LAB4

1) multiple priority Queue:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#define N 3
void pqinsert(int);
void pqdelete();
void display();
int queue[3][N];
int front[3] = {0,0,0};
int rear[3] = {-1,-1,-1};
int item, pr;
int main()
{
 int ch;
 while(1)
 {
 printf("\n PRIORITY QUEUE \n");
 printf("*******\n");
 printf("\n1: PQinsert\n");
 printf("\n2: PQdelete\n");
 printf("\n3: PQ display\n");
 printf("\n4: Exit\n");
 printf("enter the choice\n");
 scanf("%d", &ch);
 switch(ch)
 {
  case 1: printf("\n enter the priority no\n");
          scanf("%d", &pr);
          if(pr>0 && pr<4)
```

```
pqinsert(pr-1);
else
printf("\n only 3 priority exists 1 2 3\n");
break;

case 2: pq delete();
        break;
case 3: display();
        break;
case 4: exit(0);
}
}
getch();
}
void pqinsert(int pr)
{
if(rear[pr]==N-1)
printf("\n queue overflow\n");
else
{
printf("\n enter the item\n");
scanf("%d",&item);
rear[pr]++;
q[pr][rear[pr]]=item;
}
}
void pq_delete()
{
int i;
for(i=0;i<3;i++)
{
if(rear[i]==front[i]-1)
printf("\n queue empty\n");
else
```

```
printf("deleted item is %d of queue %d\n",
        queue[i][front[i]],i+1);
front[i]++;
}}
}
void display()
{
int i,j;
for(i=0; i<3;i++)
{
if(rear[i]==front[i]-1)
printf("\n queue empty %d\n",i+1);
else
{
printf("\n QUEUE %d : "i+1);
for(j=front[i]; j<=rear[i];j++)
printf("%d\t",queue[i][j]);
}}
}
```

2) PRIORITY QUEUE:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
  int priority;
  int info;
  struct node *link;
} *front = NULL;
void insert(int item, int item_priority);
int del();
void display();
```

```
int isEmpty();
int main()
{
  int choice, item, item_priority;
  while(1)
  {
printf("\n1. insert\n");
printf("2. delete\n");
printf("3. display\n");
printf("4. Quit\n");
printf("\n enter your choice:");
scanf("%d", &choice);
switch(choice)
{
case 1:
printf("\n Input the item to be added on
        the queue:");
scanf("%d", &item);
printf("\n Enter its priority:");
scanf("%d", &item_priority);
insert(item, item_priority);
break;
case 2:
        printf("\n Deleted item is %d\n", del());
        break;
case 3:
        display();
        break;
case 4:
        exit(1);
default:
        printf("\n Wrong choice\n");
}}
  return 0;
}
```

```
void insert (int item, int item_priority)
{
 struct node *temp, *p;
 temp = (struct node *) malloc (sizeof (struct node));
 if (temp == NULL)
 {
 printf ("\n Memory not available \n");
 return;
 }
 temp -> info = item;
 temp -> priority = item_priority;
 if ( isEmpty () || item_priority < front -> priority)
 {
 temp -> link = front;
 front = temp;
 }
 else
 {
 p = front;
 while (p -> link != NULL && p -> link ->
 priority <= item_priority)
 p = p -> link;
 temp -> link = p -> link;
 p -> link = temp;
 }
}
int del()
{
 struct node *temp;
 int item;
 if (isEmpty())
 {
 printf ("\n Queue underflow \n");
 exit (1);
```

```
else
{
    temp = front;
    item = temp->info;
    front = front->link;
    free(temp);
}
    return item;
}
int IsEmpty()
{
  if (front == NULL)
      return 1;
  else
      return 0;
}
void display()
{
  struct node *ptr;
  ptr = front;
  if (isEmpty())
    printf("\nQueue is Empty\n");
  else {
    printf("\nQueue is :\n");
    printf("priority item :\n");
    while(ptr != NULL)
    {
      printf("%d %d\n", ptr->priority, ptr->info);
        ptr = ptr->link;
    }
  }
}
```

3) Dequeue:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.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("Queue Empty\n");
        return;
    }
```

```
printf("item deleted is %d \n", q[f++]);
if (f>r)
{
  f=0;
  r=-1;
}
}
void insertfront()
{
  if (f!=0)
{
  f=f-1;
q[f] = item;
return;
}
  else if ((f==0)&&(r==-1))
  {
    q[++r] = item;
    return;
}
  else
  printf("insertion not possible\n");
}
void deleterear()
{
  if (isEmpty())
{
  printf("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("queue empty\n");
 return;
}
  for (i = f; i<=r; i++)
printf("%d\n", q[i]);
}
 void main()
 {
  for(;;)
{
printf("1.insert rear\n2.insert front\n3.
        delete rear\n4.delete front\n5.displa
      \n6.exit\n");
printf("enter choice\n");
scanf("%d", &ch);
switch(ch)
 {
  case 1: printf("enter the item\n");
 scanf("%d", &item);
 insert_rear();
 break;
 Case 2: printf("enter the item\n");
         scanf("%d", &item);
 insert_front();
  break;
 Case 3:
        delete_rear();
        break;
 Case 4: deletefront();
        break;
```

```
case 5 :
        display();
        break;
default : exit(0);
}}
}
```