```
//LQif.h
typedef struct{
        int arr[10];
        int f,r;
        int cap
}Queue;

void initialize(Queue *q);
int isFull(Queue *q);
int isEmpty(Queue *q);
void enqueue(Queue *q, int x);
void display(Queue *q);

//LQimpl.h
#include<stdio.h>
#include<stdlib.h>

void initialize(Queue *q){
        q->f=q->r=-1;
        q->cap=10;
}

int isEmpty(Queue *q){
        if(q->f==-1)
                return 1;
        return 0;
}

int isFull(Queue *q){
        if(q->r==(q->cap-1))
                return 1;
        return 0;
}

void enqueue(Queue *q, int x){
        if(isFull(q))
                printf("Queue is Full");
        if(q->f==-1){
                q->f++;
                q->r++;
                q->arr[q->r]=x;
        }
        else{
                q->r++;
                q->arr[q->r]=x;
        }
}

int dequeue(Queue *q){
        if(isEmpty(q)){
                printf("Queue is Empty.");
                q->f=q->r=-1;
        }

        else{
                int data=q->arr[q->f];
                q->f++;
                return data;
        }
}

void display(Queue *q){
        int i;
        if(isEmpty(q))
                printf("Queue is Empty");
        else{
                for(i=q->f;i<=q->r;i++)
                        printf("%d        ",q->arr[i]);
                }
}

//LQapp.c
#include<stdio.h>
#include<stdlib.h>
#include "LQif.h"
#include "LQimpl.h"

void main(){
        int d;
        Queue q;
        initialize(&q);
        int i,t,n;
        printf("Enter no. of queue elements: ");
        scanf("%d",&n);
        for (i=0;i<n;i++){
                printf("\nEnter element: ");
                scanf("%d",&t);
                enqueue(&q,t);
        }

        printf("\t\t\tQueue:\n");
        display(&q);

        char ch;
        printf("\nDo you want to dequeue? (y/n)\n");
        scanf(" %c",&ch);

        while(ch!='n'){
                printf("\nDequeued %d",dequeue(&q));
                printf("\nDo you want to dequeue? (y/n)\n");
                scanf(" %c",&ch);
        }

        printf("\t\t\tQueue:\n");
        display(&q);

}

/*
OUTPUT:
exam31@jtl-19:~$ gcc LQapp.c -o s
In file included from LQapp.c:3:0:
LQif.h:5:1: warning: no semicolon at end of struct or union
 }Queue;
 ^
exam31@jtl-19:~$ ./s
```

```
Enter no. of queue elements: 4

Enter element: 1

Enter element: 2

Enter element: 3

Enter element: 4
                              Queue:
1         2         3         4
Do you want to dequeue? (y/n)
y

Dequeued 1
Do you want to dequeue? (y/n)
y

Dequeued 2
Do you want to dequeue? (y/n)
y

Dequeued 3
Do you want to dequeue? (y/n)
y

Dequeued 4
Do you want to dequeue? (y/n)
n
```

```
*/
//CQif.h
typedef struct{
	int arr[10];
	int f,r;
	int cap,size;
}Queue;

void initialize(Queue *q);
int isFull(Queue *q);
int isEmpty(Queue *q);
void enqueue(Queue *q, int x);
int dequeue(Queue *q);
void display(Queue *q);

//CQV1impl.h
#include<stdio.h>
#define max 10

void initialize(Queue *q){
	q->cap=max;
	q->size=0;
	q->f=q->r=-1;
}

int isFull(Queue *q){
	if(q->size==q->cap-1)
		return 1;
	return 0;
```

```
}

int isEmpty(Queue *q){
        if((q->size==0))
                return 1;
        return 0;
}

void enqueue(Queue *q, int x){
            if(isFull(q))
                        printf("Queue is Full");
        else if(q->f==-1 && q->r==-1){
                q->f=q->r=0;
                q->arr[q->r]=x;
                q->size++;
        }
            else{
                        q->r=(q->r+1)%(q->cap);
                        q->arr[q->r]=x;
                        q->size++;
            }
            printf("Enqueued %d\n",q->arr[q->r]);
}

int dequeue(Queue *q){
        int data;
            if(isEmpty(q))
                        printf("Queue is Empty");
        else if(q->f==q->r){
            data=q->arr[q->f];
                q->f=q->r=-1;
                q->size=0;
                return data;
    }
        else{
                data=q->arr[q->f];
                q->f=(q->f+1)%q->cap;
                q->size--;
