

LAB4

1) multiple priority Queue:

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <conio.h>
```

```
#define N 3
```

```
void pqueuest(int);
```

```
void pqdelte();
```

```
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("Init 1: Pqueuest \n");
```

```
printf("Init 2: PQdelte \n");
```

```
printf("Init 3: PQ display \n");
```

```
printf("Init 4: Exit \n");
```

```
printf("enter the choice \n");
```

```
scanf("%d", &ch);
```

```
switch(ch)
```

```
{
```

```
case 1 : printf("In enter the priority no \n");
```

```
scanf("%d", &pr);
```

```
if(pr > 0 & & pr < 4)
```

if (pqinsert (pr-1),
else

printf ("In 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 ("In Queue overflow\n"),

else

{

printf ("In entered the item \n");

scanf ("%d", & item);

rear [pr]++;

q[rear [pr]] [rear [pr]] = item;

}

void pq_delete()

{

int i;

for (i=0; i<3; i++)

{

if (rear [i] == front [i]-1)

printf ("In 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("In QUEUE %d : " i+1);
            for(j = front[i]; j <= rear[i]; j++)
                printf("%d ", 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, itemPriority;  
while(1)  
{
```

```
printf ("1. insert\n");
```

```
printf ("2. delete\n");
```

```
printf ("3. display\n");
```

```
printf ("4. Quit\n");
```

```
printf ("Enter your choice : ");
```

```
scanf ("%d", &choice);
```

```
switch (choice)
```

```
{
```

case 1 :

```
printf ("Input the item to be added in  
the queue : ");
```

```
scanf ("%d", &item);
```

```
printf ("\nEnter it's priority : ");
```

```
scanf ("%d", &itemPriority);
```

```
insert (item, itemPriority);
```

```
break;
```

case 2 :

```
printf ("Deleted item is %d\n", del());
```

```
break;
```

case 3 :

```
display();
```

```
break;
```

case 4 :

```
exit(1);
```

default :

```
printf ("\nWrong choice\n");
```

```
return 0;
```

```
}
```

word insert (int item, int item priority)

```

  {
    struct node *temp, *p;
    temp = (struct node *) malloc(sizeof(struct node));
    if (temp == NULL)
      {
        printf ("In 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;
          }
      }
  }
  
```

unit del()

{

```

    struct node *temp;
    int item;
    if (!isEmpty ())
      
```

printf ("In Queue underflow \n");

}, unit ()

else
{

 tmp = front;

 item = tmp -> info;

 front = front -> link;

 free (tmp);

}
 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 <iostream.h>
#include <conio.h>
#include <stdlib.h>
#define qsize 5
int f=0, r=-1, ch;
char 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");
    else
    {
        r=r+1;
        q[r]=item;
    }
}
void delete_front()
{
    if (isEmpty())
        printf ("Queue empty\n");
    else
        return;
}
```

```
printf ("item deleted is %d \n", q[f++]);  
if (f > r)  
{
```

$f = 0;$
 $r = -1;$

```
y  
void insertfront()  
{
```

```
if (f != 0)
```

$f = f - 1;$
 $q[f] = item;$
return;

```
else if ((f == 0) && (r == -1))
```

```
{  
    q[+r] = item;  
    return;
```

y
else

```
printf ("insertion not possible \n");
```

y
void deleterear()

```
{ if (isEmpty())
```

```
    printf ("queue is empty \n");
```

return;

y
printf ("item deleted is %d \n", q[r--]);

```
{ if (f > r)
```

$f = 0;$
 $r = -1;$

```
void display()
{
    int i;
    if ( sempty() )
        printf ("Queue empty\n");
    return;
}
for ( i = 0; i < 9; i++ )
    printf ("%d\n", q[i]);
}

void main()
{
    for ( ; )
        printf ("1. insert rear\n 2. insert front\n 3.
                delete rear\n 4. delete front\n 5. di
                \n 6. exit\n");
        printf ("enter choice\n");
        scanf ("%d", & ch);
        switch (ch)
        {
            case 1: printf ("enter the item\n");
                    scmpf ("%d", & item);
                    insert_rear();
                    break;
            Case 2: printf ("enter the item\n");
                    scmpf ("%d", & item);
                    insert_front();
                    break;
            Case 4: delete_rear();
                    break;
        }
}
```

case 5 :

display();
break;

default : exit(0);

yy
y