**DAY 6 – LINKED LIST**

8. Write a menu driven C program to implement various linked list

operations.

(i) Insertion

a. Insert at the beginning

b. Insert at the end

c. Insert after a specified node

(ii) Deletion

a. Delete from the beginning

b. Delete from the end

c. Delete a specified node

(iii) Display

**PROGRAM**

#include<stdlib.h>

#include <stdio.h>

struct node

{

        int info;

        struct node \*next;

} \*start=NULL;

void display()

{

        struct node \*ptr;

        if(start==NULL)

        {

                printf("\nList is empty:\n");

                return;

        }

        else

        {

                ptr=start;

                printf("\nThe List elements are:\n");

                while(ptr!=NULL)

                {

                        printf("%d\t",ptr->info );

                        ptr=ptr->next ;

                }

        }

}

void insert\_begin()

{

        struct node \*temp;

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

        if(temp==NULL)

        {

                printf("\nOut of Memory Space:\n");

                return;

        }

        printf("\nEnter the data value for the node:\t" );

        scanf("%d",&temp->info);

        temp->next =NULL;

        if(start==NULL)

        {

                start=temp;

        }

        else

        {

                temp->next=start;

                start=temp;

        }

}

void insert\_end()

{

        struct node \*temp,\*ptr;

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

        if(temp==NULL)

        {

                printf("\nOut of Memory Space:\n");

                return;

        }

        printf("\nEnter the data value for the node:\t" );

        scanf("%d",&temp->info );

        temp->next =NULL;

        if(start==NULL)

        {

                start=temp;

        }

        else

        {

                ptr=start;

                while(ptr->next !=NULL)

                {

                        ptr=ptr->next ;

                }

                ptr->next =temp;

        }

}

void insert\_pos()

{

        struct node \*ptr,\*temp;

        int i,pos;

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

        if(temp==NULL)

        {

                printf("\nOut of Memory Space:\n");

                return;

        }

        printf("\nEnter the position for the new node to be inserted:\t");

        scanf("%d",&pos);

        printf("\nEnter the data value of the node:\t");

        scanf("%d",&temp->info) ;

        temp->next=NULL;

        if(pos==0)

        {

                temp->next=start;

                start=temp;

        }

        else

        {

                for(i=0,ptr=start;i<pos-1;i++)

                {

                        ptr=ptr->next;

                        if(ptr==NULL)

                        {

                                printf("\nPosition not found:[Handle with care]\n");

                                return;

                        }

                }

                temp->next =ptr->next ;

                ptr->next=temp;

        }

}

void delete\_begin()

{

        struct node \*ptr;

        if(start==NULL)

        {

                printf("\nList is Empty:\n");

                return;

        }

        else

        {

                ptr=start;

                start=start->next ;

                printf("\nThe deleted element is :%d\t",ptr->info);

                free(ptr);

        }

}

void delete\_end()

{

        struct node \*temp,\*ptr;

        if(start==NULL)

        {

                printf("\nList is Empty:");

                exit(0);

        }

        else if(start->next ==NULL)

        {

                ptr=start;

                start=NULL;

                printf("\nThe deleted element is:%d\t",ptr->info);

                free(ptr);

        }

        else

        {

                ptr=start;

                while(ptr->next!=NULL)

                {

                        temp=ptr;

                        ptr=ptr->next;

                }

                temp->next=NULL;

                printf("\nThe deleted element is:%d\t",ptr->info);

                free(ptr);

        }

}

void delete\_pos()

{

        int i,pos;

        struct node \*temp,\*ptr;

        if(start==NULL)

        {

                printf("\nThe List is Empty:\n");

                exit(0);

        }

        else

        {

                printf("\nEnter the position of the node to be deleted:\t");

                scanf("%d",&pos);

                if(pos==0)

                {

                        ptr=start;

                        start=start->next ;

                        printf("\nThe deleted element is:%d\t",ptr->info  );

                        free(ptr);

                }

                else

                {

                        ptr=start;

                        for(i=0;i<pos;i++) { temp=ptr; ptr=ptr->next ;

                                if(ptr==NULL)

                                {

                                        printf("\nPosition not Found:\n");

                                        return;

                                }

                        }

                        temp->next =ptr->next ;

                        printf("\nThe deleted element is:%d\t",ptr->info );

                        free(ptr);

                }

        }

}

void main()

