6a) WAP to Implement Single Link List with following operations: Sort the linked list, Reverse the linked list, Concatenation of two linked lists.

```c
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int val;
    struct node *next;
};
struct node *createsll()
{
    struct node *start = NULL , *p ,*last ;
    int item;
    printf("\nEnter element -999 to exit:");
    scanf("%d",&item);
    while(item!=-999)
    {
        p=(struct node *)malloc(sizeof(struct node));
        p->val=item;
        if(start==NULL)
        {
            p->next=NULL;
            start=p;
            last=p;
        }
        else{
            last->next=p;
            p->next=NULL;
            last=p;
        }
        scanf("%d",&item);
    }
    return start;
};

struct node *reversesll(struct node * start)
{
    struct node *prev = NULL;
    struct node *curr=start;
    struct node *next=NULL;
    while(curr!=NULL)
```

```
    {
        next=curr->next;
        curr->next=prev;
        prev=curr;
        curr=next;
    }
    return prev;
};


struct node *sortsll(struct node *start)
{
    struct node *min , *i , *j ;
    int temp;
    for(i=start ; i!=NULL ; i=i->next)
    {
        min=i;
        for(j=i->next ; j!=NULL ; j=j->next)
        {
            if(j->val < min->val)
                min=j;
        }
        temp=min->val;
        min->val=i->val;
        i->val=temp;
    }
    return start;
};

struct node *concatsll(struct node *start1 , struct node *start2)
{
    struct node *temp = start1;
    while(temp->next!=NULL)
        temp=temp->next;
    temp->next=start2;
    return start1;

};

void display(struct node *start)
{
    struct node *temp=start;
    while(temp!=NULL)
    {
        printf("%d->",temp->val);
        temp=temp->next;
    }

}
```

```c
int main()
{
    struct node *start1=NULL , *start2=NULL;
    printf("\nCreate first linked list :");
    start1=createsll();
    printf("\nCreate second linked list :");
    start2=createsll();
    printf("\nAfter sorting First sll :");
    start1=sortsll(start1);
    display(start1);
    printf("\nAfter reversing first sll:");
    start1=reversesll(start1);
    display(start1);
    printf("\nAfter concatenating both linked lists:");
    start1=concatsll(start1,start2);
    display(start1);

}
```

Output

```
Create first linked list :
Enter element -999 to exit:20
30
25
10
-999

Create second linked list :
Enter element -999 to exit:40
50
60
-999

After sorting First sll :10->20->25->30->
After reversing first sll:30->25->20->10->
After concatenating both linked lists:30->25->20->10->40->50->60->
Process returned 0 (0x0)   execution time : 18.815 s
Press any key to continue.
```

