Lab 6 - 5/2/24

a) WAP to sort , reverse and concatenate singly linked lists:

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

static void reverse(struct Node** head_ref)
{
    struct Node* prev = NULL;
    struct Node* current = *head_ref;
    struct Node* next = NULL;
    while (current != NULL) {
        next = current->next;

        current->next = prev;

        prev = current;
        current = next;
    }
    *head_ref = prev;
}
```

```
struct Node* swap(struct Node* ptr1, struct Node* ptr2)
{
   struct Node* tmp = ptr2->next;
   ptr2->next = ptr1;
   ptr1->next = tmp;
   return ptr2;
}

int bubbleSort(struct Node** head, int count)
{
   struct Node** h;
   int i, j, swapped;

   for (i = 0; i <= count; i++) {

      h = head;
      swapped = 0;

      for (j = 0; j < count - i - 1; j++) {

         struct Node* p1 = *h;
         struct Node* p2 = p1->next;

         if (p1->data > p2->data) {
```

```
            *h = swap(p1, p2);
            swapped = 1;
          }

          h = &(*h)->next;
       }

       if (swapped == 0)
          break;
    }
}

void concat(struct Node* head1,struct Node* head2){
    struct Node* temp = head1;
    while(temp->next != NULL){
       temp = temp->next;
    }
    temp->next = head2;
}



void push(struct Node** head_ref, int new_data)
{
    struct Node* new_node
```

```
        = (struct Node*)malloc(sizeof(struct Node));
    new_node->data = new_data;
    new_node->next = (*head_ref);
    (*head_ref) = new_node;
}



void printList(struct Node* head)
{
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
}

int main()
{
    struct Node* head = NULL;

    push(&head, 20);
    push(&head, 4);
    push(&head, 15);
    push(&head, 85);
```

```
    printf("Given linked list\n");
    printList(head);
    reverse(&head);
    printf("\nReversed linked list \n");
    printList(head);
    printf("\nSorted linked list \n");
    bubbleSort(&head,4);
    printList(head);

    struct Node* head2 = NULL;

    push(&head2, 2);
    push(&head2, 40);
    push(&head2, 1);
    push(&head2, 8);
    printf("\nConatenated linked list \n");
    concat(head,head2);
    printList(head);
}
```

```
Given linked list
85 15 4 20
Reversed linked list
20 4 15 85
Sorted linked list
4 15 20 85
Conatenated linked list
4 15 20 85 8 1 40 2
```

b) WAP to implement doubly linked list and perform insertion and deletion operations:

```
#include<stdio.h>
#include<stdlib.h>

struct node
{
   int data;
   struct node *next;
   struct node *prev;
};
struct node *head;

void create(int item)
{
  struct node *ptr = (struct node *)malloc(sizeof(struct node));
  if(ptr == NULL)
  {
     printf("\nOVERFLOW\n");
  }
  else
  {



  if(head==NULL)
  {
     ptr->next = NULL;
     ptr->prev=NULL;
```

```
      ptr->data=item;
      head=ptr;
   }
   else
   {
      ptr->data=item;
      ptr->prev=NULL;
      ptr->next = head;
      head->prev=ptr;
      head=ptr;
   }
    printf("\nNode Inserted\n");
}

}

void delete( )
{
    struct node *ptr, *temp;
    int val;
    printf("Enter the value");
    scanf("%d",&val);
    temp = head;
    while(temp -> data != val)
    temp = temp -> next;
    if(temp -> next == NULL)
    {
        printf("\nCan't delete\n");
```

```
    }
    else if(temp -> next -> next == NULL)
    {
        temp ->next = NULL;
        printf("\nNode Deleted\n");
    }
    else
    {
        ptr = temp -> prev;
        ptr -> next = temp -> next;
        temp -> next -> prev = ptr;
        free(temp);
        printf("\nNode Deleted\n");
    }
}

void insert(struct node* next_node, int new_data)
{
    if (next_node == NULL) {
        printf("the given next node cannot be NULL");
        return;
    }

    struct node* new_node
        = (struct node*)malloc(sizeof(struct node));

    new_node->data = new_data;
```

```
   new_node->prev = next_node->prev;

   next_node->prev = new_node;

   new_node->next = next_node;

   if (new_node->prev != NULL)
      new_node->prev->next = new_node;
   else
      head = new_node;
}

void display() {
   struct node *current = head;
   if(head == NULL) {
      printf("List is empty\n");
      return;
   }
   printf("Nodes of doubly linked list: \n");
   while(current != NULL) {
      printf("%d ", current->data);
      current = current->next;
   }
}

void main(){
   create(2);
   create(3);
```

```
    insert(head,5);
    insert(head,1);
    insert(head,6);
    display(head);
    delete();
    display(head);
}
```

```
Node Inserted

Node Inserted
Nodes of doubly linked list:
6 1 5 3 2 Enter the value5

Node Deleted
Nodes of doubly linked list:
6 1 3 2
```