**QUEUES**

**Write a menu driven C program to implement Queues and Circular Queue using arrays and perform the following operation (i) Insert (ii)Delete (iii)is empty() (iv) Is full() (v) Display**

**PROGRAM**

#include<stdio.h>

#define MAX 5

int queue\_array[MAX];

int rear = - 1;

int front = - 1;

void insert()

{

    int add\_item;

    if (rear == MAX - 1)

    printf("Queue Overflow \n");

    else

    {

        if (front == - 1)

        front = 0;

        printf("Inset the element in queue : ");

        scanf("%d", &add\_item);

        rear = rear + 1;

        queue\_array[rear] = add\_item;

    }

}

void delete()

{

    if (front == - 1 || front > rear)

    {

        printf("Queue Underflow \n");

        return ;

    }

    else

    {

        printf("Element deleted from queue is : %d\n", queue\_array[front]);

        front = front + 1;

    }

}

void qdisplay()

{

    int i;

    if (front == - 1)

        printf("Queue is empty \n");

    else

    {

        printf("Queue is : \n");

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

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

        printf("\n");

    }

}

void qisempty()

{

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

    printf("\nQueue is empty");

    else

    printf("\nQueue is not empty");

}

void qisfull()

{

    if(rear==MAX-1)

    printf("\nQueue is full");

    else

    printf("\nQueue is not full");

}

// Check if the circular queue is full

int isFull() {

  if ((front == rear + 1) || (front == 0 && rear == MAX - 1)) return 1;

  return 0;

}

// Check if the circular queue is empty

int isEmpty() {

  if (front == -1) return 1;

  return 0;

}

// Adding an element

void enQueue() {

    int element;

    printf("\nEnter the element to be inserted:");

    scanf("%d",&element);

  if (isFull())

    printf("\n Queue is full!! \n");

  else

    {

    if (front == -1) front = 0;

    rear = (rear + 1) % MAX;

    queue\_array[rear] = element;

    printf("\n Inserted -> %d", element);

    }

}

// Removing an element

void deQueue() {

  int element;

  if (isEmpty()) {

    printf("\n Queue is empty !! \n");

    return (-1);

  } else {

    element = queue\_array[front];

    if (front == rear) {

      front = -1;

      rear = -1;

    }

    else {

      front = (front + 1) % MAX;

    }

    printf("\n Deleted element -> %d \n", element);

  }

}

// Display the queue

void display() {

  int i;

  if (isEmpty())

    printf(" \n Empty Queue\n");

  else {

    printf("\n Items -> ");

    for (i = front; i != rear; i = (i + 1) % MAX) {

      printf("%d ", queue\_array[i]);

    }

    printf("%d ", queue\_array[i]);

  }

}

void main()

{

    int choice,x,qchoice;

    printf("\n\nEnter the type of queue to be used:\n1. Normal Queue\n2. Circular queue\nYour choice:");

    scanf("%d",&qchoice);

    if(qchoice==1)

    {

        while (1)

        {

            printf("\n1. Insert \n");

            printf("2. Delete\n");

            printf("3. Display\n");

            printf("4. Isempty()\n");

            printf("5. Isfull()\n");

            printf("6.Quit \n");

            printf("Enter your choice : ");

            scanf("%d", &choice);

            switch (choice)

            {

                case 1:

                insert();

                break;

                case 2:

                delete();

                break;

                case 3:

                qdisplay();

                break;

                case 4:

                qisempty();

                break;

                case 5:

                qisfull();

                break;

                case 6:

                exit(0);

                break;

                default:

                printf("Wrong choice \n");

            }

        }

    }

    else if(qchoice==2)

    {

    while (1)

        {

            printf("\n1. Insert \n");

            printf("2. Delete\n");

            printf("3. Display\n");

            printf("4. Isempty()\n");

            printf("5. Isfull()\n");

            printf("6. Quit \n");

            printf("Enter your choice : ");

            scanf("%d", &choice);

            switch (choice)

            {

                case 1:

                enQueue();

                break;

                case 2:

                deQueue();

                break;

                case 3:

                display();

                break;

                case 4:

                x=isEmpty();

                if(x==1) printf("\nQueue is empty\n");

                else printf("\nQueue is not empty\n");

                break;

                case 5:

                x=isFull();

                if(x==1) printf("\nQueue is full\n");

                else printf("\nQueue is not full\n");

                break;

                case 6:

                exit(0);

                break;

                default:

                printf("Wrong choice \n");