{

        int choice;

        while(1){

                printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

                printf("\n 1.Display    \n");

                printf("\n 2.Insert at the beginning    \n");

                printf("\n 3.Insert at the end  \n");

                printf("\n 4.Insert at specified position       \n");

                printf("\n 5.Delete from beginning      \n");

                printf("\n 6.Delete from the end        \n");

                printf("\n 7.Delete from specified position    \n");

                printf("\n 8.Exit       \n");

                printf("\n--------------------------------------\n");

                printf("Enter your choice:\t");

                scanf("%d",&choice);

                switch(choice)

                {

                        case 1:

                                        display();

                                        break;

                        case 2:

                                        insert\_begin();

                                        break;

                        case 3:

                                        insert\_end();

                                        break;

                        case 4:

                                        insert\_pos();

                                        break;

                        case 5:

                                        delete\_begin();

                                        break;

                        case 6:

                                        delete\_end();

                                        break;

                        case 7:

                                        delete\_pos();

                                        break;

                        case 8:

                                        exit(0);

                                        break;

                        default:

                                        printf("\n Wrong Choice:n");

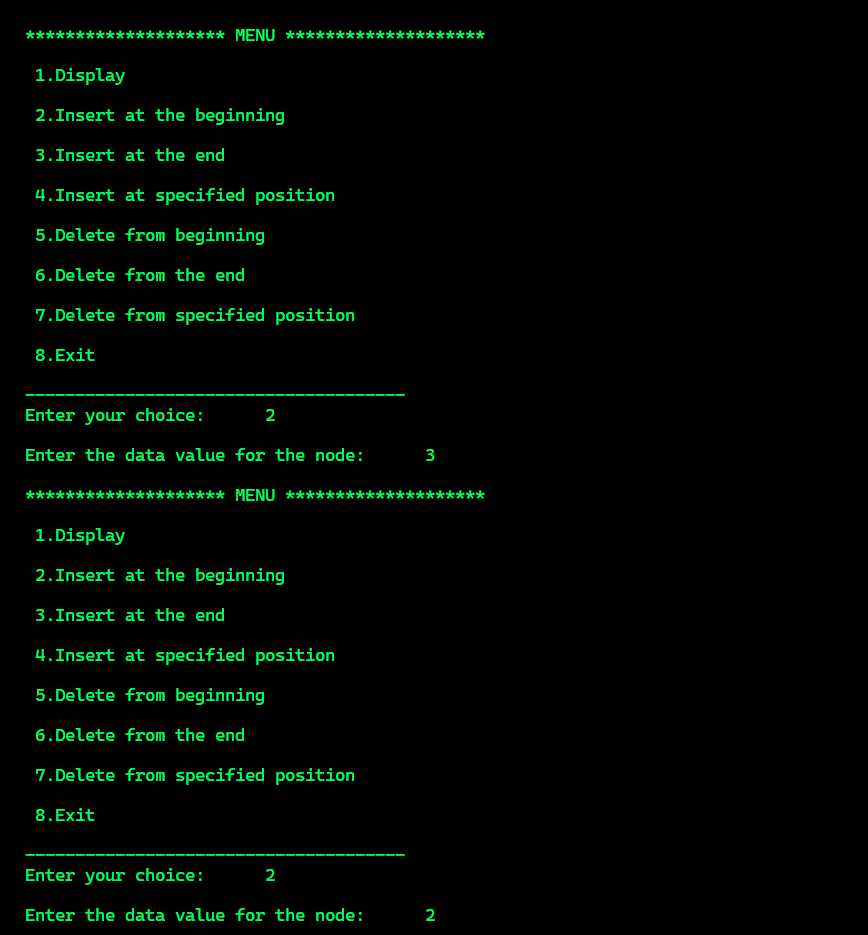
                                        break;

