Course outcomes-1

**Program 1:**

Aim:-

Implementation of Stack using C (Array Data Structure)

Source code:-

#include <stdio.h>

int main()

{

    int a[30], i, n, o, top = -1, ITEM;

    char m;

    printf("enter the no. of terms:");

    scanf("%d", &n);

    do

    {

        printf("\nwhich operation do you want?\n1.PUSH\n2.POP\n3.DISPLAY\n4.SHOW TOP\n");

        scanf("%d", &o);

        //clrscr();

        if (o == 1)

            if (top >= n - 1)

                printf("stack is full\n");

            else

            {

                printf("enter the ITEM:");

                scanf("%d", &ITEM);

                top++;

                a[top] = ITEM;

            }

        else if (o == 2)

            if (top < 0)

                printf("stack is empty\n");

            else

            {

                ITEM = a[top];

                top--;

            }

        else if (o == 3)

        {

            if (top < 0)

                printf("stack is empty\n");

            else

            {

                printf("The stack is:\n");

                for (i = top; i >= 0; i--)

                    printf("%d\t", a[i]);

            }

        }

        else if (o == 4)

        {

            if (top < 0)

                printf("stack is empty\n");

            else

            {

                printf("Top elements is:%d\n",a[top]);

            }

        }

        else

            printf("wrong input\n");

        printf("\ndo you want to continue(y/n):");

        getchar();

        scanf("%c", &m);

        //clrscr();

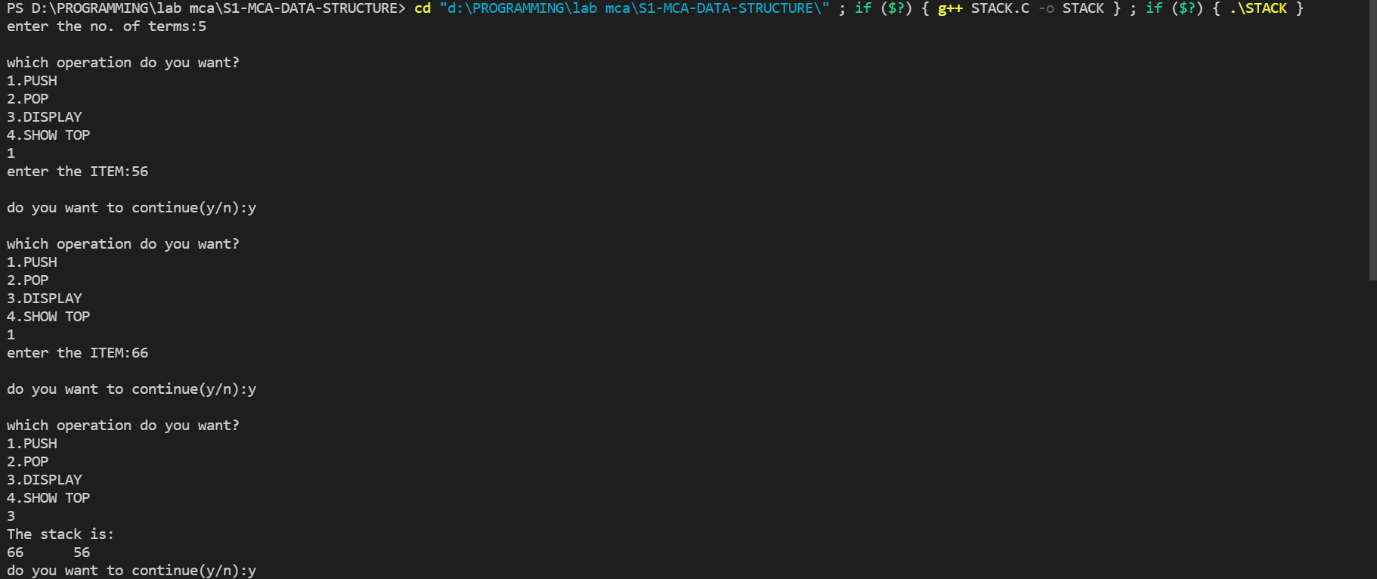
    } while (m == 'y' || m == 'Y');

    return 0;

}

Output:-





**Program 2:**

Aim:-

Implementation of Queue using C (Array Data Structure)

Source code:-

#include<stdio.h>

#define n 10

int queue[n];

int front=-1,rear=-1;

void insert();

int d\_ele();

int peek();

void display();

int main(){

int op,val;

do{

printf("\nmenu\n");

printf("\n1.Insert an elements\n2.Delete an element\n3.Peek(Show front elements)\n4.Display\n5.Exit\n");

scanf("%d",&op);

switch(op)