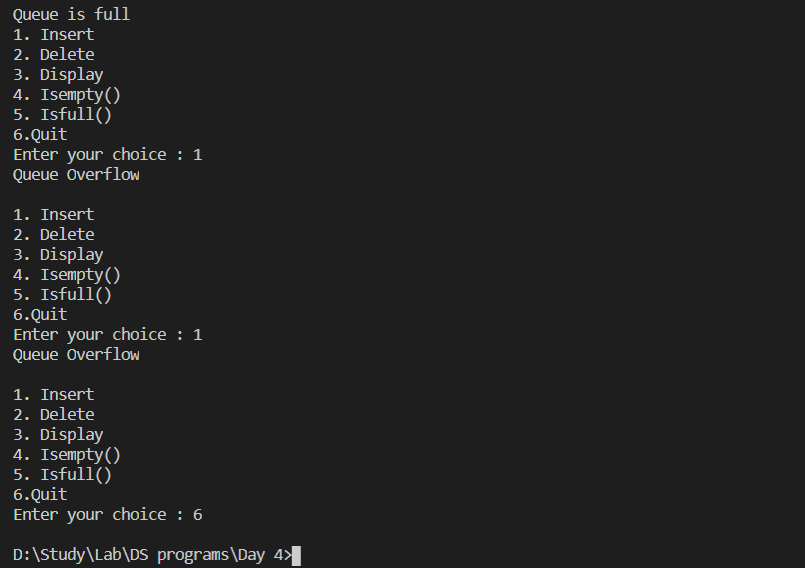
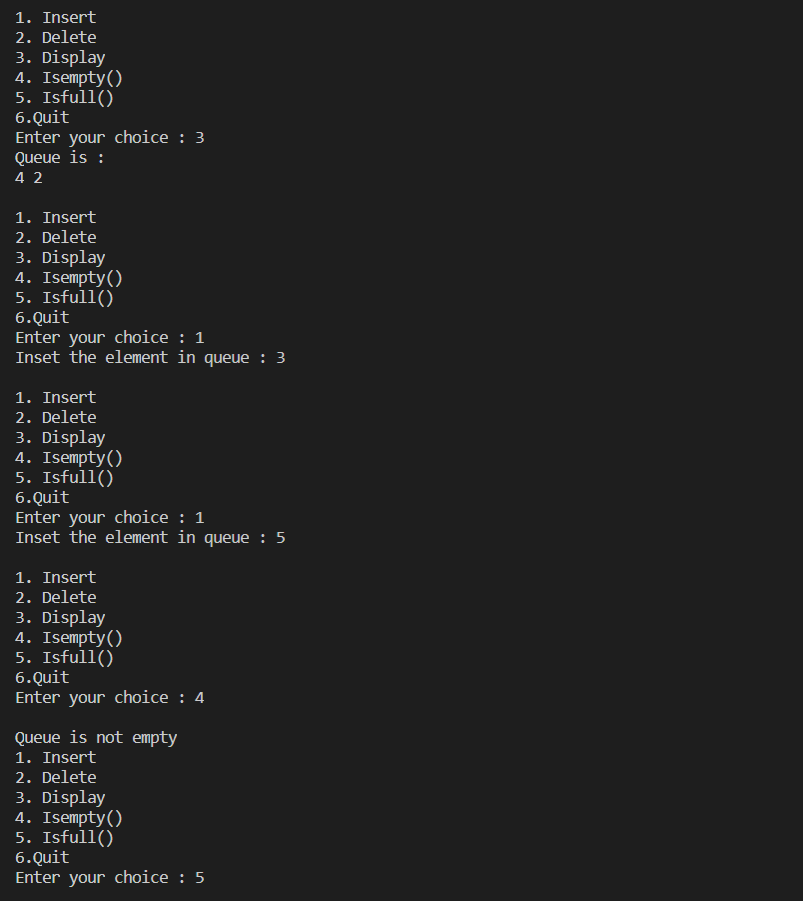
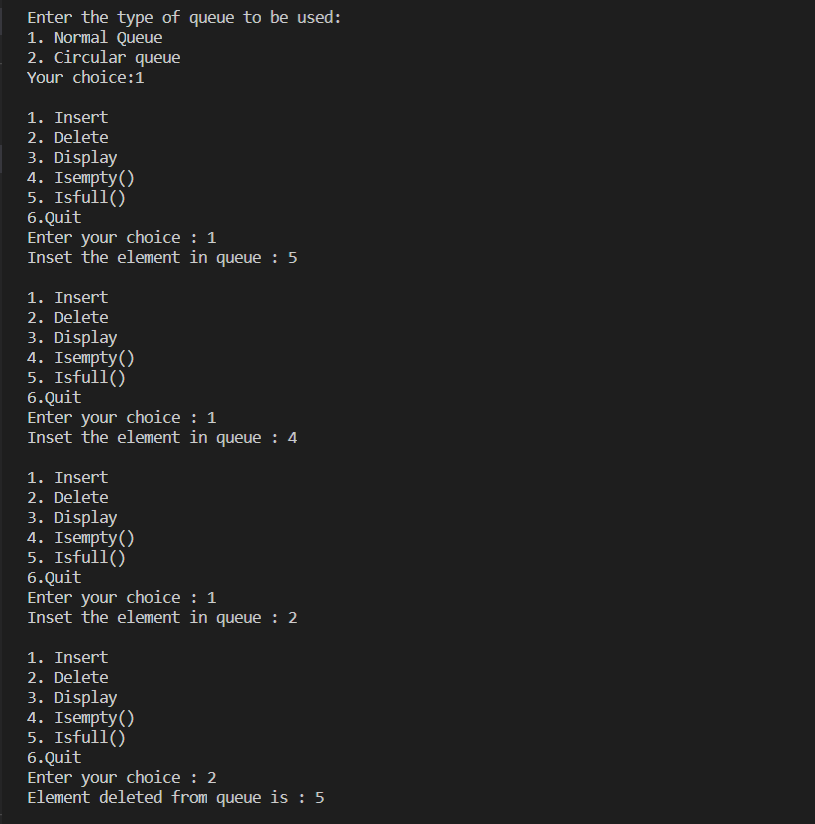
            }

