Assignment 2(Queue)

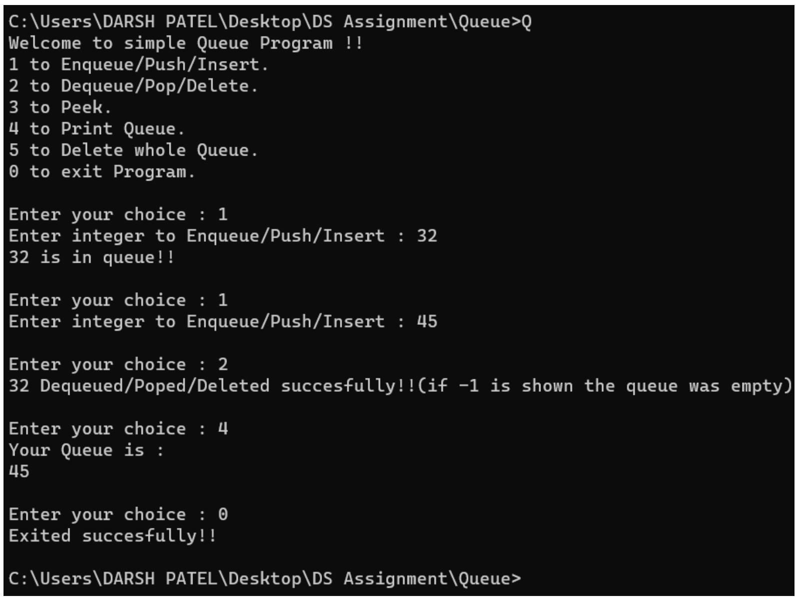
1) Implement Queue & it's function -Enqueue, Dequeue :

CODE :

```c
#include <stdio.h>
#define MAX_SIZE 100

// Global Variables
int queue[100];
int front = -1;
int rear = -1;

int isEmpty()
{
    if (front == -1 && rear == -1)
    {
        return 1;
    }
    return 0;
}

int isFull()
{
    if (rear == MAX_SIZE - 1)
    {
        return 1;
    }
    return 0;
}

void enqueue(int data)
{
    if (isFull())
    {
        printf("Queue is Full!!\n");
        return;
    }
    if (isEmpty())
    {
        front++;
        rear++;
        queue[rear] = data;
        printf("%d is in queue!!\n",data);
        return;
    }
    queue[++rear] = data;
}

int dequeue()
{
    if (isEmpty())
    {
        printf("Queue is Empty!!\n");
        return -1;
    }
    if (front == rear)
    {
        int a = queue[front];
        front = rear = -1;
        return a;
    }

    return queue[front++];
}

int peek()
{
    if (isEmpty())
        return -1;

    return queue[front];
}

void printQ()
{
    if (isEmpty())
    {
        printf("The queue is empty nothing to display!\n");
        return;
    }
    printf("Your Queue is :\n");
    for (int i = front; i <= rear; i++)
    {
        printf("%d\n",queue[i]);
    }
}

void deleteQ()
{
    front = -1;
    rear = -1;
    if (isEmpty())
    {
```

```c
        printf("Queue deleted successfully!!\n");
    }
}

int main(int argc, char const *argv[])
{
    int ch, data;
    printf("Welcome to simple Queue Program !!\n");
    printf("1 to Enqueue/Push/Insert.\n");
    printf("2 to Dequeue/Pop/Delete.\n");
    printf("3 to Peek.\n");
    printf("4 to Print Queue.\n");
    printf("5 to Delete whole Queue.\n");
    printf("0 to exit Program.\n");

loop:
    printf("\nEnter your choice : ");
    scanf("%d", &ch);
    switch (ch)
    {
    case 1:
        printf("Enter integer to Enqueue/Push/Insert : ");
        scanf("%d", &data);
        enqueue(data);
        goto loop;
    case 2:
        printf("%d Dequeued/Poped/Deleted succesfully!!(if -1 is shown the
queue was empty)\n", dequeue());
        goto loop;
    case 3:
        printf("%d Peeked succesfully!!(if -1 is shown the queue was
empty)\n", peek());
        goto loop;
    case 4:
        printQ();
        goto loop;
    case 5:
        deleteQ();
        goto loop;
    case 0:
        printf("Exited succesfully!!\n");
        break;
    default:
        printf("Error try again!!\n");
        goto loop;
    }
    return 0;
}
```

