1.INPUT RESTRICTED D-QUQUE

```
#include<stdio.h>
#include<process.h>
#define qsize 5
int f=0,r=-1,ch;
int item,q[10];
int isfull()
  {return(r==qsize-1)?1:0;
  }
int isempty()
  { return(f>r)?1:0;
  }
void insert_rear()
  {  if(isfull())
     {
      printf("Queue overflow\n");
      return;
     }
   r=r+1;
   q[r]=item;
  }
void delete_front()
  {  if(isempty())
     {
      printf("The queue is empty\n");
      return;
     }
   printf("Item deleted is %d\n",q[(f)++]);
   if(f>r)
     {
      f=0;
      r=-1;
     }
  }
void delete_rear()
  { if(isempty())
     {
      printf("The queue is empty\n");
      return;
```

```
    }
   printf("Item deleted is %d\n",q[(r)--]);
   if(f>r)
     { f=0;r=-1;
     }
  }
void display()
  {int i;
   if(isempty())
     {
      printf("The queue is empty\n");
      return;
     }
   for(i=f;i<=r;i++)
    printf("%d\n",q[i]);
  }
int main()
 { for(;;)
   {
     printf("INPUT RESTRICTED DQUEUE");
printf("\n1.insert_rear\n2.delete_rear\n3.delete_front\n4.
display\n5.exit\n");
    printf("Enter choice :");
    scanf("%d",&ch);
    switch(ch)
    {  case 1:printf("Enter the item :");
              scanf("%d",&item);
              insert_rear();
              break;
       case 2:delete_rear();
              break;
       case 3:delete_front();
              break;
       case 4:display();
              break;
       default:exit(0);
      }
    }
   return 0;
```

```
}
```

```
D:\sem3\ds_lab\02-10-2020\INRDQUEUE.exe
INPUT RESTRICTED DQUEUE
1.insert_rear
2.delete_rear
3.delete_front
4.display
5.exit
Enter choice :3
The queue is empty
INPUT RESTRICTED DQUEUE
1.insert_rear
2.delete_rear