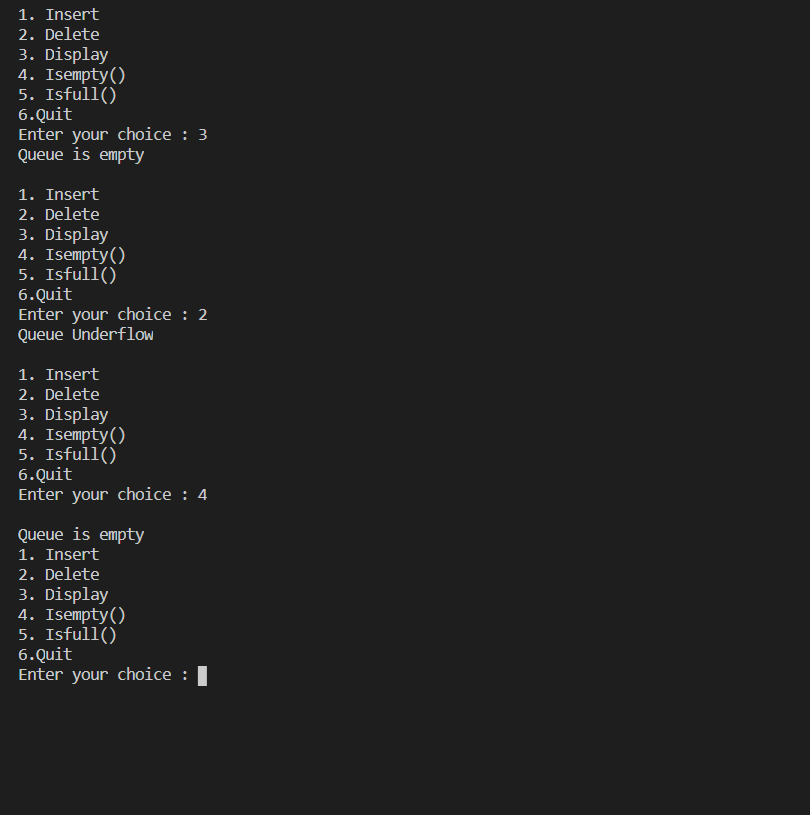
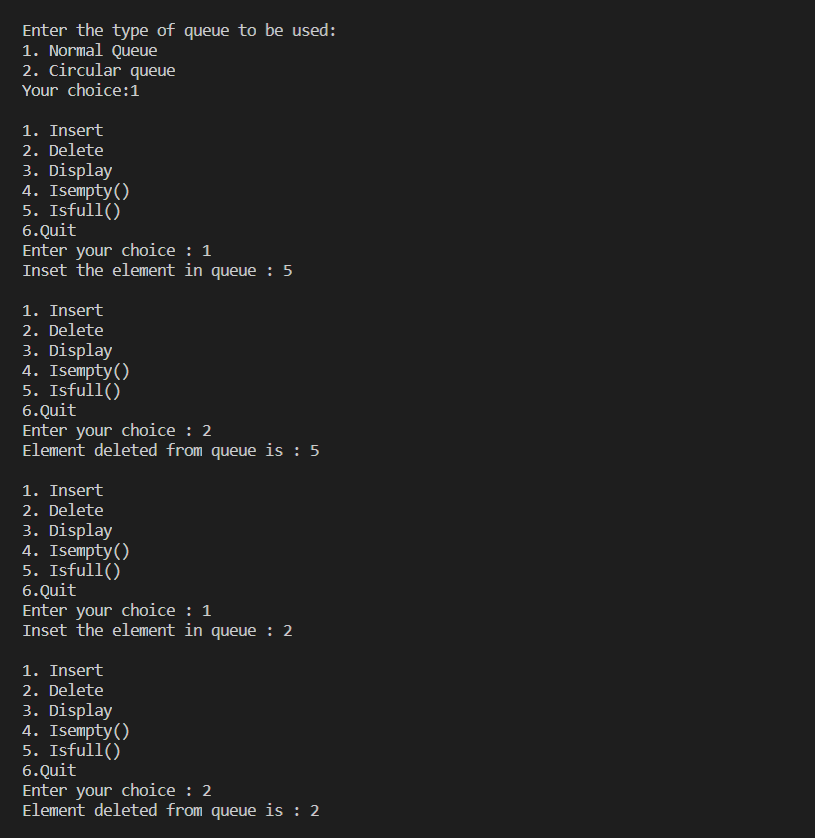
        }