{

case 1:insert();break;

case 2:val=d\_ele();

        if(val==-1){

          printf("Oops,Somethng went wrong!");

          }

         else

         {

          printf("Element deleted value is%d",val);

         }

       break;

case 3:val=peek();

         if(val!=-1){

          printf("The peek element is %d",val);

          }

          else{ printf("Oops,Somethng went wrong!");

             }break;

case 4:display();break;

}

}while(op!=5);

return 0;

}

void insert()

{

int num;

printf("Enter number to be inserted in the queue");

scanf("%d",&num);

if(rear==n-1)

printf("\noverflow");

else if(front==-1&&rear==-1)

front=rear=0;

else

rear++;

queue[rear]=num;

}

int d\_ele()

{

int val;

if(front==-1||front>rear)

{

printf("\nUnderflow");

return -1;

}

else

{

val=queue[front];

front++;

if(front>rear){

front=rear=-1;

}

return val;

}

}

int peek(){

if(front==-1||front>rear)

{

printf("Empty\n");

return -1;

}

else

{

return queue[front];

}

}

void display(){

int i;

if(front==-1||front>rear)

printf("\nEmpty");

else

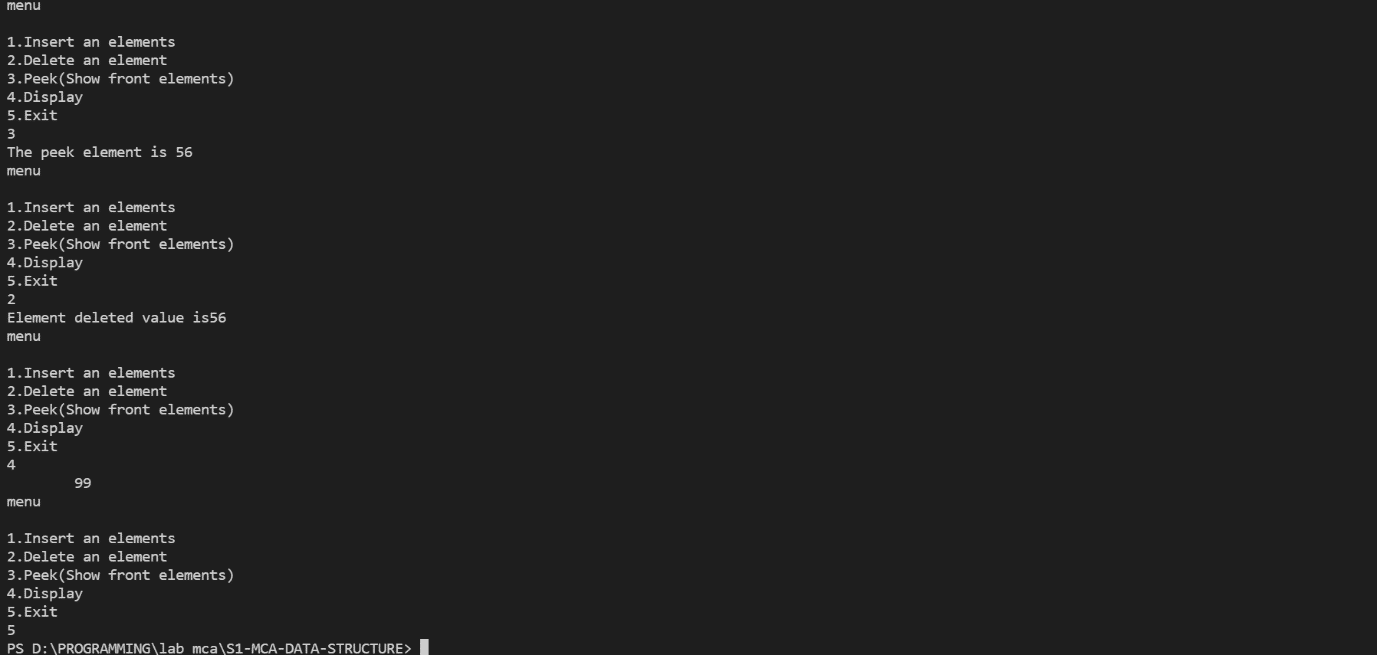
{

for(i=front;i<=rear;++i)

printf("\t%d",queue[i]);

}

}

Output:-



**Program 3:**

Aim:-

Implementation of Operations on Singly Linked List using C

Source code:-

#include<stdio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*next;

};

struct node \*start=NULL;

struct node \*ptr,\*tmp,\*kmp;

struct node \*create(){//create single node

struct node \*newnode;

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

printf("Enter Data");

scanf("%d",&newnode->data);

newnode->next=NULL;

return newnode;

}

void ifornt(){//front insertion

tmp=create();

ptr=start;

if(start==NULL){

start=tmp;

