The node denoted by the pointer variable SPLIT is to be the first element in the second linked list. Formulate a step-by-step algorithm to perform this task.

5. Formulate an algorithm which will reverse a singly linked list. Assume a typical node consists of an INFO and LINK field. The parameter to the function should be a pointer to the original list, and the function should return a pointer to the reversed list.

Computer Engineering Department, S V N I T, Surat.
B Tech-II (CO) 3rd semester
Course: Data Structure and Algorithm (CO-203)
Tutorial – 2

1. Formulate an algorithm which will perform an insertion to the immediate left of the k^{th} node in the list.

2. You’re given the pointer to the head node of a doubly linked list. Reverse the order of the nodes in the list. The head node might be NULL to indicate that the list is empty.

3. You're given the pointer to the head node of a sorted linked list, where the data in the nodes is in ascending order. Delete as few nodes as possible so that the list does not contain any value more than once. The given head pointer may be null indicating that the list is empty.

4. You’re given the pointer to the head node of a sorted doubly linked list and an integer to insert into the list. Create a node and insert it into the appropriate position in the list. The head node might be NULL to indicate that the list is empty.

5. You’re given the pointer to the head nodes of two sorted linked lists. The data in both lists will be sorted in ascending order. Change the next pointers to obtain a single, merged linked list which also has data in ascending order. Either head pointer given may be null meaning that the corresponding list is empty.