    }

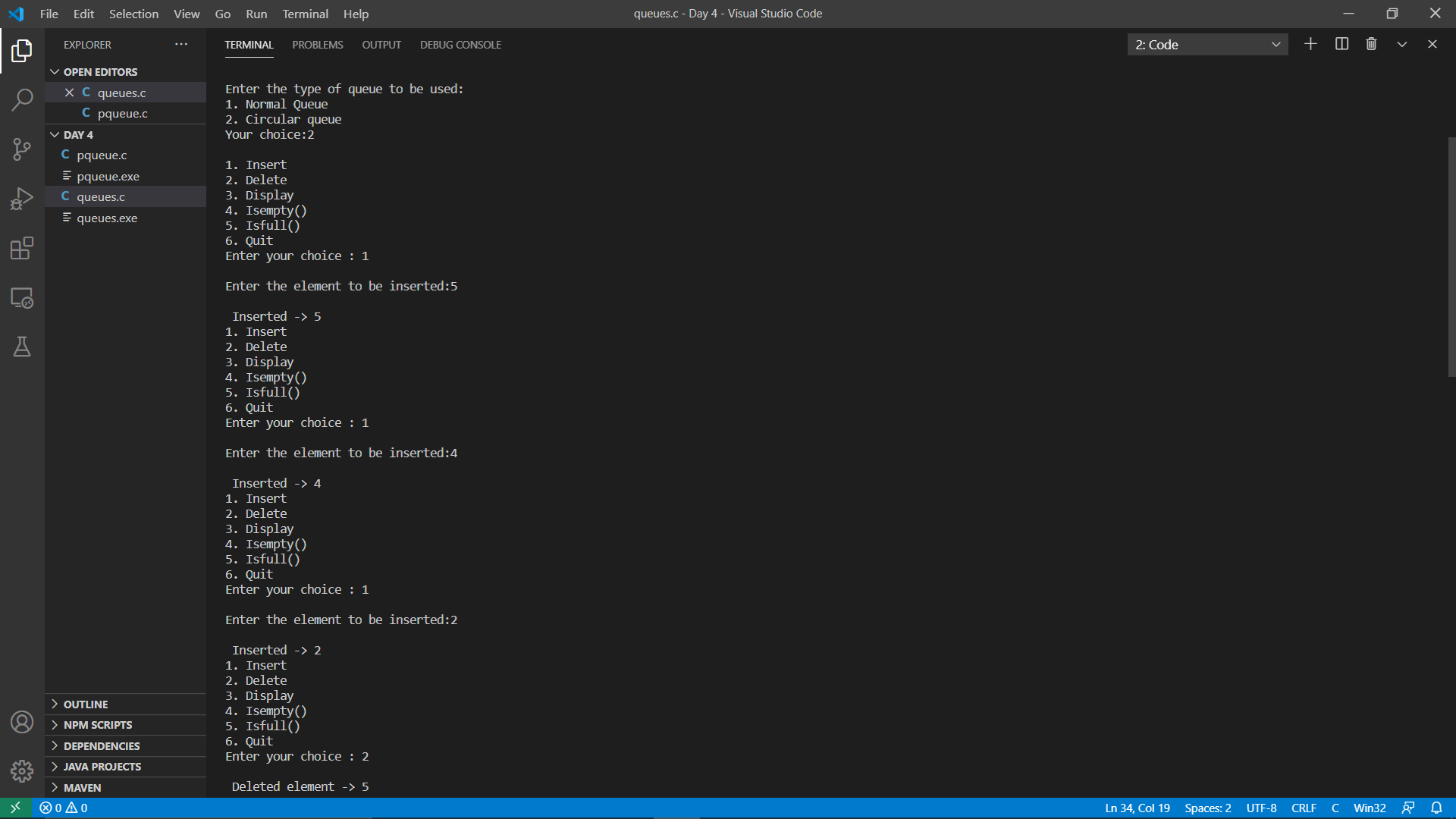
}

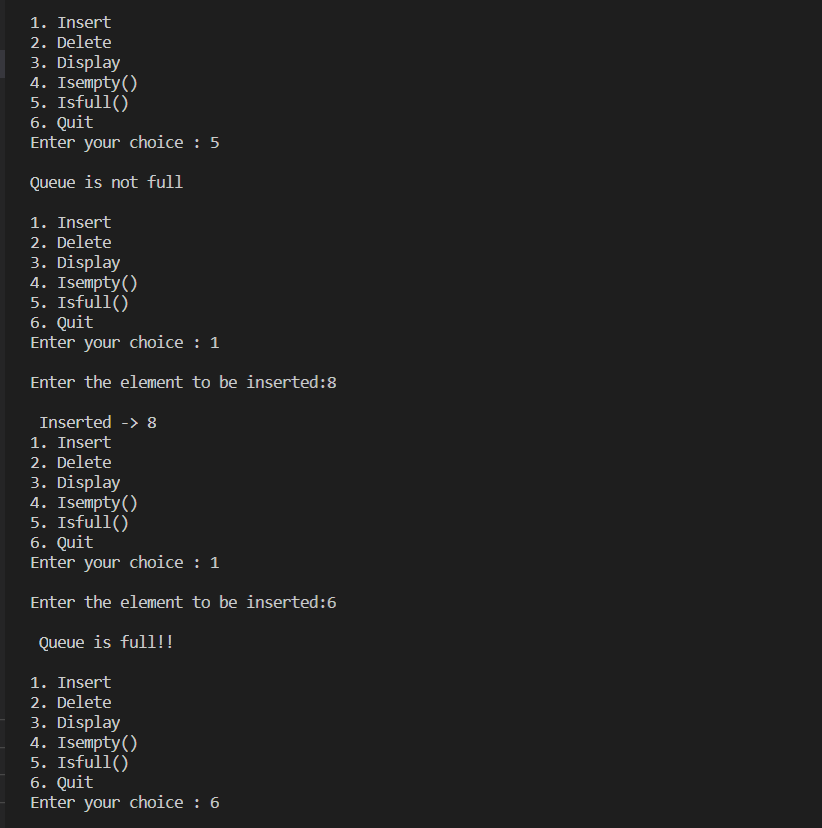
