WEEK 8:

**Q.Program to implement Doubly linked list Insertion and deletion:**

#include<stdio.h>  
#include<stdlib.h>  
  
struct node  
{  
    struct node\*prev;  
    int data;  
    struct  node\*next;  
};  
struct node\* create(int d)  
{  
    struct node\* newnode=(struct node\*)malloc(sizeof(struct node));  
    newnode->data=d;  
};  
void insert\_first(struct node\*\* head,int d)  
{  
    struct node\* newnode=create(d);  
    if(\*head==NULL)  
    {  
        newnode->next=NULL;  
        newnode->prev=NULL;  
        \*head=newnode;  
    }  
    else{  
        newnode->next=\*head;  
        newnode->prev=NULL;  
        \*head=newnode;  
    }  
}  
void insert\_end(struct node\*\* head,int d)  
{  
    struct node\* newnode=create(d);  
    if(\*head==NULL)  
    {  
        \*head=newnode;  
        newnode->next=NULL;  
        newnode->prev=NULL;  
        \*head=newnode;  
    }  
    else{  
        struct node\* temp=\*head;  
        while(temp->next!=NULL)  
        {  
  
            temp=temp->next;  
        }  
        temp->next=newnode;  
        newnode->prev=temp;  
        newnode->next=NULL;  
    }  
}  
  
void del\_first(struct node\*\* head)  
{  
  
    if(\*head==NULL)  
    {  
  
        printf("empty\n");  
        return;  
    }  
    else{  
        struct node\* temp=\*head;  
        \*head=temp->next;  
        free(temp);  
    }  
}  
  
void del\_end(struct node\*\* head)  
{struct node\* temp=\*head;  
    if(\*head==NULL)  
    {  
        printf("empty!");  
    }  
    else if(temp->next==NULL)  
    {  
  
        \*head=NULL;  
    }  
    else{  
  
        while(temp->next->next!=NULL)  
        {  
  
            temp=temp->next;  
        }  
        temp->next=NULL;  
  
    }  
}  
  
void display(struct node\*\* head)  
{  
  
    struct node\* temp=\*head;  
    if(\*head==NULL)  
    {  
  
        printf("empty!!");  
    }  
    else{  
        while(temp->next!=NULL)  
        {  
  
            printf("%d->",temp->data);  
            temp=temp->next;  
        }  
        printf("%d->NULL",temp->data);  
    }  
}  
void insert\_at\_pos(struct node\*\* head,int d,int pos)  
{  
    int count;  
    struct node\* t=\*head;  
    struct node\* temp=\*head;  
    struct node\* newnode=create(d);  
    while(t->next!=NULL)  
    {  
        count++;  
        t=t->next;  
    }  
    count++;  
    if(count<pos)  
    {  
        printf("invalid position");  
    }  
    else if(pos==0)  
    {  
        insert\_first(head,d);  
    }  
    else if(pos==count)  
    {  
        insert\_end(head,d);  
    }  
    else{  
        for(int i=0;i<pos-2;i++)  
        {  
            temp=temp->next;  
        }  
        newnode->next=temp->next;  
        temp->next=newnode;  
        newnode->prev=temp;  
    }  
  
}  
void del\_at\_pos(struct node\*\* head,int pos)  
{  
    int count;  
    struct node\* t=\*head;  
    struct node\* temp=\*head;  
    while(t->next!=NULL)  
    {  
        count++;  
        t=t->next;  
    }  
    count++;  
    if(count<pos)  
    {  
        printf("invalid position");  
    }  
    else if(pos==0)  
    {  
        del\_first(head);  
    }  
    else if(pos==count)  
    {  
        del\_end(head);  
    }  
    else{  
        for(int i=0;i<pos-1;i++)  
        {  
            temp=temp->next;  
        }  
        temp->next=temp->next->next;  
        temp->next->prev=temp;  
    }  
  