3.delete_front
4.display
5.exit
Enter choice :2
The queue is empty
INPUT RESTRICTED DQUEUE
1.insert_rear
2.delete_rear
3.delete_front
4.display
5.exit
Enter choice :4
The queue is empty
INPUT RESTRICTED DQUEUE
1.insert_rear
2.delete_rear
3.delete_front
4.display
5.exit
Enter choice :1
Enter the item :7
INPUT RESTRICTED DQUEUE
1.insert_rear
2.delete_rear
3.delete_front
4.display
5.exit
Enter choice :4
The contents of queue:7
INPUT RESTRICTED DQUEUE
1.insert_rear
2.delete_rear
```

2.OUTPUT RESTRICTED D-QUQUE

```
#include<stdio.h>
#include<process.h>
#define qsize 5
int f=0,r=-1,ch;
int item,q[10];
```

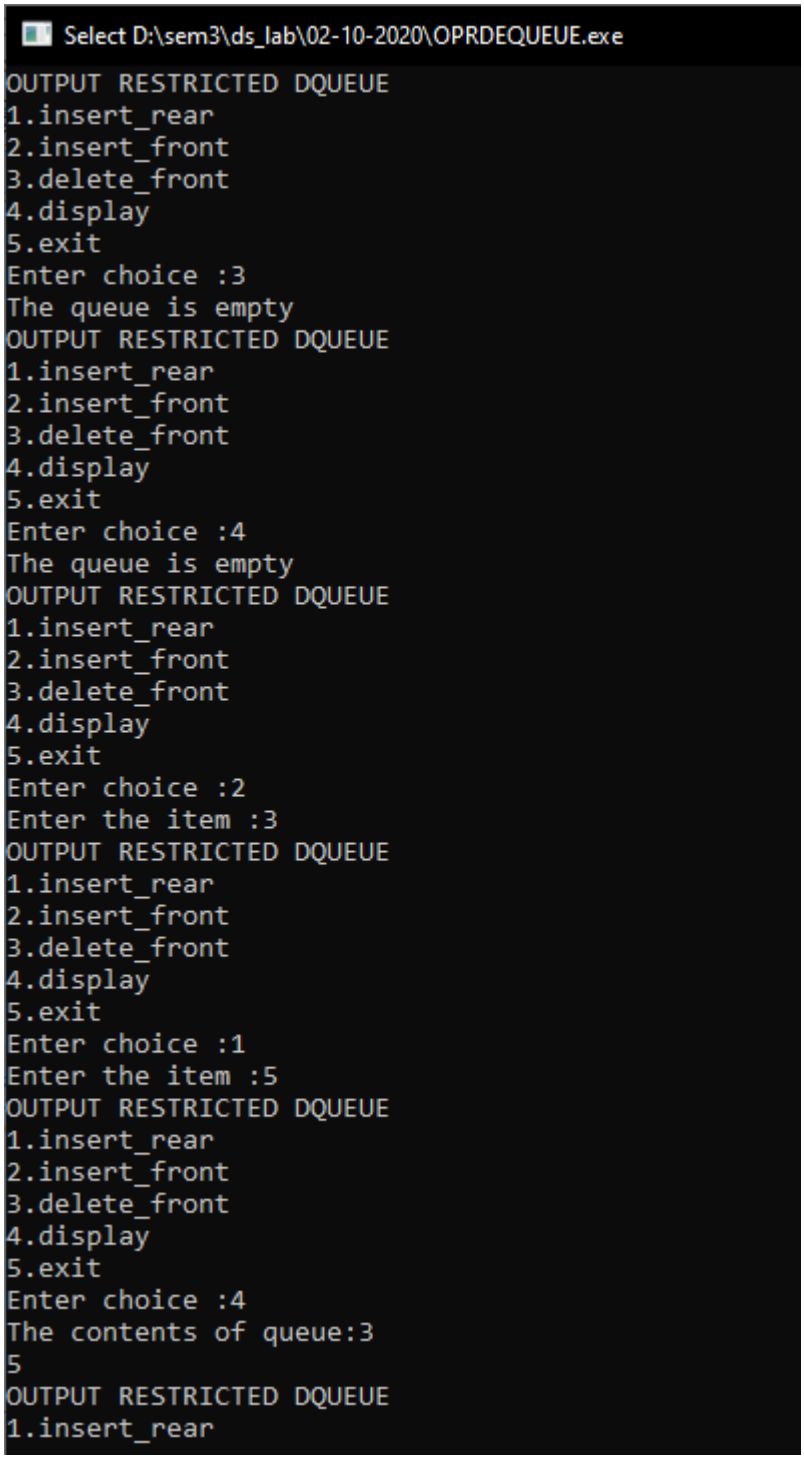
```
int isfull()
  {
   return(r==qsize-1)?1:0;
  }
int isempty()
  {
   return(f>r)?1:0;
  }
void insert_rear()
  {
   if(isfull())
     {
      printf("Queue overflow\n");
      return;
     }
   r=r+1;
   q[r]=item;
  }
void delete_front()
  {
   if(isempty())
     {
      printf("The queue is empty\n");
      return;
     }
   printf("Item deleted is %d\n",q[(f)++]);
   if(f>r)
     {
      f=0;
      r=-1;
     }
  }
void insert_front()
  {
   if(f!=0)
     {
      f=f-1;
      q[f]=item;
      return;
```

```
     }
    else if((f==0)&&(r==-1))
     {
      q[++(r)]=item;
      return;
     }
    else
      printf("Insertion from front not possible\n");
  }
void delete_rear()
  {
   if(isempty())
     {
      printf("The queue is empty\n");
      return;
     }
   printf("Item deleted is %d\n",q[(r)--]);
   if(f>r)
     {
      f=0;
      r=-1;
     }
  }
void display()
  {
   int i;
   if(isempty())
     {
      printf("The queue is empty\n");
      return;
     }