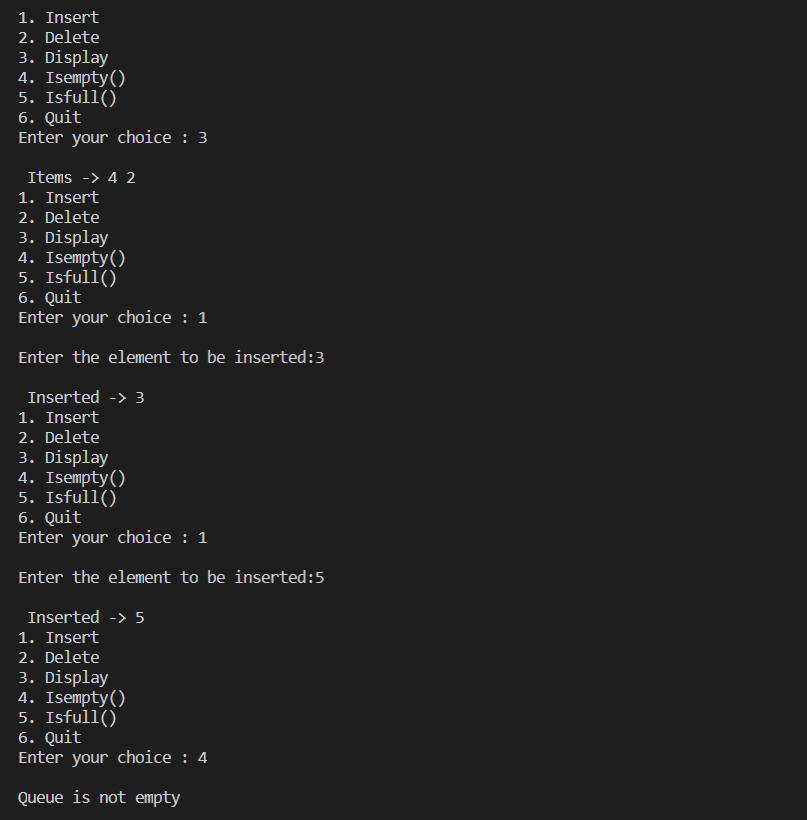
**OUTPUT**