}  
int main()  
{  
  
    struct node\* head=NULL;  
    int d,pos,ele,c;  
    while(1)  
    {  
        printf("enter the operation you want to perform");  
        printf("1.INSERT AT FRONT\n");  
        printf("2.INSERT AT END\n");  
        printf("3INSERT AT POSITION\n");  
        printf("4.DELETE FRONT\n");  
        printf("5.DELETE AT END\n");  
        printf("6.DELETE   AT POSITION\n");  
        printf("7.DISPLAY\n");  
        printf("8.EXIT\n");  
        scanf("%d",&c);  
        switch(c)  
        {  
        case 1:  
            printf("enter the data");  
            scanf("%d",&d);  
            insert\_first(&head,d);  
            break;  
        case 2:  
            printf("enter the data");  
            scanf("%d",&d);  
            insert\_end(&head,d);  
            break;  
        case 3:  
            printf("enter the data");  
            scanf("%d",&d);  
            printf("enter the position");  
            scanf("%d",&pos);  
            insert\_at\_pos(&head,d,pos);  
            break;  
        case 4:  
            del\_first(&head);  
            break;  
        case 5:  
            del\_end(&head);  
            break;  
        case 6:  
            printf("enter the position");  
            scanf("%d",&pos);  
            del\_at\_pos(&head,pos);  
            break;  
        case 7:  
            display(&head);  
            break;  
        case 8:  
            printf("exiting---\n");  
            return 0;  
        }  
  
    }  
    return 0;  
}

#include<stdio.h>

#include<stdlib.h>

struct node

{

    struct node\*prev;

    int data;

    struct  node\*next;

};

struct node\* create(int d)

{

    struct node\* newnode=(struct node\*)malloc(sizeof(struct node));

    newnode->data=d;

};

void insert\_first(struct node\*\* head,int d)

{

    struct node\* newnode=create(d);

    if(\*head==NULL)

    {

        newnode->next=NULL;

        newnode->prev=NULL;

        \*head=newnode;

    }

    else{

        newnode->next=\*head;

        newnode->prev=NULL;

        \*head=newnode;

    }

}

void insert\_end(struct node\*\* head,int d)

{

    struct node\* newnode=create(d);

    if(\*head==NULL)

    {

        \*head=newnode;

        newnode->next=NULL;

        newnode->prev=NULL;

        \*head=newnode;

    }

    else{

        struct node\* temp=\*head;

        while(temp->next!=NULL)

        {

            temp=temp->next;

        }

        temp->next=newnode;

        newnode->prev=temp;

        newnode->next=NULL;

    }

}

void del\_first(struct node\*\* head)

{

    if(\*head==NULL)

    {

        printf("empty\n");

        return;

    }

    else{

        struct node\* temp=\*head;

        \*head=temp->next;

        free(temp);

    }

}

void del\_end(struct node\*\* head)

{struct node\* temp=\*head;

    if(\*head==NULL)

    {

        printf("empty!");

    }

    else if(temp->next==NULL)

    {

        \*head=NULL;

    }

    else{

        while(temp->next->next!=NULL)

        {

            temp=temp->next;

        }

        temp->next=NULL;

    }

}

void display(struct node\*\* head)

{

    struct node\* temp=\*head;

    if(\*head==NULL)

    {

        printf("empty!!");

    }

    else{

        while(temp->next!=NULL)

        {

            printf("%d->",temp->data);

            temp=temp->next;

        }

        printf("%d->NULL",temp->data);

    }

}

void insert\_at\_pos(struct node\*\* head,int d,int pos)

{

    int count;

    struct node\* t=\*head;

    struct node\* temp=\*head;

    struct node\* newnode=create(d);

    while(t->next!=NULL)

    {

        count++;

        t=t->next;

    }

    count++;

    if(count<pos)

    {

        printf("invalid position");

    }

    else if(pos==0)

    {

        insert\_first(head,d);

    }

    else if(pos==count)

    {

        insert\_end(head,d);

    }

    else{

        for(int i=0;i<pos-2;i++)

        {

            temp=temp->next;

        }

        newnode->next=temp->next;

        temp->next=newnode;

        newnode->prev=temp;

    }

}

void del\_at\_pos(struct node\*\* head,int pos)

{

    int count;

    struct node\* t=\*head;

    struct node\* temp=\*head;

    while(t->next!=NULL)

    {

        count++;

        t=t->next;

    }

    count++;

    if(count<pos)

