Assignment No.7

Assignmet Title: Write C++ program to maintain club member‘s information using singly linked list.

Aim: 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) Display list in reverse order using recursion
e) Two linked lists exists for two divisions. Concatenate two lists

Input: Individual details
Output: Maintain information of the Club member's
Objectives:
To maintain club member's information by performing different operations like add, delete, reverse, concatenate on singly linked list.

Theory:
Linked List :

- **Definition :** A linked list is a sequence of data structures, which are connected together via links.

Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array. Following are the important terms to understand the concept of Linked List.

Link − Each link of a linked list can store a data called an element.

Next − Each link of a linked list contains a link to the next link called Next.

LinkedList − A Linked List contains the connection link to the first link called First.

Linked List Representation

Linked list can be visualized as a chain of nodes, where every node points to the next node.

- **Types of linked list :** Following are the various types of linked list.

Simple Linked List − Item navigation is forward only.

Doubly Linked List − Items can be navigated forward and backward.

Circular Linked List − Last item contains link of the first element as next and the first element has a link to the last element as previous.

- **Singly linked list**

Singly linked list can be defined as the collection of ordered set of elements. The number of elements may vary according to need of the program. A node in the singly linked list consist of two parts: data part and link part. Data part of the node stores actual information that is to be represented by the node while the link part of the node stores the address of its immediate successor.In other words, we can say that each node contains only next pointer, therefore we can not traverse the list in the reverse direction.

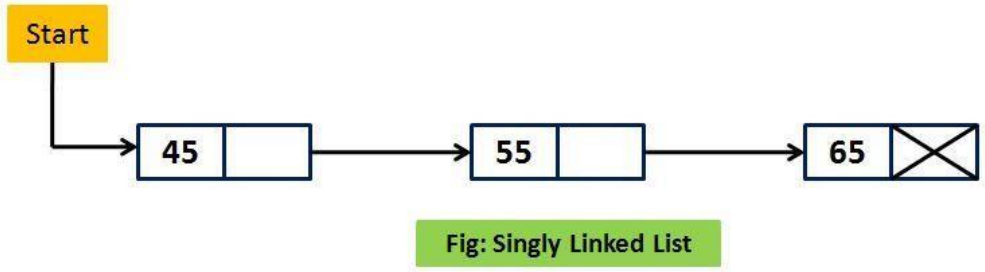

Fig: Singly Linked List

-

Advantages of Singly Linked List

There are some advantages of singly Linked List

- It is very easier for the accessibility of a node in the forward direction.
- The insertion and deletion of a node are very easy.
- The Requirement will less memory when compared to doubly, circular or doubly circular linked list.
- The Singly linked list is the very easy data structure to implement.
- During the execution, we can allocate or deallocate memory easily.
- Insertion and deletion of elements don't need the movement of all the elements when compared to an array.

Disadvantages of Singly Linked List

The disadvantages of singly Linked List are following

- Therefore, Accessing the preceding node of a current node is not possible as there is no backward traversal.
- The Accessing of a node is very time-consuming.

Operations on Singly Linked List

There are various operations which can be performed on singly linked list. A list of all such operations is given below.

Node Creation

```
struct node {
  int data;
 struct node *next;
};
struct node *head, *ptr;
ptr = (struct node *)malloc(sizeof(struct node *));
```

Insertion

The insertion into a singly linked list can be performed at different positions. Based on the position of the new node being inserted, the insertion is categorized into the following categories.

SN	Operation	Description
1	Insertion at beginning	It involves inserting any element at the front of the list. We just need to a few link adjustments to make the new node as the head of the list.
2	Insertion at end of the list	It involves insertion at the last of the linked list. The new node can be inserted as the only node in the list or it can be inserted as the last one. Different logics are implemented in each scenario.
3	Insertion after specified node	It involves insertion after the specified node of the linked list. We need to skip the desired number of nodes in order to reach the node after which the new node will be inserted. .

Deletion and Traversing

The Deletion of a node from a singly linked list can be performed at different positions. Based on the position of the node being deleted, the operation is categorized into the following categories.

SN	Operation	Description
1	Deletion at beginning	It involves deletion of a node from the beginning of the list. This is the simplest operation among all. It just need a few adjustments in the node pointers.
2	Deletion at the end of the list	It involves deleting the last node of the list. The list can either be empty or full. Different logic is implemented for the different scenarios.
3	Deletion after specified node	It involves deleting the node after the specified node in the list. we need to skip the desired number of nodes to reach the node after which the node will be deleted. This requires traversing through the list.
4	Traversing	In traversing, we simply visit each node of the list at least once in order to perform some specific operation on it, for example, printing data part of each node present in the list.
5	Searching	In searching, we match each element of the list with the given element. If the element is found on any of the location then location of that element is returned otherwise null is returned. .

Concatenating or joining two linked lists
Is not at all a difficult task. We just need to follow some very simple steps and the steps to join two lists (say 'a' and 'b') are as follows:
Traverse over the linked list 'a' until the element next to the node is not NULL.
If the element next to the current element is NULL (a->next == NULL) then change the element next to it to 'b' (a->next = b).

Algorithm

(Write your own algorithm for your program)

Program :

Questions:
1. What is a Linked list?
2. Can you represent a Linked list graphically?
3. How many pointers are required to implement a simple Linked list?
4. How many types of Linked lists are there?
5. How to represent a linked list node?
6. Describe the steps to insert data at the starting of a singly linked list.
7. How to insert a node at the end of Linked list?
8. How to delete a node from linked list?
9. How to reverse a singly linked list?
10. What is the difference between singly and doubly linked lists?
11. What are the applications that use Linked lists?
12.What will you prefer to use a singly or a doubly linked lists for traversing through a list of elemen