```
#include <stdio.h>

#define n 4
int que[n] ,front=-1 ,rear=-1 ;

void display(int que[] ,int front ,int rear)
{
        if( front==-1 && rear==-1 )
        {
                printf("Queue is empty\n");
        }
        else
        {
            int i=front;
                while(i!=rear){
                        printf("%d\t",que[i]);
                        //if(i==(front+n-1)%n)
                          //  break;
                        i=(i+1)%n;
                }
                printf("%d\t",que[rear]);
                printf("\n");
        }
        //printf("**%d,%d**",front,rear);
}

void frontq(int que[] ,int* front ,int* rear ){
        if( (*rear+1)%n == *front)
        {
                printf("Queue is full\n");
                //*rear+=1;
                return;
        }
        if( *front ==-1 )
        {
                *front=(*front+1)%n;
        }
        int item;
        printf("What should I insert? :  ");
    scanf("%d",&item);

        *rear=(*rear+1)%n;
        que[*rear]=item;

        //printf("**%d,%d**",*front,*rear);
}

void dequeue(int que[] ,int* front ,int* rear ){
        int item;
        if( *front==-1 && *rear==-1 ){
```

```
            printf("Empty Queue\n");
        }
        else{
        item=que[*front];
                if( *front==*rear ){
                        *rear=-1;
                        *front=-1;
                }
                else{
                //que[*front]='\0';
                        *front=(*front+1)%n;
                }
        printf("%d is removed\n",item);
        }
        //printf("**%d,%d**",*front,*rear);
}
void inject(int que[] ,int* front ,int* rear ){
        if( (*front+n-1)%n == *rear)
        {
                printf("Queue is full\n");
                //*front=(*front+n-1)%n;
                return;
        }
        if( *rear ==-1 )
        {
                *front=1;//(*front+1)%n;
                *rear=0;//(*rear+1)%n;
        }
        int item;
        printf("What should I insert? :  ");
    scanf("%d",&item);

        *front=(*front+n-1)%n;
        que[*front]=item;

        //printf("**%d,%d**",*front,*rear);
}

void eject(int que[] ,int* front ,int* rear ){
        int item;
        if( *front==-1 && *rear==-1 ){
            printf("Empty Queue\n");
        }
        else{
        item=que[*rear];
                if( *front==*rear ){
                        *rear=-1;
                        *front=-1;
                }
                else{
```

```
                //que[*front]='\0';
                        *rear=(*rear+n-1)%n;
                }
        printf("%d is removed\n",item);
        }
        //printf("**%d,%d**",*front,*rear);
}


int main(){
        //printf("Enter the total size of the Queue : ");
    //int n=3;
    //scanf("%d", & n);
    int choice;
    int item=0;
    printf("1...display\n");
    printf("2...enque\n");
    printf("3...deque\n");
    printf("4...inject\n");
    printf("5...eject\n");
    printf("6...quit\n");

    int quit=1;
    while(quit!=0){
        printf("\nOption : ");
        scanf("%d",&choice);

        switch(choice){
          case 1: display(que,front,rear);
            break;
          case 2:       frontq(que,&front,&rear);
            break;
          case 3: dequeue(que,&front,&rear);
            break;
          case 4:       inject(que,&front,&rear);
            break;
          case 5:       eject(que,&front,&rear);
            break;
          case 6: quit=0;
              printf("*****Program aborted*****");
              break;
          default:printf("\n1...display\n");
            printf("2...frontq\n");
            printf("3...deque\n");
            printf("4...inject(enque)\n");
            printf("5...eject\n");
            printf("6...quit\n");
        }
    }
}
```

```
1...display
2...frontq
3...deque
4...inject
5...eject
6...quit

Option : 2
What should I insert? :  1

Option : 2
What should I insert? :  2

Option : 2
What should I insert? :  3

Option : 4
What should I insert? :  4

Option : 1
4   1   2   3

Option : 4
Queue is full

Option : 2
Queue is full

Option : 3
4 is removed

Option : 1
1   2   3

Option : 5
```

```
Option : 5
3 is removed

Option : 5
2 is removed

Option : 1
1

Option : 2
What should I insert? :  5

Option : 1
1   5

Option : 6
*****Program aborted*****;
```

# 9- Deque

## Aim

Do a deque datastructure.

## Algorithm

PUSH DQ ITEM
~~ENQUEUE~~ (~~ITEM~~)

1. START
2. If (FRONT = = REAR = = 1) Then
    1. FRONT = 0
    2. REAR = 0
    3. DQ[FRONT] = ITEM
3. Else
    1. If (FRONT = = 0) Then
        1. temp = N-1
    2. Else
        1. temp = FRONT - 1
    3. End If
    4. If (temp = = REAR) Then
        1. Print "DQ Full"
    5. Else
        1. FRONT = temp
        2. DQ[FRONT] = ITEM
    6. End If
4. End If
5. STOP

1. START
2. If (FRONT == REAR == -1)
    1. print "DQ Empty"
3. Else
    1. ITEM = DQ [FRONT]
    2. If (FRONT == REAR)
        1. FRONT = -1
        2. REAR = -1
    3. Else
        1. FRONT = (FRONT + 1) % size
    4. End If.
4. End If
5. STOP.

Enqueue

## INJECT (ITEM)

1. START
2. If (FRONT = = (REAR + 1) mod size) Then
    1. Print "DQ Full"
3. Else
    1. If (FRONT == REAR == -1) Then
        1. FRONT = 0
        2. REAR = 0
        3. DQ[REAR] = ITEM
    2. Else
        1. REAR = (REAR + 1) % size
        2. DQ[REAR = ITEM
    3. End If
4. End If
5. STOP

# EJECT ()

1. START
2. If (FRONT == REAR == -1) Then
    1. print "DQ Empty"
3. Else
    1. If (FRONT == REAR) Then
        1. ITEM = DQ [REAR]
        2. FRONT = -1
        3. REAR = -1
    2. Else
        1. ITEM = DQ[REAR]
        2. If (REAR == 0) Then
            1. REAR = N-1
        3. Else
            1. REAR = REAR-1
        4. End If
    3. End If
4. End If
5. STOP