# Experiment No- 07 (Group B)

**Title:-** C++ program to compute various operation using singly linked list.

**Objectives:-** To understand the operation of singly linked list

# Problem Statement:-

Department of Computer Engineering has student's club named 'Pinnacle Club'. Students of second, third and final year of department can be granted membership on request. Similarly one may cancel the membership of club. First node is reserved for president of club and last node is reserved for secretary of club. Write C++ program to maintain club member‘s information using singly linked list. Store student PRN and Name. Write functions to:

a) Add and delete the members as well as president or even secretary.

b) Compute total number of members of club

c) Display members

d) Two linked lists exists for two divisions. Concatenate two lists.

# Theory:-

# A singly linked list is a type of linked list that is unidirectional, that is, it can be traversed in only one direction from head to the last node (tail). Each element in a linked list is called a node

**Primitive operation on singly linked list: -**

1) Creation 2) Insertion 3) Deletion 4) Search 5) Display

**Algorithms:**

1. **Creation of Linked List:-**
2. Create a Node which has 3 parts, PRN number of a student, name of a student and next. Next is a pointer to the next node in the list. Next pointer for any new node is NULL
3. Create another class which has two attributes: head and tail.
4. To add a newly created node to the list:
   1. It first checks, whether the head is equal to null which means the list is empty.
   2. If the list is empty, both head and temp will point to the newly added node.
   3. If the list is not empty, the new node will be added to end of the list such that

temp-> next will point to a newly added node. This new node will become the new temp of the list

# Display of linked list:-

1. Define a node current which will initially point to the head of the list.
2. Traverse through the list till current points to null.
3. Display each node by making current to point to node next to it in each iteration.
4. **Counting the nodes present in the list:**
5. Define a node current which will initially point to the head of the list.
6. Declare and initialize a variable count to 0.
7. Traverse through the list till current point to null.
8. Increment the value of count by 1 for each node encountered in the list.

# Insertion of any element at anywhere in the linked list:-

There are three possible cases when we want to insert an element in the linked list –

# Insertion of a nose as a head node(for President)

1. Create new node p by using getnode() function
2. Change next of new node p to point to head
3. Change head to point to recently created node p
   1. **Insertion of a node after some node (For any member)**
4. Create new node p by using getnode() function
5. Ask for PRN number of a node after which you want to insert p
6. Traverse to node with required PRN number with the help of pointer temp
7. Attach next pointer of p to temp->next and change next pointer of temp to point to to p.
   1. **Insertion of a node at last (for secretary)**
8. Create new node p by using getnode() function
9. Initialize node temp with value of head pointer
10. Traverse next till temp->next !=NULL
11. Attach next pointer of p to temp->next and change next pointer of temp to point to to p.

# Deletion of any element at in the linked list-

1. Initialize node temp and prev with value of head pointer
2. Ask for PRN number of a node which you want to delete
3. Traverse to node with required PRN number with the help of pointer temp and shift pointer prev to follow temp
4. Attach next pointer of prev node to temp->next and change next pointer of temp to NULL
5. Free up the memory allocated for temp.

# Conclusion: Thus we studied, how to represent Singly linked list as an ADT.