   for(i=f;i<=r;i++)
    printf("%d\n",q[i]);
  }
int main()
 {

  for(;;)
   {
```

```
     printf("OUTPUT RESTRICTED DQUEUE");

printf("\n1.insert_rear\n2.insert_front\n3.delete_front\n4
.display\n5.exit\n");
    printf("Enter choice :");
    scanf("%d",&ch);
    switch(ch)
    {
       case 1:printf("Enter the item :");
              scanf("%d",&item);
              insert_rear();
              break;
       case 2:printf("Enter the item :");
              scanf("%d",&item);
              insert_front();
              break;
       case 3:delete_front();
              break;
       case 4:display();
              break;
       default:exit(0);
      }
    }
   return 0;
  }
```



3.ASCENDING PRIORITY QUEUE

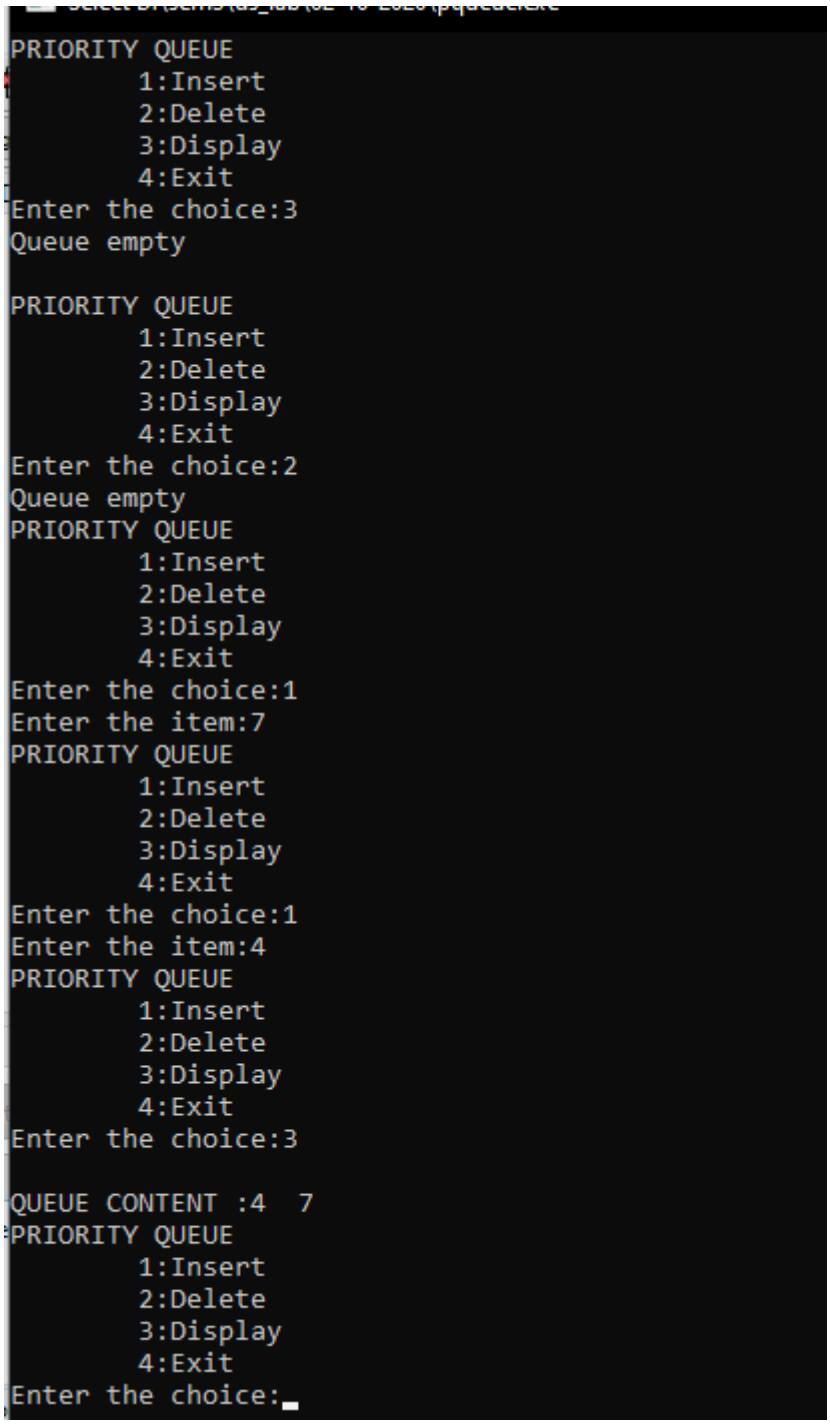
```
#include<stdio.h>
#include<process.h>
#define N 5
int queue[5], front=0, rear=-1,item;
void selsort()
{
```

```
     int i,j,small,temp;
     for(i=front;i<=rear;i++)
     {
          small=i;
          for(j=i+1;j<=rear;j++)
          {
               if(queue[j]<queue[small])
                    small=j;
          }
          if(small!=i)
          {
               temp=queue[i];
               queue[i]=queue[small];
               queue[small]=temp;
          }
     }
}
void pqinsert()
{
      if(rear==N-1)
          printf(" Queue overflow\n");
      else
      {
           printf("Enter the item:");
           scanf("%d",&item);
           rear+=1;
           queue[rear]=item;
           selsort();
      }
      return;
}
void pqdelete()
{
     int i;
       if(rear<front)
       printf("Queue empty\n");
       else
       {
       printf("Deleted item is %d \n",queue[front]);