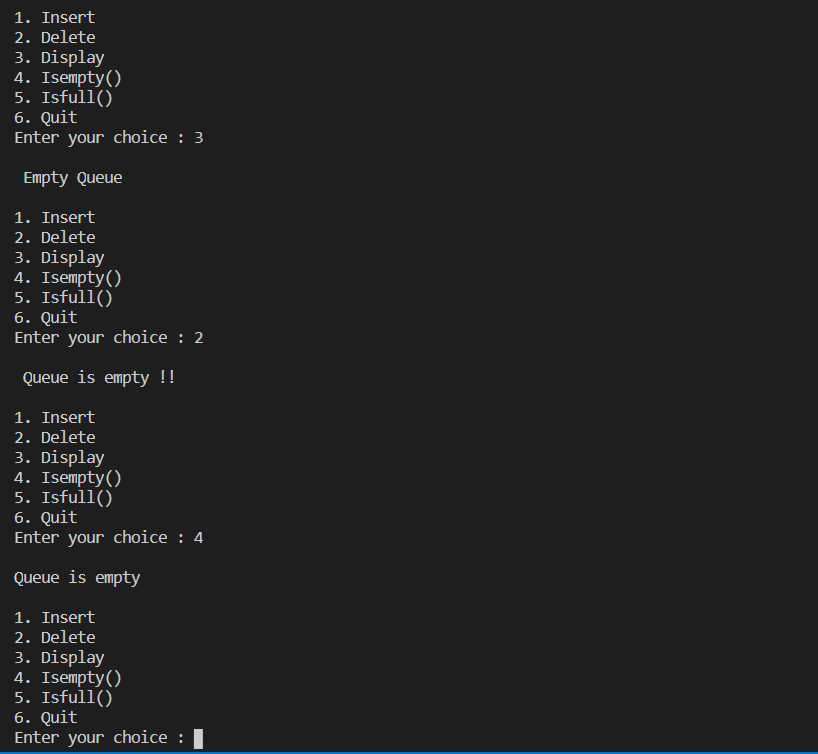
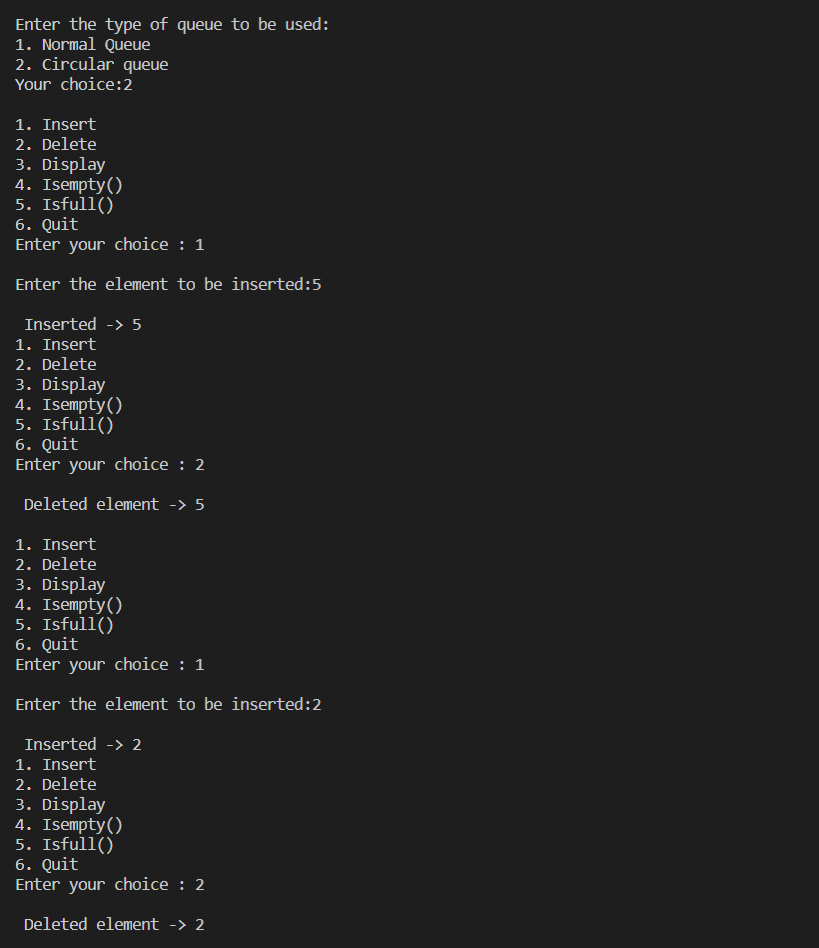
**TEST CASE 1:**

