# ***Day2_25-02-2025***
# ***Linked Lists***

## Type declaration :

```
struct ListNode {
      int data;
      struct ListNode *next;
};
```

**Traversing :**

```
int ListLength(struct ListNode *head) {
      struct ListNode *current = head;
      int count = 0;
      while(current != NULL) {
            count++;
            current = current->next;
      }
      return count;
}
```

**TC : O(n)**
**SC : O(1)**

**Insertion at the beginning :**

```
struct ListNode *insertAtBeginning(struct ListNode *head, int data) {
    struct ListNode *temp;
    temp=(struct ListNode *)malloc(sizeof(struct ListNode));
    temp->data=data;
    temp->next=NULL;
    if(head == NULL) {
        head=temp;
        head->next=NULL;
    } else {
        temp->next=head;
        head=temp;
    }
    return head;
}
```

**Insertion at the ending :**

```
struct ListNode *insertAtEnd(struct ListNode *head, int data) {
    struct ListNode *temp, *curr;
    temp=(struct ListNode *)malloc(sizeof(struct ListNode));
    temp->data=data;
    temp->next=NULL;
    *curr = head;
    if(curr == NULL) {
        head=temp;
    } else {
        while(curr->next != NULL) {
            curr=curr->next;
        }
        curr->next=temp;
    }
    return head;
}
```

**Insertion at a given position :**

```
struct ListNode *insertAtGivenPosition(struct ListNode *head, struct
ListNode *new, int n) {
    struct ListNode *pred = head;
    if(n <= 1) {
        new->next = head;
        return new;
    }
    while(--n && pred != NULL) {
        pred = pred->next;
    }
    if(pred == NULL) {
        return NULL;
    }
    new->next = pred->next;
    pred->next = new;
    return head;
}
```

**Deleting the first node :**

```
void *deleteFirst(struct ListNode **head) {
    struct ListNode *temp;
    if(*head == NULL) {
        return;
    }
    temp = *head;
    *head = (*head)->next;
    free(temp);
    return;
}
```

**Deleting the last node :**

```
void *Last(struct ListNode **head) {
    struct ListNode *temp = NULL;
    struct ListNode *current = *head;
    if(*head == NULL) {
        return;
    }
    while(current->next) {
        temp = current;
        current = current->next;
    }
    temp->next = NULL;
    free(current);
    return;
}
```

**Deleting intermediate node :**

```
void *delete(struct ListNode **head, int position) {
    int k = 1;
    struct ListNode *p, *q;
    if(*head == NULL) {
        cout << "List Empty";
        return;
    }
    p = *head;
    if(position == 1) {
        *head = (*head)->next;
        free(p);
        return;
    } else {
        while(p != NULL && k < position) {
            k++;
            q = p;
            p = p->next;
        }
        if(p == NULL) {
            cout << "Position doesn't exist";
            return;
        } else {
            q->next = p->next;
            free(p);
        }
        return;
    }
}
```

**Delete LL :**

```
void deleteLL(struct ListNode **head) {
    struct ListNode *auxilaryNode, *iterator;
    iterator = *head;
    while(iterator) {
        auxilaryNode = iterator->next;
        free(iterator);
        iterator = auxilaryNode;
    }
    *head = NULL;
}
```