}

else{

tmp->next=ptr;

start=tmp;

}

}

void iend(){//end insertion

tmp=create();

ptr=start;

if(ptr==NULL){

start=tmp;

}

else{

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=tmp;}

}

void iposition(){//given posirion

int pos;

printf("\nEnter data after the data has to be inerted\n");

scanf("%d",&pos);

tmp=create();

ptr=start;

while(ptr->data!=pos){

ptr=ptr->next;

}

tmp->next=ptr->next;

ptr->next=tmp;

}

void dfront(){//deletion at front

ptr=start;

if(ptr==NULL){

printf("Underflow");

}

else

start=start->next;

}

void dend(){//del at end

ptr=start;

if(ptr==NULL){

printf("Underflow");

}

else

{

while(ptr->next!=NULL){

tmp=ptr;

ptr=ptr->next;

}tmp->next=NULL;

}

}

void dspc(){//del at end

int xz;

ptr=start;

if(ptr==NULL){

printf("Underflow");

}

else

{

    printf("ENter data to delete?\n");

    scanf("%d",&xz);

while(ptr->data!=xz){

tmp=ptr;

ptr=ptr->next;

}tmp->next=ptr->next;

}

}

void displ(){//fn to disply

ptr=start;

while(ptr->next!=NULL){

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

ptr=ptr->next;

}

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

}

void main(){

int o1,o2;

do{

printf("\nEnter The operation to perform\n");

printf("1.Insertion\n2.Delesion\n3.Display\n4.Exit\n");

scanf("%d",&o1);

switch(o1){

case 1: //insertion

    h1:printf("\nSelect one of the following\n");

    printf("1.Insertion at front\n2.Insersion at end\n3.Insertion After specific data\n");

    scanf("%d",&o2);

    if(o2==1){//front

    ifornt();

                 }

    else if(o2==2){//end

            iend();

            }

    else if(o2==3){//position

iposition();

            }

    else {printf("\nEnter a valid coice\n");goto h1;

        }break;

case 2://Delesion

    h3:printf("\nSelect one of the following\n");

    printf("1.Delesion at front\n2.Delesion at end\n3.Delesion of specific data\n");

    scanf("%d",&o2);

    if(o2==1){//front

               dfront();

                 }

    else if(o2==2){//end

    dend();

            }

    else if(o2==3){//position

    dspc();

            }

    else {printf("\nEnter a valid coice\n");goto h3;

        }break;

case 3://display

       displ();

    break;

case 4:break;

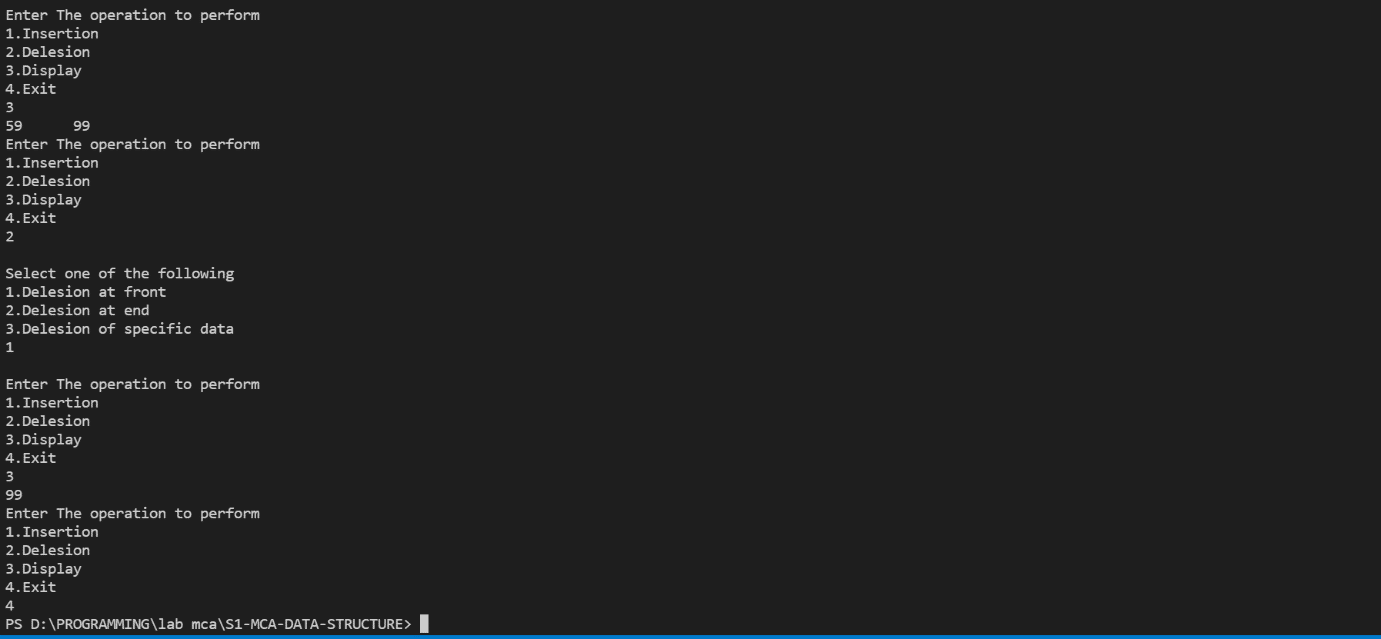
default:printf("Enter a valid choice");