**TEST CASE 2:**

**TEST CASE 3:**

****

****

**TEST CASE 4:**

**Write a menu driven C program to implement Priority Queues using arrays.**

**PROGRAM**

#include<stdio.h>

#define N 5

int Q[N],Pr[N];

int r = -1,f = -1;

void enqueue(int data,int p)

{

    int i;

    if(r==N-1)

        printf("Queue is full");

    else

    {

        if(f==-1)

        {

            f = r = 0;

            Q[r] = data;

            Pr[r] = p;

        }

        else

        {

            for(i = r;i>=f;i--)

            {

                if(p>Pr[i])

                {

                    Q[i+1] = Q[i];

                    Pr[i+1] = Pr[i];

                }

                else

                    break;

            }

            Q[i+1] = data;

            Pr[i+1] = p;

            r++;

        }

    }

}

void print() //print the data of Queue

{

int i;

    for(i=f;i<=r;i++)

    {

        printf("\nElement = %d\tPriority = %d",Q[i],Pr[i]);

    }

}

int dequeue() //remove the data from front

{

    if(f == -1)

    {

        printf("Queue is Empty");

    }

    else

    {

        printf("\ndeleted Element = %d\t Its Priority = %d",Q[f],Pr[f]);

        if(f==r)

            f = r = -1;

        else

            f++;

    }

}

void main()

{

    int opt,data,p;

    do{

        printf("\n\n1. Insert the Data\n2. Show the Data\n3. Delete the data\n0. Exit");

        printf("\nEnter Your Choice:-");

        scanf("%d",&opt);

        switch(opt){

            case 1:

                printf("\nEnter your data and Priority of data:-");

                scanf("%d %d",&data,&p);

                enqueue(data,p);

                break;

            case 2:

                print();

                break;

            case 3:

                 dequeue();

                break;

            case 0:

                break;

            default:

                printf("\nIncorrect Choice");