```

```
       for(i=front;i<=rear;i++)
       {
          queue[i]=queue[i+1];
       }
       rear=rear-1;
       return;
       }
}

void display()
{
int i;
if(rear<front)
  printf("Queue empty \n");
else
  {
  printf("\nQUEUE CONTENT :");
  for(i=front;i<=rear;i++)
    printf("%d  ",queue[i]);
  }
  printf("\n");
}
int main()
{
     int ch;
     while(1)
     {
          printf("PRIORITY QUEUE\n");
          printf("\t1:Insert\n");
          printf("\t2:Delete\n");
          printf("\t3:Display\n");
          printf("\t4:Exit\n");
          printf("Enter the choice:");
          scanf("%d",&ch);
          switch(ch)
          {
           case 1:pqinsert();
                    break;
          case 2:pqdelete();
```

```
                break;
        case 3:display();
                break;
        case 4: exit(0);
        }
    }
    return 0;
}
```



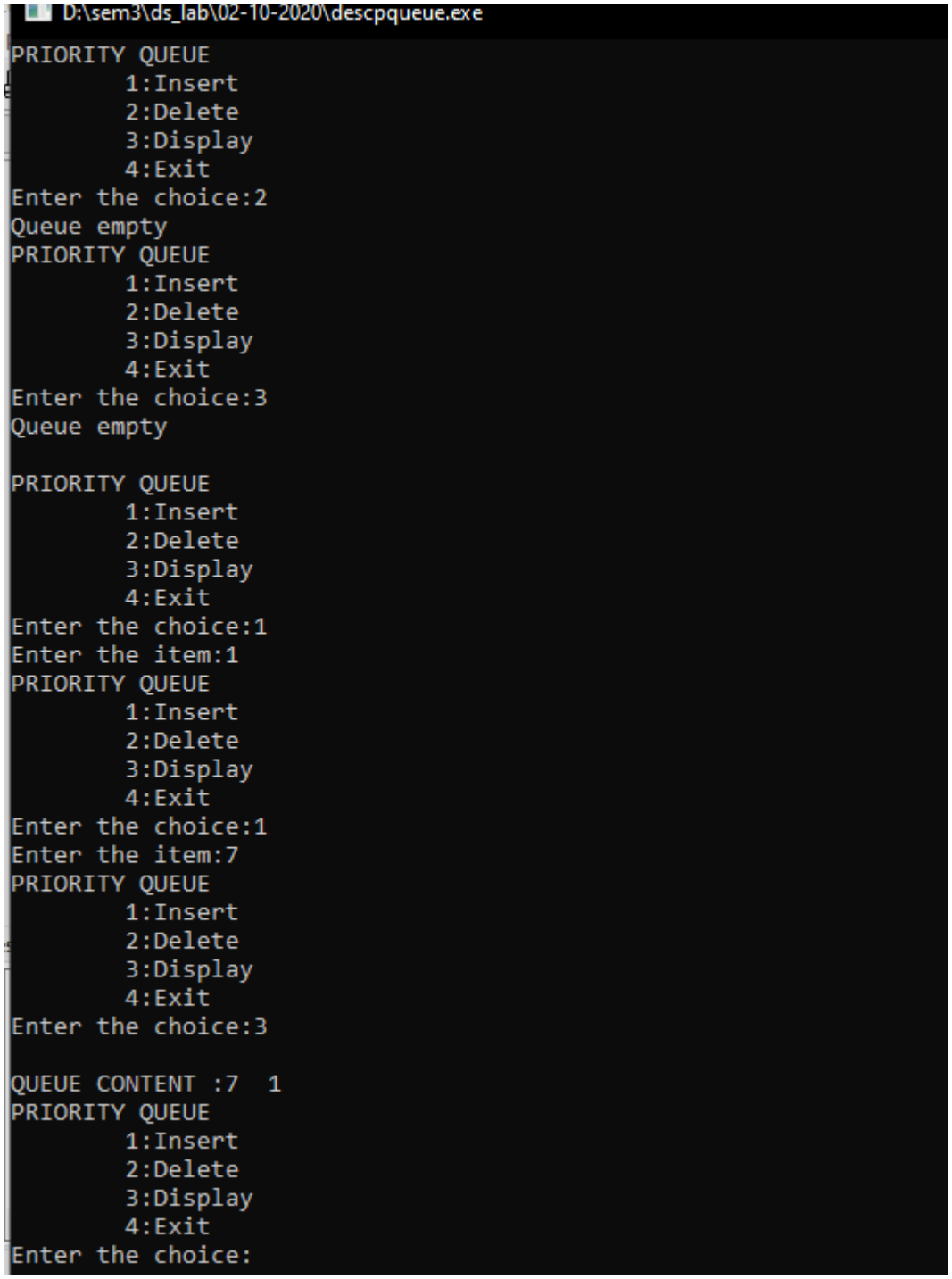
4.DESCENDING PRIORITY QUEUE

```
#include<stdio.h>
#include<process.h>
#define N 5
int queue[5], front=0, rear=-1,item;
void selsort()
{
     int i,j,big,temp;
     for(i=front;i<=rear;i++)
     {
          big=i;
          for(j=i+1;j<=rear;j++)
          {
               if(queue[j]>queue[big])
                    big=j;
          }
          if(big!=i)
          {
               temp=queue[i];
               queue[i]=queue[big];
               queue[big]=temp;
          }
     }
}
void pqinsert()
{
      if(rear==N-1)
          printf(" Queue overflow\n");
      else
      {
           printf("Enter the item:");
           scanf("%d",&item);
           rear+=1;
           queue[rear]=item;
