25/1/24

WEEK - 5

1. WAP to implement singly linked list with operations : Sort the linked list, reverse the list and concatenation of two linked lists.

```
#include <stdio.h>
#include <stdlib.h>
void reverse();
void sort();
void concatenate();
void insert1();
void insert2();
void display();
~~void~~ struct node
{
    int data;
    struct node *next;
};
struct node *head1 = NULL, *head2 = NULL;
void main()
{
    printf("Insert the elements in first list:\n");
    insert1();
    printf("Sorted list:\n");
    sort();
    display();
    printf("Reversed list:\n");
    reverse();
    display();
```

```
printf("Insert the elements of second
        list to concatenate :\n");
insert2();
concatenate();
printf("Concatenated list :\n");
display();
}

void sort()
{
    struct node *curr = head;
    struct node *ptr = NULL;
    int temp;
    while(curr != NULL)
    {
        ptr = curr -> next;
        while(ptr != NULL)
        {
            if(curr -> data > ptr -> data)
            {
                temp = curr -> data;
                curr -> data = ptr -> data;
                ptr -> data = temp;
            }
            ptr = ptr -> next;
        }
        curr = curr -> next;
    } free(ptr);
}
```

Ended 25/1/24

```
void reverse ()
{
    struct node *prev = NULL;
    struct node *ptr = NULL;
    while (head1 != NULL)
    {
        ptr = head1 -> next;
        head1 -> next = prev;
        prev = head1;
        head1 = ptr;
    }
    head1 = prev;
}

void concatenate ()
{
    struct node *temp = head1;

    if (head1 == NULL)
    {
        struct node *p = head2;
        while (p != NULL)
        {
            printf("%d -> ", p -> data);
            p = p -> next;
        }
        printf("NULL \n");
    }
    else
    {
        while (temp1 -> next1 != NULL)
            temp = temp -> next;
```

```
            temp -> next = head2;
        }
    }

void display ()
{
    struct node *p = head1;
    while(p != NULL)
    {
        printf("%d -> ", p -> data);
        p = p -> next;
    }
    printf("NULL \n");
}

void insert1()
{
    int data, n;
    struct node *last = head1;
    struct node * new_node = (struct node *)
            malloc(sizeof(struct node));
    printf("Enter no of nodes: ");
    scanf("%d", &n);
    for(int i=0; i<n; i++)
    {
        printf("Enter the data: ");
        scanf("%d", &new_node -> data);
        new_node -> next = NULL;
        while(last -> next != NULL)
            last = last -> next;
        last -> next = new_node;
    }
}
```

```
void insert2()
{
    int data, n;
    printf(" Enter no. of nodes:");
    scanf("%d", &n);
    for (int i=0; i<n; i++)
    {
        struct node *last2=head2;
        struct node * new_node;
        printf(" Enter data:");
        scanf("%d", &new_node->data);
        while (last->next != NULL)
            last = last->next;
        last->next = new_node;
    }
}
```

OUTPUT:

```
Insert the elements in first list:
Enter no. of nodes : 5
Enter the data : 3
Enter the data : 2
Enter the data : 5
Enter the data : 1
Enter the data : 4
Sorted list :
1 -> 2 -> 3 -> 4 -> 5 -> NULL
Reversed list :
5 -> 4 -> 3 -> 2 -> 1 -> NULL
Insert elements in second list :
Enter no. of nodes : 2
Enter the data : 6
Enter the data : 7
Concatenated list :
5 -> 4 -> 3 -> 2 -> 1 -> 6 -> 7 -> NULL
```