Program

```
#include <stdio.h>
# define max 3
void enqueue (int q[], int *f, int *r)
{
    if(*r - *f == max-1 || *r == *f-1)
    printf("Queue is full \n\n");
    else
      {
      if(*r == -1)
          *f = *r = 0;
      else
          *r = (*r+1) % max;
      printf("Enter the element :\n");
      scanf("%d", &q[*r]);
      }
}
```

```
void dequeue (int q[], int *f, int *r)
{
    if (*f == -1)
        printf ("Queue is empty \n\n");
    else
    {
        printf ("%d is deleted \n", q[*f]);
        if (*f == *r)
            *f = *r = -1;
        else
            *f = (*f + 1) % max;
    }
}
void display (int q[], int *f, int *r)
{
    if (*f == -1)
        printf ("Queue is empty \n\n");
    else
    {
        for (int i = *f ; ; i++)
        {
            i = i % max;
            printf ("%d", q[i]);
            if (*r == i)
                break;
        }
        printf ("\n");
    }
}
int main ()
{
    int choice, f = -1, r = -1, q[max];
    do
```

```
{
    printf("1: Insert \n 2: Delete \n 3: display \n: Exit \n");
    printf("Enter your choice \n");
    scanf("%d", &choice);
    switch(choice)
    {
        case 1: enqueue(q, &f, &r);
            break;
        case 2: dequeue(q, &f, &r);
            break;
        Case 3: display(q, f, r);
            break;
        case 4:
            break;
        default: printf("Invalid choice\n");
    }
} while(choice != 4);
}
```

Circular Queue

Pseudo

```
A(SIZE)
FRONT = -1
REAR = -1

IsFull()
{
  if ( front == (rear+1) % N)
      return true
    else
      return False
}

IsEmpty()
{
  if( front == -1 && rear == -1)
      return True
    else
      return false
}

Enqueue(x)
{
    if ( IsFull())
        printf("Q is Full")
      else if ( IsEmpty())
        front <- rear <- 0
      else
        rear <- (rear+1) % N
```

```
Dequeue()
{
   if (IsEmpty())
     printf("Q is empty")
   else if (front == rear)
     x <- A[front]
       front <- rear <-1
   else
     {x <- A[front]
         front <- (front+1)%N
     }
   return x
}
```