Lab 4 – Linked Lists

Note:

- **Use Singly Linked lists for all the programs given.**
- **When finished with all the questions w.r.t Singly linked Lists, write subsequent C programs to answer all the questions using Doubly Linked Lists.**

1. Write a C Program to count the number of elements in a given linked list.
2. Given a linked list L, write a C program to insert a node at the kth position.
3. Given a linked list L with the elements of L arranged in non-decreasing order, write a C program to insert a new node in its correct position.
4. 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. For example, the list {2, 3, 5, 7, 11} should yield the two lists {2, 3, 5} and {7, 11}.
5. Let $L = \{x1, x2, \ldots, xn\}$ be a list of *n* elements. Let us search for a key K in the list L. If the key is presented in the list *L* then partition the list *L* into disjoint ordered lists *L1* and *L2* such that $L1 = \{x: x \in L$ such that $x \leq K\}$ and $L2 = \{y: y \in L$ such that $y > K\}$. If the key is not present in the list output is "empty". Write an algorithm (using linked list) and subsequent C program for your algorithm to compute lists *L1* and *L2* for the given list *L* and key K. Note: Don't use any inbuilt functions in your program such as *sort.*

   Example1: If L={16, 15, 1, 27, 19, 100, 200,3} and key k= 27 then L1={1,3,15,16,19,27} and L2={100,200}.

   Example 2: If L={16, 15, 1, 27, 19, 100, 200,3} and key k= 127 then empty.

6. Write an algorithm and subsequently a C program that will take as input a number k < n, where n is the size of the singly linked list; and interchange the kth and the (k+1)th element in the list. Your algorithm should not interchange the values directly but instead interchange the elements by modifying the pointer/address variables only. For example, if the list given is 1-2-3-4-5 and k=3, then your output should be 1-2-4-3-5.
7. Given two linked lists L_1 and L_2, write a C program to output and store the elements of $L_1 \cap L_2$ into a third list L_3.

8. Given a linked list L, it is well known that accessing the elements one by one takes a longer time to access the last element of the list. Device a strategy to speed up the process and write a C program to demonstrate your principle.