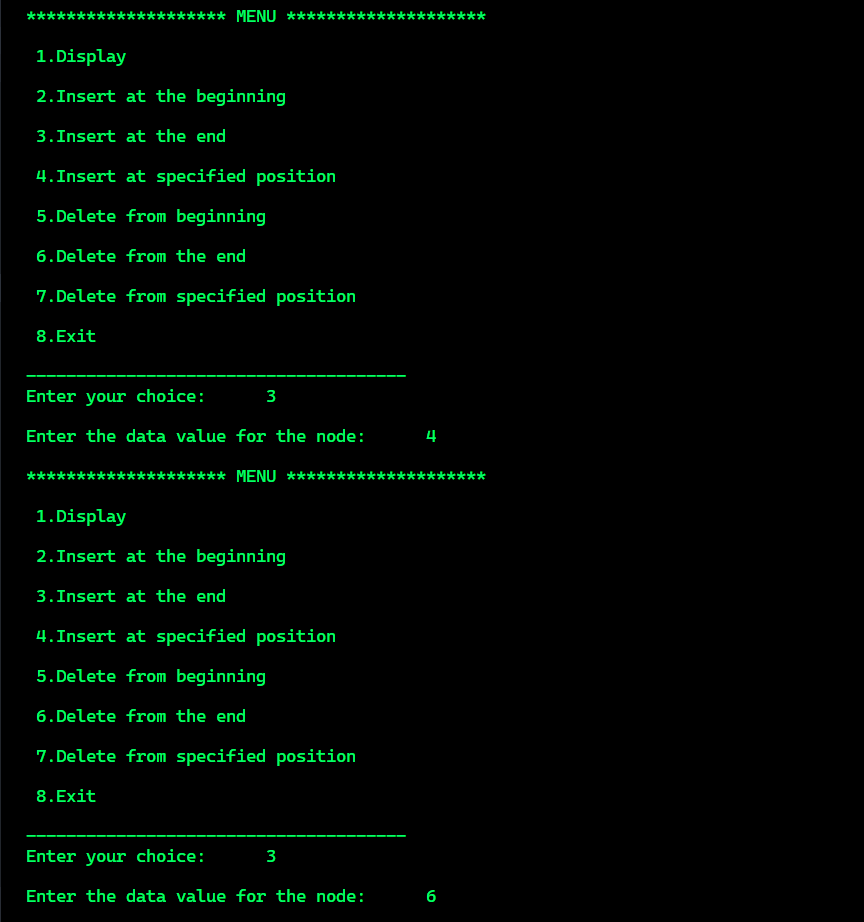
                }

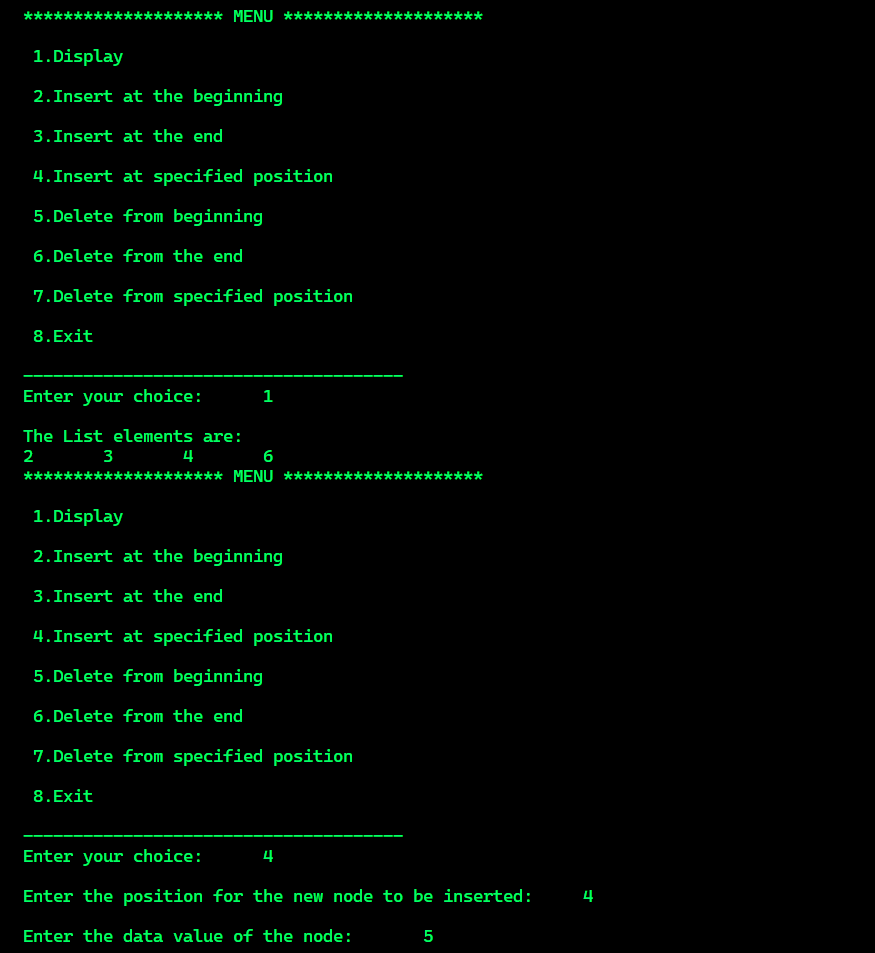
