Program

```c
#include <stdio.h>
#define size 5

struct pq
{
    int item;
    int priority;
} pq[size];

int front = -1, rear = -1, loc;

void enqueue(int item, int priority)
{
    int i;
    if (front == 0 && rear == size - 1)
    {
        printf("Queue Overflow");
    }
    else if (rear == -1 && front == -1)
    {
        front = rear = 0;
        pq[rear].item = item;
        pq[rear].priority = priority;
    }
    else if (rear == size - 1)
    {
        for (i = front; i <= rear; i++)
        {
            pq[i - 1] = pq[i];
        }
        front--;
        rear--;
        for (i = rear; i >= front; i--)
        {
            if (pq[i].priority <= priority)
            {
                break;
            }
        }
        loc = i + 1;
        for (i = rear; i >= loc; i--)
        {
            pq[i + 1] = pq[i];
        }
        pq[loc].item = item;
        pq[loc].priority = priority;
        rear++;
    }
    else
    {
        for (i = rear; i >= front; i--)
        {
            if (pq[i].priority <= priority)
            {
                break;
            }
        }
        loc = i + 1;
        for (i = rear; i >= loc; i--)
        {
```

Experiment No: 11
Date:

PRIORITY QUEUE

Aim:

To write a C program to create, insert, delete and display elements in a Priority Queue.

Algorithm:

1. Start
2. Define MAX=5
3. Create struct pq

```
 struct pq
{
 int item;
 int priority;
}
```

4. Create array of structure pq[size]
5. Set front=-1,rear=-1
6. Create function enqueue

```
initialize integer variable i

If front is 0 and rear is size-1
       print "Queue Overflow"
Else if rear is -1 and front is -1
       front=rear=0
       pq[rear].item=item
       pq[rear].priority=priority

Else if rear is size-1
       for i from front to rear
           pq[i-1]=pq[i]
       end for
 front--
 rear--
       for i from rear to front
           if pq[i].priority<=priority
               break
           end if
       end for
```

```
            pq[i + 1] = pq[i];
        }
        pq[loc].item = item;
        pq[loc].priority = priority;
        rear++;
    }
}

void dequeue()
{
    int data, priority;
    if (front == -1)
    {
        printf("Queue Underflow");
    }
    else if (front == rear)
    {
        data = pq[front].item;
        priority = pq[front].priority;
        rear = front = -1;
        printf("Item deleted: %d \t priority: %d", data, priority);
    }
    else
    {
        data = pq[front].item;
        priority = pq[front].priority;
        front++;
        printf("Item deleted: %d \t priority: %d", data, priority);
    }
}

void display()
{
    int i;
    for (i = front; i != rear; i++)
    {
        printf("\nitem %d\t priority %d", pq[i].item, pq[i].priority);
    }
    printf("\nitem %d\t priority %d", pq[i].item, pq[i].priority);
}

void main()
{
    int ch, item, priority;
    char cont;
    do
    {
        printf("\n1. Enqueue");
        printf("\n2. Dequeue");
        printf("\n3. Display");
        printf("\nEnter your choice:");
        scanf("%d", &ch);
        switch (ch)
        {
            case 1:
            {
                printf("Enter item to insert:");
                scanf("%d", &item);
                printf("Enter priority:");
                scanf("%d", &priority);
                enqueue(item, priority);
                break;
            }
```

```
        loc=i+1
        for i from rear to loc
            pq[i+1]=pq[i]
            pq[loc].item=item
            pq[loc].priority=priority
            rear++
        end for
Else
        for i from rear to front
            if pq[i].priority<=priority
                 break
            end if
        end for
        loc=i+1
        for i=rear to loc
            pq[i+1]=pq[i]
        end for
 pq[loc].item=item
 pq[loc].priority=priority
 rear++
```

7. Create a function Dequeue():

```
int data,priority
if front is -1
     print "Queue Underflow"
else if front==rear
   data=pq[front].item
   priority=pq[front].priority
   set rear=front=-1
   print Item deleted as data.priority
else
     data=pq[front].item
   priority=pq[front].priority
   front++
   print Item deleted as data.priority
```

8. Create function Display()

```
        initialize integer i
for i from front to rear
   print pq[i].priority
end for
 print pq[i].priority)
```

9. Create function main()

```
declare int item,ch,priority and char cont
declare char cont
 start do
{
print 1. Enqueue"
print 2. Dequeue"
print 3. Display"
print Enter your choice:"
     scan to ch
switch(ch)
case 1:
```

```
            case 2:
            {
                dequeue();
                break;
            }
            case 3:
            {
                display();
                break;
            }
            default:
            {
                printf("Invalid choice");
                break;
            }
        }
        printf("\nDo you want to continue?");
        scanf(" %c", &cont);
    } while (cont == 'y');
}
```

Output

```
1. Enqueue
2. Dequeue
3. Display
Enter your choice:1
Enter item to insert:2
Enter priority:3
Do u want to continue?y
1. Enqueue
2. Dequeue
3. Display
Enter your choice:1
Enter item to insert:2
Enter priority:1
Do u want to continue?y
1. Enqueue
2. Dequeue
3. Display
Enter your choice:3
item 2 priority 1
item 2 priority 3
Do u want to continue?y
1. Enqueue
2. Dequeue
3. Display
Enter your choice:2
Item deleted:2 priority:1
Do u want to continue?n
```

```
    print "Enter item to insert:"
    scan to item
    print "Enter priority:"
    scan to priority)
    call enqueue(item,priority)
    break
case 2:
  call dequeue()
  break
case 3:
  call display()
  break
default:
  print "Invalid choice"
  break
print Do u want to continue?"
scanf to cont
continue while cont is 'y'
```

10. Stop

Result:

Program has been successfully executed and result obtained.