                return data;
        }
}

void display(Queue *q){
        int i;
    printf("\nElements in Circular Queue are: ");
    if(isEmpty(q))
        printf("Queue is Empty");
    else if (q->r>=q->f){
        for (i=q->f;i<=q->r; i++)
            printf("%d    ",q->arr[i]);
    }
    else
    {
        for (i=q->f;i<q->size;i++)
            printf("%d    ",q->arr[i]);

        for (i=0;i<=q->r;i++)
            printf("%d    ",q->arr[i]);
        }
}

//CQV1app.c
#include<stdio.h>
#include<stdlib.h>
#include "CQif.h"
#include "CQV1impl.h"

void main(){
            int d;
            Queue q;
            initialize(&q);
        printf("\nRear: %d,Front: %d",q.r,q.f);
        printf("\nSize:%d,Capacity:%d",q.size,q.cap);
            int i,t,n;
            printf("\nEnter no. of queue elements: ");
            scanf("%d",&n);

            for (i=0;i<n;i++){
                        printf("\n\nEnter element: ");
                        scanf("%d",&t);
                        enqueue(&q,t);
               printf("Rear: %d,Front: %d",q.r,q.f);
                        printf("        Size %d",q.size);
                        display(&q);
            }
               printf("\n\nDequeued %d",dequeue(&q));
               printf("\nFront=%d      Rear=%d     Size: %d",q.f,q.r,q.size);
               display(&q);

               printf("\nDequeued %d",dequeue(&q));
               printf("\nFront=%d      Rear=%d     Size: %d",q.f,q.r,q.size);
               display(&q);

               printf("\nDequeued %d",dequeue(&q));
               printf("\nFront=%d      Rear=%d     Size: %d",q.f,q.r,q.size);
               display(&q);

               printf("\nDequeued %d",dequeue(&q));
               printf("\nFront=%d      Rear=%d     Size: %d",q.f,q.r,q.size);
               display(&q);
}

/*
OUTPUT:
Rear: -1,Front: -1
Size:0,Capacity:10
Enter no. of queue elements: 4



Enter element: 1
Enqueued 1
Rear: 0,Front: 0     Size 1
Elements in Circular Queue are: 1
```

```
Enter element: 2

Enqueued 2

Rear: 1,Front: 0        Size 2

Elements in Circular Queue are: 1     2


Enter element: 3

Enqueued 3

Rear: 2,Front: 0        Size 3

Elements in Circular Queue are: 1     2     3


Enter element: 4

Enqueued 4

Rear: 3,Front: 0        Size 4

Elements in Circular Queue are: 1     2     3     4


Dequeued 1

Front=1          Rear=3        Size: 3

Elements in Circular Queue are: 2     3     4

Dequeued 2

Front=2          Rear=3        Size: 2

Elements in Circular Queue are: 3     4

Dequeued 3

Front=3          Rear=3        Size: 1

Elements in Circular Queue are: 4

Dequeued 4

Front=-1          Rear=-1        Size: 0

*/


//CQV2impl.h
```

```
#include<stdio.h>

#include<stdlib.h>

#define max 100

typedef struct

{

	int jobno,bursttime;

}job;

typedef struct

{

	job arr[max];

	int size,cap,f,r;

}queue;


queue* initialize(queue *q);

int isfull(queue *q);

int isempty(queue *q);

void enqueue(queue *q,job x);

job dequeue(queue *q);

int queuetime(queue *q);

void disp(job j);

void display(queue *q);

int queuetime(queue *q);


queue* initialize(queue *q){

	q=(queue*)malloc(sizeof(q));

	q->f=q->r=-1;

	q->size=0;

	q->cap=max;

}
```

```
int isFull(queue *q){
        if(q->size==q->cap-1)
                return 1;
        return 0;
}

int isEmpty(queue *q){
     if((q->size==0))
          return 1;
     return 0;
}

void enqueue(queue *q,job x){
     printf("\nBefore: f:%d r:%d",q->f,q->r);
     if(isFull(q))
                printf("Queue is Full");
     else if(q->f==-1 && q->r==-1){
          q->f=q->r=0;
          q->arr[q->r]=x;
          q->size++;
     }
        else{
                q->r=(q->r+1)%(q->cap);
                q->arr[q->r]=x;
                q->size++;
        }
        printf("\nAfter: f:%d r:%d",q->f,q->r);
}
```

```
job dequeue(queue *q){
        job data;
            if(isEmpty(q))
                        printf("Queue is Empty");
        else if(q->f==q->r){
            data=q->arr[q->f];
                        q->f=q->r=-1;