        }

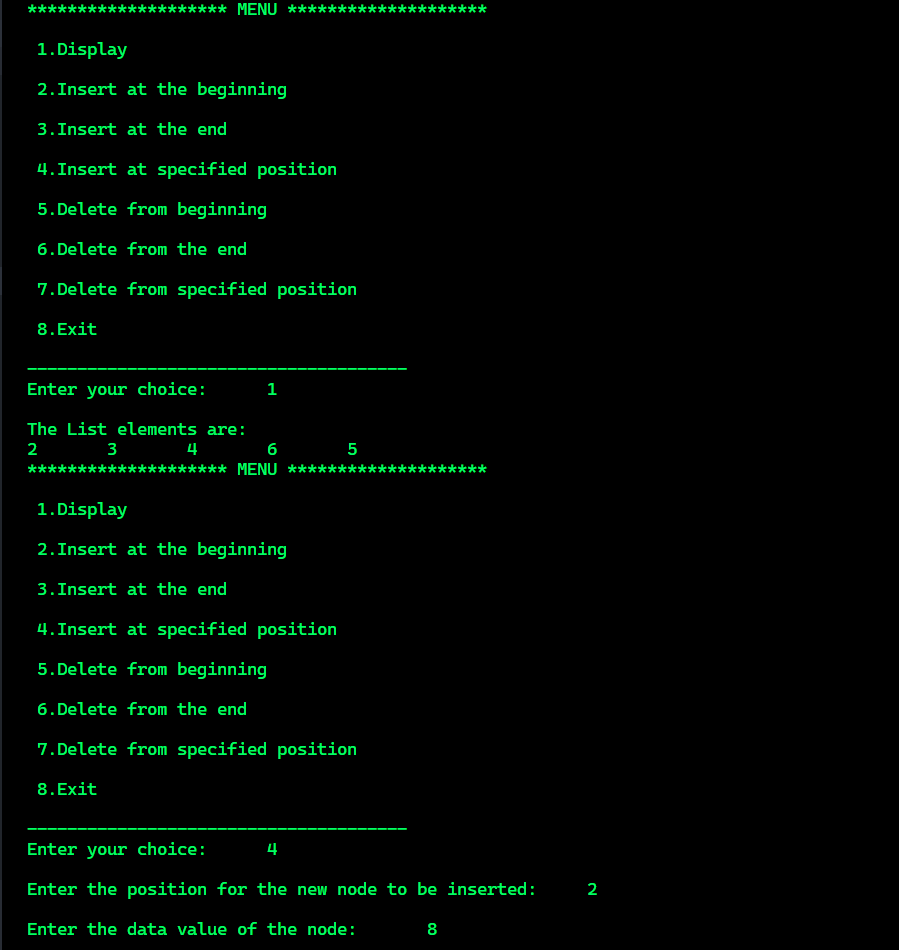
}

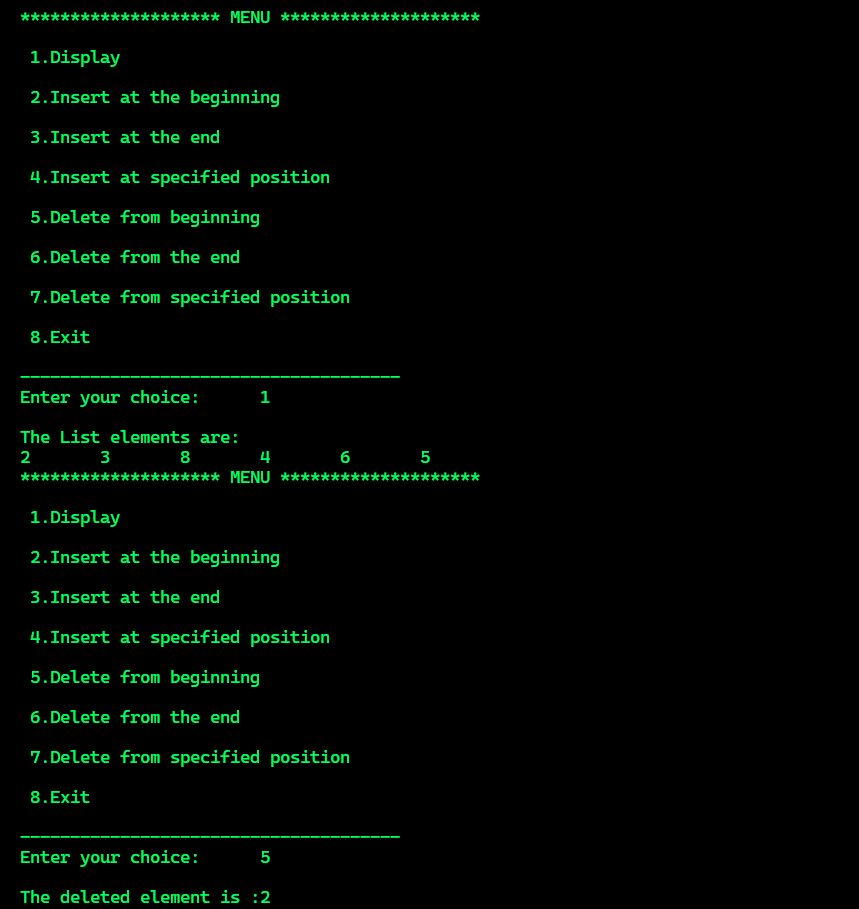