    {

        printf("invalid position");

    }

    else if(pos==0)

    {

        del\_first(head);

    }

    else if(pos==count)

    {

        del\_end(head);

    }

    else{

        for(int i=0;i<pos-1;i++)

        {

            temp=temp->next;

        }

        temp->next=temp->next->next;

        temp->next->prev=temp;

    }

}

int main()

{

    struct node\* head=NULL;

    int d,pos,ele,c;

    while(1)

    {

        printf("enter the operation you want to perform");

        printf("1.INSERT AT FRONT\n");

        printf("2.INSERT AT END\n");

        printf("3INSERT AT POSITION\n");

        printf("4.DELETE FRONT\n");

        printf("5.DELETE AT END\n");

        printf("6.DELETE   AT POSITION\n");

        printf("7.DISPLAY\n");

        printf("8.EXIT\n");

        scanf("%d",&c);

        switch(c)

        {

        case 1:

            printf("enter the data");

            scanf("%d",&d);

            insert\_first(&head,d);

            break;

        case 2:

            printf("enter the data");

            scanf("%d",&d);

            insert\_end(&head,d);

            break;

        case 3:

            printf("enter the data");

            scanf("%d",&d);

            printf("enter the position");

            scanf("%d",&pos);

            insert\_at\_pos(&head,d,pos);

            break;

        case 4:

            del\_first(&head);

            break;

        case 5:

            del\_end(&head);

            break;

        case 6:

            printf("enter the position");

            scanf("%d",&pos);

            del\_at\_pos(&head,pos);

            break;

        case 7:

            display(&head);

            break;

        case 8:

            printf("exiting---\n");

            return 0;

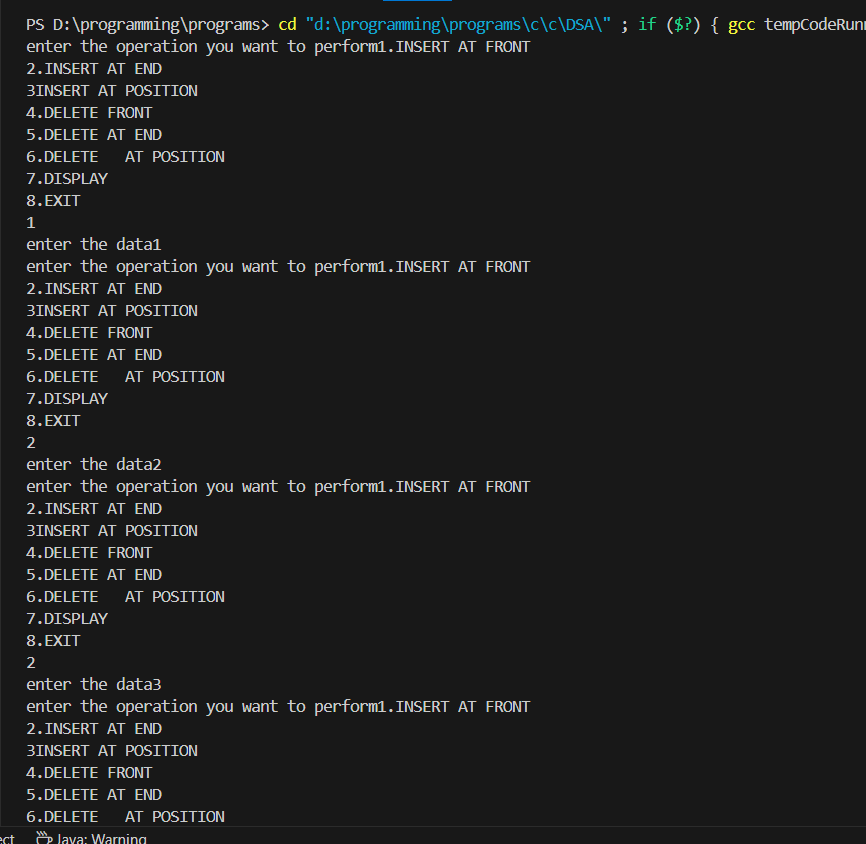
        }

    }

    return 0;

}

**OUTPUTS:**

****