6b) WAP to Implement Single Link List to simulate Stack & Queue Operations

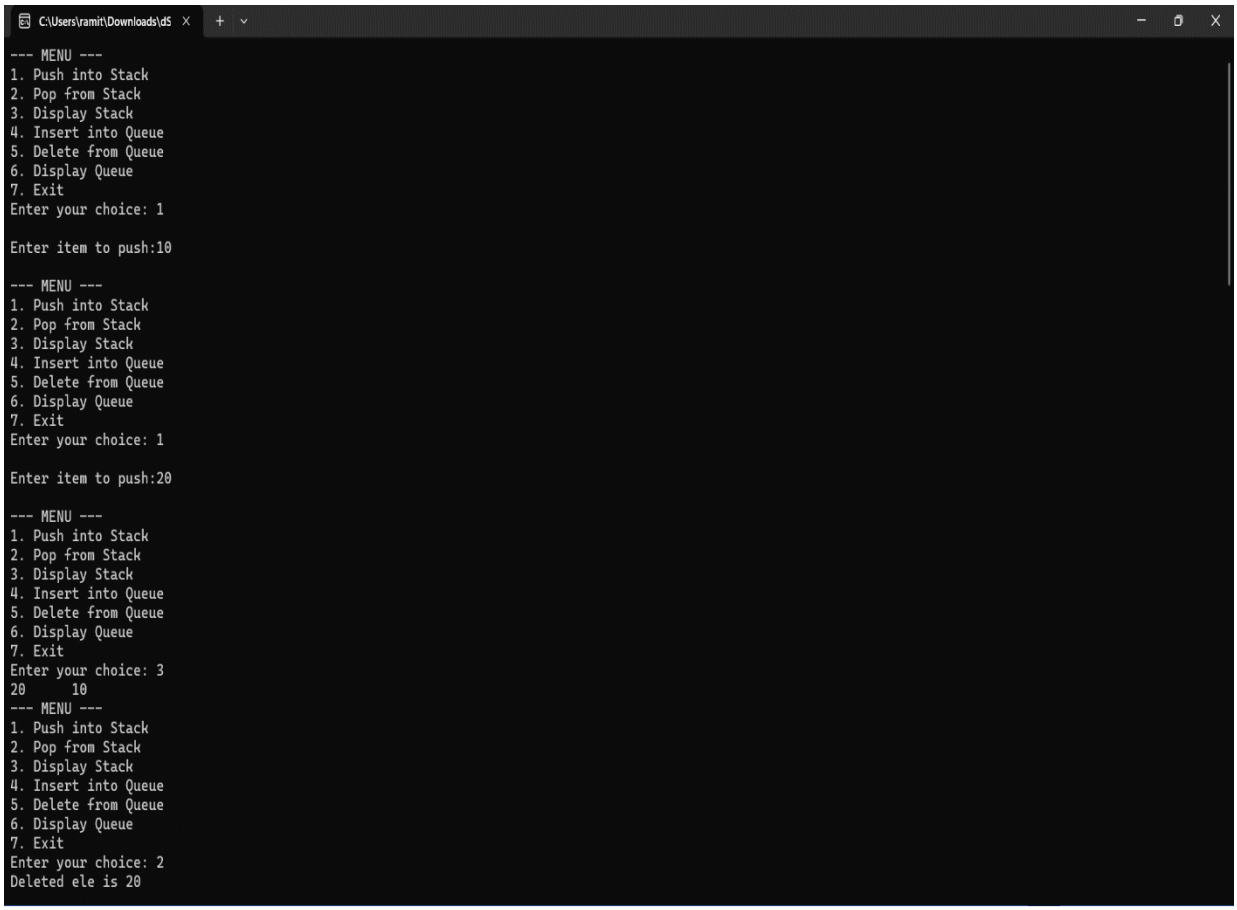
```c
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int val;
    struct node *next;
};
struct node *createsll()
{
    struct node *start = NULL , *p ,*last ;
    int item;
    printf("\nEnter element -999 to exit:");
    scanf("%d",&item);
    while(item!=-999)
    {
        p=(struct node *)malloc(sizeof(struct node));
        p->val=item;
        if(start==NULL)
        {
            p->next=NULL;
            start=p;
            last=p;
        }
        else{
            last->next=p;
            p->next=NULL;
            last=p;
        }
        scanf("%d",&item);
    }
    return start;
};
struct node *stackpush(struct node *start)
{
    struct node *p;
    int item;
    printf("\nEnter item to push:");
    scanf("%d",&item);
    p=(struct node *)malloc(sizeof(struct node));
    p->val=item;
    if(start==NULL)
    {
```