**OUTPUT**

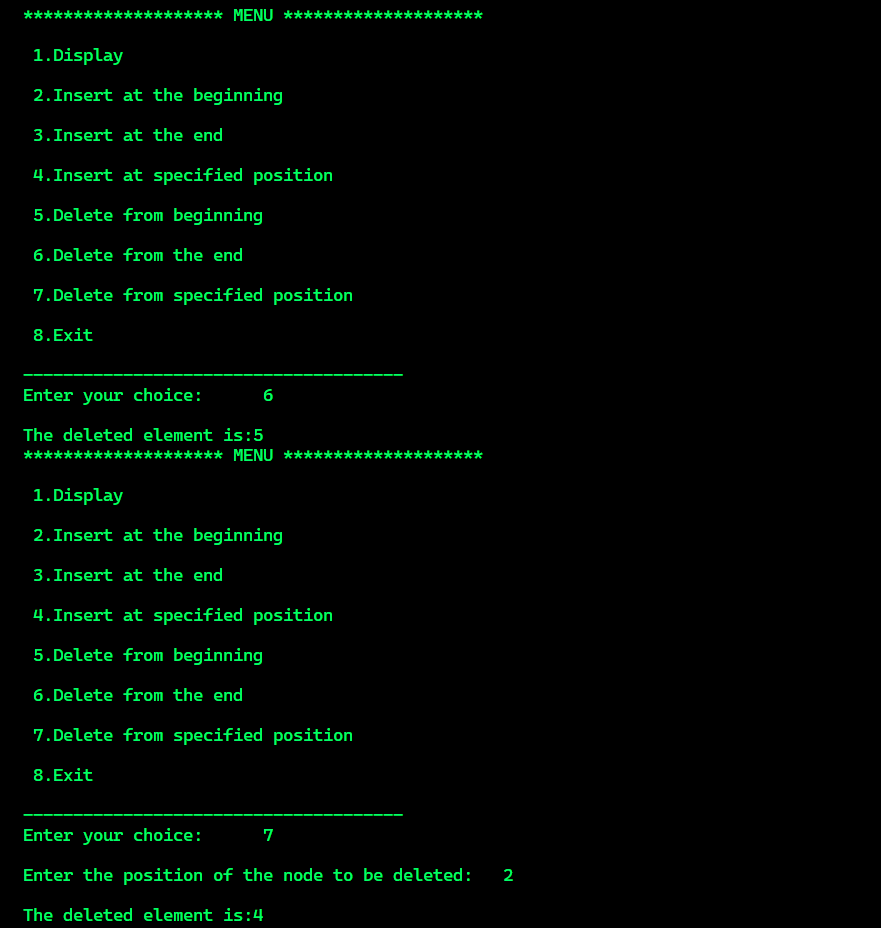
****

****

****

****

****

****