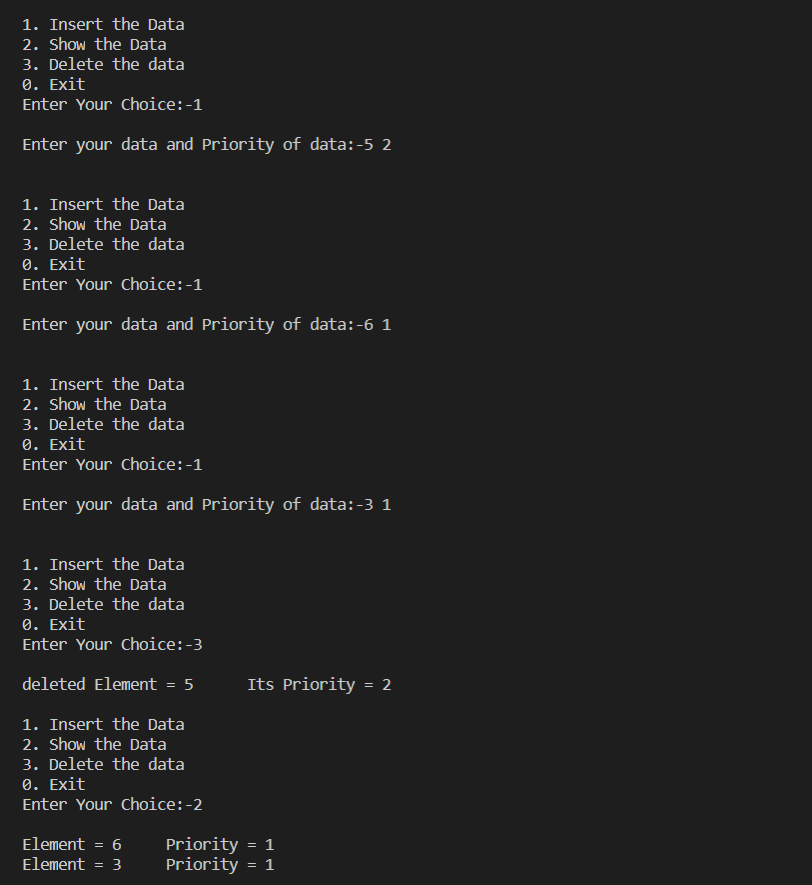
        }

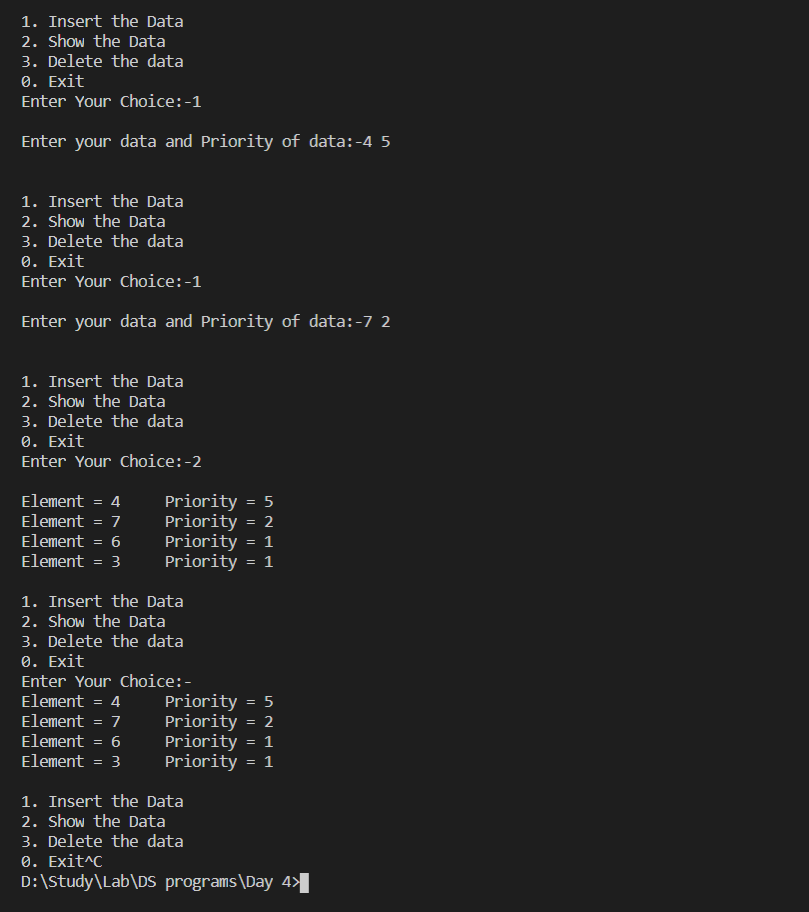
    }while(opt!=0);

}

**OUTPUT**

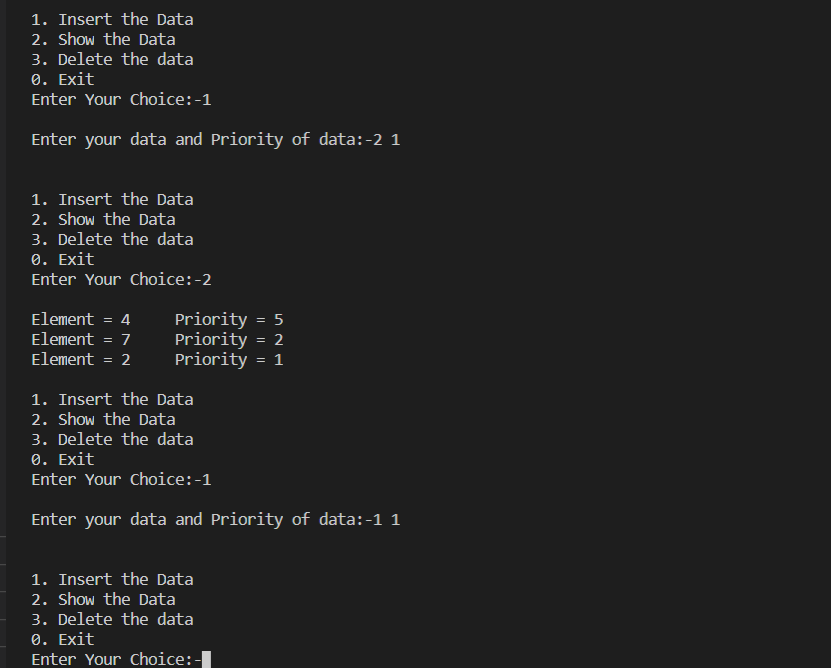
**TEST CASE 1:**

****

****

**TEST CASE 2:**

****

****