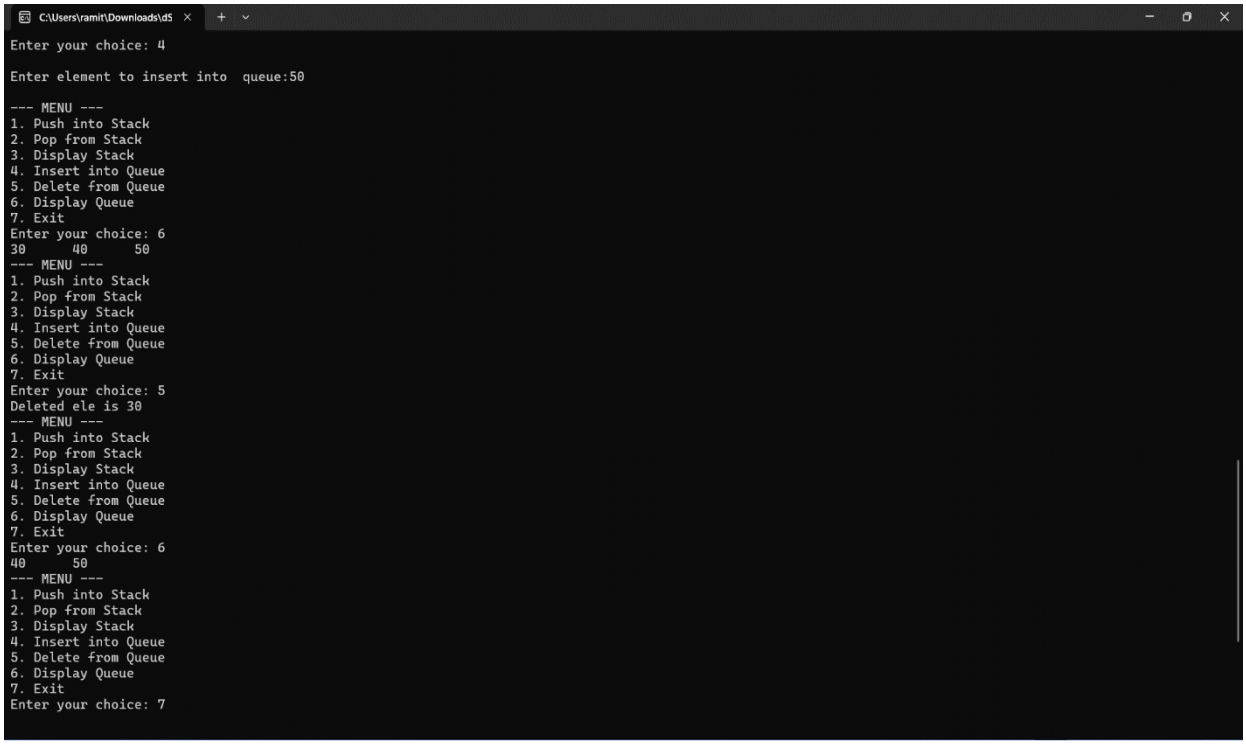
```
        p->next=NULL;
        start=p;
    }
    else
    {
        p->next=start;
        start=p;
    }
    return start;

};

struct node *pop(struct node *start)
{
    struct node *p;
    if(start ==NULL)
    {
        printf("Stack/Queue is empty");
    }
    else{
        p=start;
        start=start->next;
        printf("Deleted ele is %d",p->val);
        free(p);
    }
    return start;
};

struct node *queueinsert(struct node *start)

{
    struct node *p , *temp;
    int ele;
    printf("\nEnter element to insert into  queue:");
    scanf("%d",&ele);
    p=(struct node *)malloc(sizeof(struct node ));
    p->val=ele;
    if(start==NULL)
    {
        p->next=NULL;
        start=p;

    }
    else{
        temp=start;
        while(temp->next!=NULL)
            temp=temp->next;
        temp->next=p;
```

```
        p->next=NULL;
    }
    return start;
};

void display(struct node *start)
{
    struct node *temp=start;
    while(temp!=NULL)
        {
            printf("%d\t",temp->val);
            temp=temp->next;
        }



}

int main() {
    struct node *stack = NULL;
    struct node *queue = NULL;
    int choice;

    while (1) {
        printf("\n--- MENU ---\n");
        printf("1. Push into Stack\n");
        printf("2. Pop from Stack\n");
        printf("3. Display Stack\n");
        printf("4. Insert into Queue\n");
        printf("5. Delete from Queue\n");
        printf("6. Display Queue\n");
        printf("7. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                stack = stackpush(stack);
                break;
            case 2:
                stack = pop(stack);
                break;
            case 3:
                display(stack);
                break;
            case 4:
                queue = queueinsert(queue);
                break;
            case 5:
```

```
                queue = pop(queue);
                break;
            case 6:
                display(queue);
                break;
            case 7:
                printf("\nExiting program...\n");
                exit(0);
            default:
                printf("\nInvalid choice! Try again.\n");
        }
    }
    return 0;
}
```

OUTPUT:




C:\Users\ramit\Downloads\d5
Enter your choice: 4
Enter element to insert into queue:50
--- MENU ---
1. Push into Stack
2. Pop from Stack
3. Display Stack
4. Insert into Queue
5. Delete from Queue
6. Display Queue
7. Exit
Enter your choice: 6
30 40 50
--- MENU ---
1. Push into Stack
2. Pop from Stack
3. Display Stack
4. Insert into Queue
5. Delete from Queue
6. Display Queue
7. Exit
Enter your choice: 5
Deleted ele is 30
--- MENU ---
1. Push into Stack
2. Pop from Stack
3. Display Stack
4. Insert into Queue
5. Delete from Queue
6. Display Queue
7. Exit
Enter your choice: 6
40 50
--- MENU ---
1. Push into Stack
2. Pop from Stack
3. Display Stack
4. Insert into Queue
5. Delete from Queue
6. Display Queue
7. Exit
Enter your choice: 7