                        q->size=0;
                        return data;
        }
            else{
                        data=q->arr[q->f];
                        q->f=(q->f+1)%q->cap;
                        q->size--;
                        return data;
            }
}

void disp(job j){
            printf("(J%d,%d) ",j.jobno,j.bursttime);
}

void display(queue *q){
            int i;
        printf("\nElements in Circular Queue are: ");
        if(isEmpty(q))
            printf("Queue is Empty");
        else if (q->r>=q->f){
            for (i=q->f;i<=q->r; i++)
                    disp(q->arr[i]);
        }
        else{
            for (i=q->f;i<q->size;i++)
                    disp(q->arr[i]);

            for (i=0;i<=q->r;i++)
                    disp(q->arr[i]);
        }
}

int queuetime(queue *q)
{
            int i,sum=0;
            if(q->f==0 && q->r==0)
               return q->arr[q->r].bursttime;

            else if(q->f<q->r){
                        for(i=q->f;i<=q->r;i++)

                                    sum+=q->arr[i].bursttime;
            }

            else if(q->f>q->r){
                        for(i=q->f;i<q->size;i++)
                                    sum+=q->arr[i].bursttime;
                        for(i=0;i<q->r;i++)
                                    sum+=q->arr[i].bursttime;
```

```
	}
	return sum;
}

//CQV2app.c
#include "CQV2impl.h"
void main()
{
	queue *q1,*q2;
	int sm,k=0;
	job s;
	q1=initialize(q1);
	q2=initialize(q2);
	int ch;

	do
	{
		printf("\n\tMENU:\n[1]Insert \n[2]Display \n[0]EXIT\n");
		printf("Your choice : ");
		scanf("%d",&ch);
		switch(ch)
		{
			case 1 :printf("Enter Burst time: ");
			  scanf("%d",&sm);
			  s.jobno=k;
			  s.bursttime=sm;
			  k++;
			  printf("Inserting:");
			  disp(s);
```

```
                    printf("\nBefore: Q1:%d      Q2:%d",queuetime(q1),queuetime(q2));

                    if(queuetime(q1)<queuetime(q2))
                        enqueue(q1,s);
                    else
                        enqueue(q2,s);
                    display(q1); display(q2);
                    printf("\nAfter:Q1:%d      Q2:%d",queuetime(q1),queuetime(q2));
                    break;
                  case 2 :display(q1);
                    printf("\n");
                    display(q2);
                    break;
                  case 0 :break;
                  default:printf("Invalid \n");
            }
      }while(ch!=0);
}
/*
OUTPUT:

        MENU:
[1]Insert
[2]Display
[0]EXIT
Your choice : 1
Enter Burst time: 4
Inserting:(J0,4)
Before: Q1:0      Q2:0
```

```
Before: f:-1 r:-1

After: f:0 r:0

Elements in Circular Queue are: Queue is Empty

Elements in Circular Queue are: (J0,4)

After:Q1:0      Q2:4

                MENU:

[1]Insert

[2]Display

[0]EXIT

Your choice : 1

Enter Burst time: 2

Inserting:(J1,2)

Before: Q1:0      Q2:4

Before: f:-1 r:-1

After: f:0 r:0

Elements in Circular Queue are: (J1,2)

Elements in Circular Queue are: (J0,4)

After:Q1:2      Q2:4

                MENU:

[1]Insert

[2]Display

[0]EXIT

Your choice : 3

Invalid


                MENU:

[1]Insert

[2]Display

[0]EXIT
```

```
Your choice : 1

Enter Burst time: 5

Inserting:(J2,5)

Before: Q1:2      Q2:4

Before: f:0 r:0

After: f:0 r:1

Elements in Circular Queue are: (J1,2) (J2,5)

Elements in Circular Queue are: (J0,4)

After:Q1:7      Q2:4

                MENU:

[1]Insert

[2]Display

[0]EXIT

Your choice : 1

Enter Burst time: 9

Inserting:(J3,9)

Before: Q1:7      Q2:4

Before: f:0 r:0

After: f:0 r:1

Elements in Circular Queue are: (J1,2) (J2,5)

Elements in Circular Queue are: (J0,4) (J3,9)

After:Q1:7      Q2:13

                MENU:

[1]Insert

[2]Display

[0]EXIT

Your choice : 1

Enter Burst time: 3

Inserting:(J4,3)
```

```
Before: Q1:7      Q2:13

Before: f:0 r:1

After: f:0 r:2

Elements in Circular Queue are: (J1,2) (J2,5) (J4,3)

Elements in Circular Queue are: (J4,3) (J3,9)

After:Q1:10      Q2:12

            MENU:

[1]Insert

[2]Display

[0]EXIT

Your choice : 0

*/
```