           selsort();
      }
      return;
}
void pqdelete()
```

```
{
     int i;
       if(rear<front)
       printf("Queue empty\n");
       else
       {
       printf("Deleted item is %d \n",queue[front]);
       for(i=front;i<=rear;i++)
       {
          queue[i]=queue[i+1];
       }
       rear=rear-1;
       return;
       }
}

void display()
{
int i;
if(rear<front)
  printf("Queue empty \n");
else
  {
  printf("\nQUEUE CONTENT :");
  for(i=front;i<=rear;i++)
    printf("%d  ",queue[i]);
  }
  printf("\n");
}
int main()
{
     int ch;
     while(1)
     {
          printf("PRIORITY QUEUE\n");
          printf("\t1:Insert\n");
          printf("\t2:Delete\n");
          printf("\t3:Display\n");
          printf("\t4:Exit\n");
```

```
            printf("Enter the choice:");
            scanf("%d",&ch);
            switch(ch)
            {
             case 1:pqinsert();
                      break;
            case 2:pqdelete();
                    break;
            case 3:display();
                    break;
            case 4: exit(0);
            }
        }
        return 0;
}
```


D:\sem3\ds_lab\02-10-2020\descpqueue.exe
PRIORITY QUEUE
1:Insert
2:Delete
3:Display
4:Exit
Enter the choice:2
Queue empty
PRIORITY QUEUE
1:Insert
2:Delete
3:Display
4:Exit
Enter the choice:3
Queue empty
PRIORITY QUEUE
1:Insert
2:Delete
3:Display
4:Exit
Enter the choice:1
Enter the item:1
PRIORITY QUEUE
1:Insert
2:Delete
3:Display
4:Exit
Enter the choice:1
Enter the item:7
PRIORITY QUEUE
1:Insert
2:Delete
3:Display
4:Exit
Enter the choice:3
QUEUE CONTENT :7 1
PRIORITY QUEUE
1:Insert
2:Delete
3:Display
4:Exit
Enter the choice:

****