```
/*
   Program: WAP program to implement priority Queue using array.
   Programmer: Walunj S.M
*/
#include<stdio.h>
#include<conio.h>
#define MAX 4
int order;
void initialize();
void Insert();
void Delete();
void Retrieve();
void Display();
int isFull();
int isEmpty();
void isOverflow();
void isUnderflow();

struct Data
{
  char Job[10];
  int Priority;
  int Incoming_Order;
};

struct PriorityQueue
{
   int rear,front;
   struct Data que[MAX];
};
struct PriorityQueue q;
void main()
{
 int choice,x;
 char ch;
do
{
  initialize();
do
{
 clrscr();
 printf("\n\n\t Menu ");
 printf("\n\n\t 1. Insert ");
 printf("\n\n\t 2. Delete ");
 printf("\n\n\t 3. Retrieve ");
 printf("\n\n\t 4. Display ");
 printf("\n\n\t 5. isFull ");
 printf("\n\n\t 6. isEmpty ");
 printf("\n\n\t 7. isOverflow ");
```

```
printf("\n\n\t 8. isUnderflow ");
printf("\n\n\t 9. Exit ");
printf("\n\n\t Enter your choice (1,2,3,4,5,6,7,8,9) : ");
scanf("%d",&choice);
switch(choice)
{
 case 1:
     Insert();
     break;
 case 2:
     Delete();
     break;
 case 3:
     Retrieve();
     break;
 case 4:
     Display();
     break;
 case 5:
      x=isFull();
      if(x==1)
      {
        printf("\n\n\t Queue is full");
      }
      else
      {
        printf("\n\n\t Queue is not full");
      }
     break;
 case 6:
     x=isEmpty();
     if(x==1)
      {
        printf("\n\n\t Queue is empty");
      }
      else
      {
        printf("\n\n\t Queue is not empty");
      }
     break;
 case 7:
     isOverflow();
     break;
 case 8:
     isUnderflow();
     break;
 case 9:
     exit(0);
}
```

```
 printf("\n\n\t Do you want to continue(y/n) : ");
 ch=getche();
 }while(ch=='y'||ch=='Y');
 printf("\n\n\t Do you want to continue in Main Menu(y/n) : ");
 ch=getche();
 }while(ch=='y'||ch=='Y');
 getch();
}

void initialize()
{
  q.front = -1;
  q.rear = -1;
  order=0;
}

void Insert()
{
   struct Data data,temp;
   int i=0,j=0;

   if(isFull()==1)
   {
     printf("\n\n\t Queue is full");
   }
   else
   {
     q.rear =  q.rear + 1;
     printf("\n\n\t Enter Job Name : ");
     scanf("%s",&data.Job);

     printf("\n\n\t Enter priority of job : ");
     scanf("%d",&data.Priority);
     data.Incoming_Order = ++order;

     q.que[q.rear] = data;
     if(q.front ==-1)
     {
      q.front = 0;
     }
     //arrange data according to highest priority and inconing order
     for(i=q.front;i<q.rear;i++)
     {
    for(j=i+1;j<=q.rear;j++)
    {
      if(q.que[i].Priority < q.que[j].Priority)
      {
        temp = q.que[i];
        q.que[i] = q.que[j];
```

```
        q.que[j] = temp;
       }
       else
       {
        if(q.que[i].Priority == q.que[j].Priority)
        {
         if(q.que[i].Incoming_Order > q.que[j].Incoming_Order)
          {
           temp = q.que[i];
           q.que[i] = q.que[j];
           q.que[j] = temp;
          }
        }
       }
     }
      }
   }
}

void Delete()
{
   struct Data data;
   if(isEmpty()==1)
   {
     printf("\n\n\t Queue is empty");
   }
   else
   {
     data = q.que[q.front];
     printf("\n\n\t %s element is deleted : ",data.Job);
     if(q.front == q.rear)
     {
      q.front = -1;
      q.rear = -1;
     }
     else
     {
      q.front = q.front + 1;
     }
   }
}

void Retrieve()
{
   struct Data data;
   if(isEmpty()==1)
   {
     printf("\n\n\t Queue is empty");
   }
```

```
    else
    {
      data = q.que[q.front];
      printf("\n\n\t %s element is retrieved from queue",data.Job);
    }
 }

 void Display()
 {
  int i;
  if(isEmpty()==1)
  {
    printf("\n\n\t Queue is empty");
  }
  else
  {
     printf("\n\n\t");
     for(i=q.front;i<=q.rear;i++)
     {
      printf("   %s %d
%d",q.que[i].Job,q.que[i].Priority,q.que[i].Incoming_Order);
     }
  }
 }

 int isFull()
 {
   if(q.rear == MAX - 1)
   {
     return 1;
   }
   else
   {
      return 0;
   }
 }

 int isEmpty()
 {
   if(q.front == -1)
   {
     return 1;
   }
   else
   {
      return 0;
   }
 }
```

```
void isUnderflow()
{
   if(isEmpty()==1)
   {
     //if u are trying to delete element from
     //queue when queue is empty then underflow
     //occurs
     printf("\n\n\t Underflow occur");
   }
   else
   {
      printf("\n\n\t Underflow not occur");
   }
}

void isOverflow()
{
   if(isFull()==1)
   {
     //if u are trying to insert element into
     //queue when queue is full then overflow
     //occurs
     printf("\n\n\t Overflow occurs");
   }
   else
   {
      printf("\n\n\t overflow not occur");
   }
}
```