OUTPUT :



```
C:\Users\DARSH PATEL\Desktop\DS Assignment\Queue>Q
Welcome to simple Queue Program !!
1 to Enqueue/Push/Insert.
2 to Dequeue/Pop/Delete.
3 to Peek.
4 to Print Queue.
5 to Delete whole Queue.
0 to exit Program.

Enter your choice : 1
Enter integer to Enqueue/Push/Insert : 32
32 is in queue!!

Enter your choice : 1
Enter integer to Enqueue/Push/Insert : 45

Enter your choice : 2
32 Dequeued/Poped/Deleted succesfully!!(if -1 is shown the queue was empty)

Enter your choice : 4
Your Queue is :
45

Enter your choice : 0
Exited succesfully!!

C:\Users\DARSH PATEL\Desktop\DS Assignment\Queue>
```

2) Implement Circular Queue :

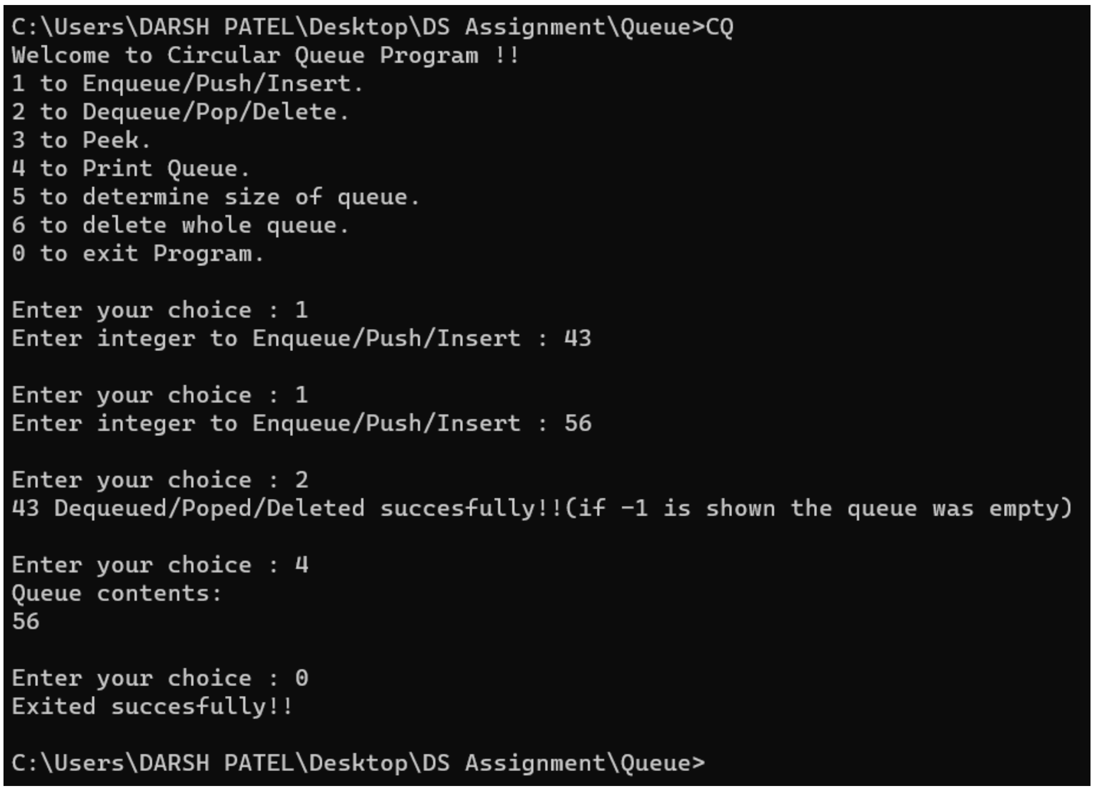
CODE :

```
#include <stdio.h>
#define SIZE 5

// Global Variables
int cQueue[SIZE];
int front = -1;
int rear = -1;
int size = 0;

int isEmpty()
{
    return (rear == -1 && front == -1);
}

int inc(int op)
{
    return (op + 1) % SIZE;
}

int isFull()
{
    return (inc(rear) == front );
}

void enqueue(int data)
{
    if (isFull())
    {
        printf("The is Full!!\n");
        return;
    }

    if (isEmpty())
    {
        front = inc(front);
        rear = inc(rear);
        cQueue[rear] = data;
        size++;
        return;
    }
    rear = inc(rear);
    cQueue[rear] = data;
    size++;
}

int dequeue()
{
    if (isEmpty())
    {
        printf("The queue is empty\n");
        return -1;
    }
    if (front == rear)
    {
        int a = cQueue[front];
        front = -1;
        rear = -1;
        size--;
        return a;
    }
    int a = front;
    front = inc(front);
    size--;
    return cQueue[a];
}

int peek()
{
    if (isEmpty())
    {
        printf("The queue is empty\n");
        return -1;
    }
    return cQueue[front];
}
```

```
void printQ()
{
    if (isEmpty())
    {
        printf("The queue is empty nothing to diplay!\n");
        return;
    }
    printf("Queue contents: \n");
    while (!isEmpty())
    {
        printf("%d\n",dequeue());
    }
}

void delete()
{
    for (int i = 0; i < SIZE; i++)
    {
        cQueue[i]=0;
    }

    front = -1;
    rear = -1;
    printf("Deleted queue succesfully!\n");
}

int main(int argc, char const *argv[])
{
    int ch, data;
    printf("Welcome to Circular Queue Program !!\n");
    printf("1 to Enqueue/Push/Insert.\n");
    printf("2 to Dequeue/Pop/Delete.\n");
    printf("3 to Peek.\n");
    printf("4 to Print Queue.\n");
    printf("5 to determine size of queue.\n");
    printf("6 to delete whole queue.\n");
    printf("0 to exit Program.\n");

loop:
    printf("\nEnter your choice : ");
    scanf("%d", &ch);
    switch (ch)
    {
    case 1:
        printf("Enter integer to Enqueue/Push/Insert : ");
        scanf("%d", &data);
        enqueue(data);
        goto loop;
```

```c
    case 2:
        printf("%d Dequeued/Poped/Deleted succesfully!!(if -1 is shown the
queue was empty)\n", dequeue());
        goto loop;
    case 3:
        printf("%d Peeked succesfully!!(if -1 is shown the queue was
empty)\n", peek());
        goto loop;
    case 4:
        printQ();
        goto loop;
    case 5:
        printf("The size is : %d\n", size);
        goto loop;
    case 6:
        delete();
        goto loop;
    case 0:
        printf("Exited succesfully!!\n");
        break;
    default:
        printf("Error try again!!\n");
        goto loop;
    }
    return 0;
}
```