}

}while(o1!=4);

}

Output:-





**Program 4:**

Aim:-

Implementation of Operations on Doubly Linked List using C

Source code:-

#include<stdio.h>

#include<stdlib.h>

struct node

{

    struct node\*prev;

    int data;

    struct node\*next;

};

struct node\*header,\*temp,\*newnode,\*ptr,\*preptr;

void begin();

void af\_pos();

void end();

void del\_begin();

void del\_af\_pos();

void del\_end();

void display();

int main()

{

    int c=0,option;

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

    header->next=NULL;

    header->prev=NULL;

    printf("\n\*\*\*\* Doubly Linked List \*\*\*\*\n");

    while(c==0)

    {

        printf("\n\*\*\*\* Main Menu \*\*\*\*\n");

        printf("1. Insert at begining\n");

        printf("2. Insert after a data\n");

        printf("3. Insert at end\n");

        printf("4. Delete from begining\n");

        printf("5. Delete after a data\n");

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

        printf("7. Display list\n");

        printf("Enter your option : ");

        scanf("%d",&option);

        switch(option)

        {

        case 1:begin();

            break;

        case 2:af\_pos();

            break;

        case 3:end();

            break;

        case 4:del\_begin();

            break;

        case 5:del\_af\_pos();

            break;

        case 6:del\_end();

            break;

        case 7:display();

            break;

        default:printf("Invalid Operator");

        }

        printf("Do you want to continue(0/1) : ");

        scanf("%d",&c);

    }

    return 0;

}

void begin()

{

    int val;

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

    printf("Enter the data : ");

    scanf("%d",&val);

    newnode->data=val;

    newnode->prev=header;

    newnode->next=header->next;

    header->next=newnode;

}

void af\_pos()

{

    int dat,val;

    ptr=header;

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

    printf("Enter the data of the node after the new node has to be placed : ");

    scanf("%d",&dat);

    printf("Enter the data of the new node : ");

    scanf("%d",&val);

    while(ptr->data!=dat)

    {

        ptr=ptr->next;

    }

    newnode->data=val;

    newnode->next=ptr->next;

    newnode->prev=ptr;

    ptr->next->prev=newnode;

    ptr->next=newnode;

}

void end()

{

    int val;

    ptr=header;

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

    printf("Enter the data of the new node : ");

    scanf("%d",&val);

    while(ptr->next!=NULL)

    {

        ptr=ptr->next;

    }

    newnode->data=val;

    newnode->next=ptr->next;

    newnode->prev=ptr;

    ptr->next=newnode;

}

void del\_begin()

{

    ptr=header;

    if(header->next==NULL)

    {

        printf("\nUnderflow\n");

    }

    else

    {

        ptr=ptr->next;

        header->next=ptr->next;

        ptr->next->prev=header;

        printf("Data of node deleted : %d\n",ptr->data);

        free(ptr);

    }

}

void del\_af\_pos()

{

    int val;

    ptr=header;

    if(header->next==NULL)

    {

        printf("\nUnderflow\n");

    }

    else

    {

        printf("Enter the data of the node after which the node has to be deleted : ");

        scanf("%d",&val);

        while(ptr->data!=val)

        {

            ptr=ptr->next;

        }

        temp=ptr->next;

        ptr->next=temp->next;

        temp->next->prev=ptr;

        printf("Data of node deleted : %d\n",temp->data);

        free(temp);

    }

}

void del\_end()

{

    ptr=header;

    if(header->next==NULL)

    {

        printf("\nUnderflow\n");

    }

    else

    {

        while(ptr->next!=NULL)

        {

            ptr=ptr->next;

        }

        ptr->prev->next=ptr->next;

        printf("Data of node deleted : %d\n",ptr->data);

        free(ptr);

    }

}

void display()

{

    temp=header;

    if(header->next==NULL)

    {

        printf("\nUnderflow\n");

    }

    else

    {

        while(temp->next!=NULL)

        {

            temp=temp->next;

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

        }

        printf("\n");

    }

}

Output;-

