//CQ USING ARRAYS

```
#include<stdio.h>
#define max 5
typedef struct CQueue
{
	int q[max];
	int front;
	int rear;
}CQUEUE;

void init(CQUEUE *pcq)
{
	pcq->rear = max-1;
	pcq->front = max-1;
}
int isfull(CQUEUE *pcq)
{
	return ((pcq->rear+1)%max == pcq->front);
}
int isempty(CQUEUE* pcq)
{
	return (pcq->rear == pcq->front);
}
void enqueue(CQUEUE *pcq, int ele)
{
	if (isfull(pcq))
		printf("full\n");
	else
	{
		pcq->rear = (pcq->rear+1)%max;
		pcq->q[pcq->rear] = ele;
	}
}
int dequeue(CQUEUE* pcq)
{
	int ele;
	if (isempty(pcq))
		printf("empty\n");
	else
	{
		pcq->front = (pcq->front+1)%max;
		ele = pcq->q[pcq->front];
	}
}
int display(CQUEUE* pcq)
{
	int k;
	if(isempty(pcq))
		printf("empty");
	else
	{
		k=(pcq->front+1)%max;		//inc. suitable f as it is one place behind
		while(k != pcq->rear)		//why not <=??
		{
```

```
                printf("%d\t", pcq->q[k]);
                k=(k+1)%max;        //imp
            }
            printf("%d\t", pcq->q[k]);   //last element
        }
}
int FoQ(CQUEUE* pcq)
{
        return pcq->q[(pcq->front+1)%max];
}

int main()
{
        CQUEUE pcq;
        init(&pcq);
        int ch,ele;
        while(1)
        {       printf("enter the operation to be performed:\n");
                printf("1-enqueue\n2-dequeue\n3-display\n4-isempty\n5-ToQ\n");
                scanf("%d", &ch);
                switch(ch)
                {
                        case 1: printf("enter the element\n");
                                scanf("%d",&ele);
                                enqueue(&pcq,ele);
                                break;

                        case 2: ele = dequeue(&pcq);
                                printf("deleted element is %d",ele);
                                break;

                        case 3: printf("the elements are:\n");
                                display(&pcq);
                                break;

                        case 4: if(isempty(&pcq))
                                        printf("Empty\n");
                                else
                                        printf("!empty\n");
                                break;

                        case 5: printf("%d\n",FoQ(&pcq));
                                break;
                }
        }
return 0;
}
/////////-----------------------------------------------------------------------------------------
```

//CQ USING LL WITHOUT LIST STRUCTURE

```
#include<stdio.h>
#include<stdlib.h>  //imp
typedef struct node
{
        int data;
        struct node* link;

}NODE;
NODE* front = NULL;        //making it global
NODE* rear = NULL;         //to avoid passing as function arguments


void display( );
void insert_last();
void delete_beg();

int main()
{
NODE *front, *rear;
front = NULL;              //initially as NULL
rear = NULL;
int n,c,ch;
do
{
        printf("enter the choice :\n 1-display\n 2-insert at last\n 3-delete from beginning \n");
        scanf("%d", &ch);
        switch(ch)
        {
                case 1: printf("entered choice is 1-display\n");
                        display();
                        break;

                case 2: printf("entered choice is 3-insert at last\n");
                        insert_last();
                        break;


                case 3: printf("entered choice is 5-delete at beginning\n");
                        delete_beg();
                        break;
        }
}while(1);
return 0;
}
void display()
{
        NODE* temp;
        temp = front; //first node
        if((front==NULL)&&(rear==NULL))          //check for only front== NULL only
                printf("Queue is Empty\n");
        else
        {
                do{
                    printf("%d\t",temp->data);
                    temp = temp->link;
            }while(temp != front);          //inc. temp will again point it back to front as it is CQ
        }
        printf("\n");
}
//-----------------------------------------------------------insert_beg-------------------------
```

void insert_last()

```
{
        printf("\nenter the data:\n");
        int value;
        scanf("%d", &value);
                //--------DATA ENQUEUE IN TEMP----------------
        NODE* temp = (NODE*)malloc(sizeof(NODE));
        temp->data = value;
        temp->link = NULL;

        if (front == NULL && rear == NULL)        //NO NODE
        {
                front = temp;           //both f and r points to temp
                rear = temp;
                rear->link = front;     //rear points back to front as it is CQ
        }

        else                            //some node are already there linked
        {
                rear->link = temp;      //change rear->link-- make a connection to existing ll
                rear = temp;            //update rear
                temp->link = front;     //As it is CQ //check for rear->link = front ??
        }

}
```

void delete_beg()

```
{
        NODE* temp = front;             //points to beginning

        if(front==NULL && rear == NULL)
                printf("list is empty");

        else if(front == rear) //one node only in ll
        {
                front = NULL;
                rear =NULL;
        }
        else                            //multiple nodes
        {
                front = front->link;    //update front
                rear->link = front;     // update rear
        }
printf("\ndeleted node is --- %d\n", temp->data);
                free(temp);
}
```

//ASCENDING PQ

```
#include<stdio.h>
#define max 15

typedef struct Input_info
{
	int data;
	int pr;

}INFO;
typedef struct Priority_queue
{
	INFO q[max];
	int front;
	int rear;
}PQUEUE;

int count =0;
void init(PQUEUE* ppq)
{
	ppq->front = 0;
	ppq->rear = -1;
}

int isempty(PQUEUE* ppq)
{
	return (ppq->front > ppq->rear);
}

int isfull(PQUEUE* ppq)
{
	return (ppq->rear == max-1);
}
void enqueue(PQUEUE* ppq, int ele, int Epr, int co)//500
{
	int i,pos,k;
	if(isfull(ppq))
		printf("full queue\n");
	else
	{
		i = ppq->rear;
			ppq->rear++;
			while(ppq->q[i].pr >= Epr && i>=0)
			{
				ppq->q[i+1] = ppq->q[i];
				i--;
			}
			ppq->q[i+1].data = ele;
			ppq->q[i+1].pr = Epr;
	}
}
```

```
int dequeue(PQUEUE* ppq)
{       int ele;
        if(isempty(ppq))
        {
                printf("empty queue\n");
                ele = -1;
        }
        else
        {
                ele = ppq->q[ppq->front].data;
                ppq->front++;
        }
        return ele;
}
void display(PQUEUE* ppq)
{       int k = ppq->front;
        while(k<=ppq->rear)
        {
                printf("%d\t", ppq->q[k].data);
                k++;
        }
}
int FoQ(PQUEUE* ppq)
{
        return (ppq->q[ppq->front].data);
}
int main()
{
        PQUEUE ppq;
        init(&ppq);
        int ch,ele,Epr;
        while(1)
        {       printf("enter the operation to be performed:\n");
                printf("1-enqueue\n2-dequeue\n3-display\n4-isempty\n5-FoQ\n");
                scanf("%d", &ch);
                switch(ch)
                {
                        case 1: printf("enter the element and its priority\n");
                                scanf("%d%d", &ele,&Epr);
                                enqueue(&ppq,ele,Epr,count);count++;
                                break;
                        case 2: ele = dequeue(&ppq);
                                printf("deleted element is %d\n", ele);count--;
                                break;
                        case 3: printf("Queue contents are:\n");
                                display(&ppq);
                                break;
                        case 4: if(isempty(&ppq))
                                        printf("empty queue\n");
                                else
                                        printf("Not empty queue\n");
                                break;
                        case 5: printf("front value= %d\n", FoQ(&ppq));
                                break;
        }}}
```