OUTPUT :



```
C:\Users\DARSH PATEL\Desktop\DS Assignment\Queue>CQ
Welcome to Circular Queue Program !!
1 to Enqueue/Push/Insert.
2 to Dequeue/Pop/Delete.
3 to Peek.
4 to Print Queue.
5 to determine size of queue.
6 to delete whole queue.
0 to exit Program.

Enter your choice : 1
Enter integer to Enqueue/Push/Insert : 43

Enter your choice : 1
Enter integer to Enqueue/Push/Insert : 56

Enter your choice : 2
43 Dequeued/Poped/Deleted succesfully!!(if -1 is shown the queue was empty)

Enter your choice : 4
Queue contents:
56

Enter your choice : 0
Exited succesfully!!

C:\Users\DARSH PATEL\Desktop\DS Assignment\Queue>
```

3) Implement Priority Queue :

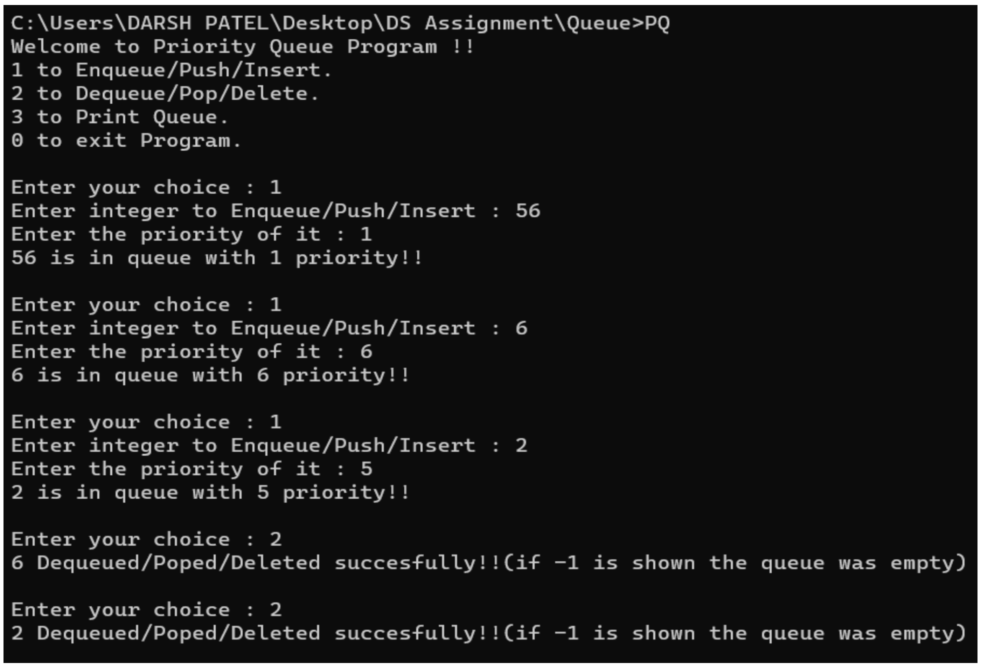
CODE :

```
#include <stdio.h>
#define SIZE 10

struct PQ
{
    int data;
    int priority;
} pq[SIZE];

int rear = -1;

int isEmpty()
{
    return (rear == -1);
}

int isFull()
{
    return (rear == SIZE - 1);
}

void enqueue(int data, int priority)
{
    if (isFull())
    {
        printf("Queue is Full!!\n");
        return;
    }
    rear++;
    pq[rear].data = data;
    pq[rear].priority = priority;
    printf("%d is in queue with %d priority!!\n", pq[rear].data,
pq[rear].priority);
}

int highestPriority()
{
    int i, p = -1;
    if (!isEmpty())
    {
        for (i = 0; i <= rear; i++)
        {
            if (pq[i].priority >= p)
            {
                p = pq[i].priority;
```

```c
            }
        }
    }
    return p;
}

int dequeue()
{
    if (isEmpty())
    {
        printf("The queue is empty!!\n");
        return -1;
    }
    int i, j, p, data;
    p = highestPriority();
    for (i = 0; i <= rear; i++)
    {
        if (pq[i].priority == p)
        {
            data = pq[i].data;
            break;
        }
    }
    if (i < rear)
    {
        for (j = i; j <= rear; j++)
        {
            pq[j].data = pq[j + 1].data;
            pq[j].priority = pq[j + 1].priority;
        }
    }
    rear = rear - 1;
    return data;
}

void printQ()
{
    if (isEmpty())
    {
        printf("The queue is empty nothing to display!\n");
        return;
    }
    printf("Your Priority Queue is :\n");
    for (int i = 0; i <= rear; i++)
    {
        printf("Data : %d, Priority : %d\n", pq[i].data, pq[i].priority);
    }
}
```

```c
int main(int argc, char const *argv[])
{
    int ch, data, priority;
    printf("Welcome to Priority Queue Program !!\n");
    printf("1 to Enqueue/Push/Insert.\n");
    printf("2 to Dequeue/Pop/Delete.\n");
    printf("3 to Print Queue.\n");
    printf("0 to exit Program.\n");

loop:
    printf("\nEnter your choice : ");
    scanf("%d", &ch);
    switch (ch)
    {
    case 1:
        printf("Enter integer to Enqueue/Push/Insert : ");
        scanf("%d", &data);
        printf("Enter the priority of it : ");
        scanf("%d", &priority);
        enqueue(data, priority);
        goto loop;
    case 2:
        printf("%d Dequeued/Poped/Deleted succesfully!!(if -1 is shown the
queue was empty)\n", dequeue());
        goto loop;
    case 3:
        printQ();
        goto loop;
    case 0:
        printf("Exited succesfully!!\n");
        break;
    default:
        printf("Error try again!!\n");
        goto loop;
    }

    return 0;
}
```

OUTPUT :



```
C:\Users\DARSH PATEL\Desktop\DS Assignment\Queue>PQ
Welcome to Priority Queue Program !!
1 to Enqueue/Push/Insert.
2 to Dequeue/Pop/Delete.
3 to Print Queue.
0 to exit Program.

Enter your choice : 1
Enter integer to Enqueue/Push/Insert : 56
Enter the priority of it : 1
56 is in queue with 1 priority!!

Enter your choice : 1
Enter integer to Enqueue/Push/Insert : 6
Enter the priority of it : 6
6 is in queue with 6 priority!!

Enter your choice : 1
Enter integer to Enqueue/Push/Insert : 2
Enter the priority of it : 5
2 is in queue with 5 priority!!

Enter your choice : 2
6 Dequeued/Poped/Deleted succesfully!!(if -1 is shown the queue was empty)

Enter your choice : 2
2 Dequeued/Poped/Deleted succesfully!!(if -1 is shown the queue was empty)
```