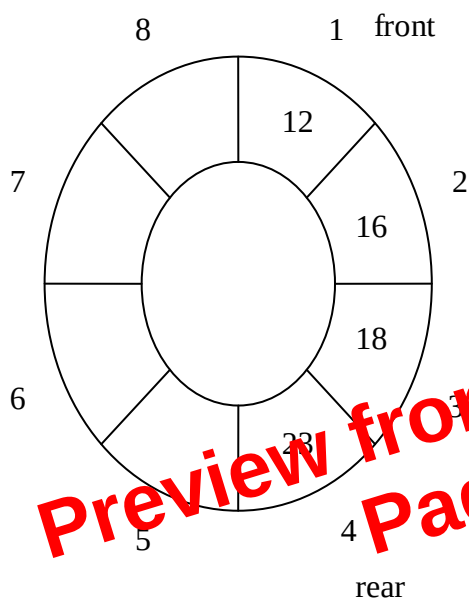


```
{
    int i;
    printf("\nQueue elements are \n");
    for(i=Q->front; i<=Q->rear; i++)
    {
        printf("%d\t",Q->data[i]);
    }
}
```

## 8.2 Circular Queue:



**Note:** If we start circular array from 1 to N, then we increment REAR and FRONT by formula  $(\text{REAR} \% N + 1)$ ,  $(\text{FRONT} \% N + 1)$  and if we start circular array from 0 to N-1, then we increment REAR and FRONT by formula  $(\text{REAR} + 1) \% N$ ,  $(\text{FRONT} + 1) \% N$ .

### 8.2.1 Algorithm for inserting element into Circular-Queue.

CQ-INSERT(Q, N, FRONT, REAR, ITEM)

Where Q is a circular queue of N indexes. FRONT and REAR are two integer variable, FRONT stores the position of first element and REAR stores the position of last element of queue. ITEM is an element to be inserted.

1. if  $((\text{REAR} \% N + 1) == \text{FRONT})$  then //check overflow condition
  - a. Print(Queue is Full)
  - b. Return
2. If  $(\text{FRONT} == 0)$  then
  - a.  $\text{FRONT} = \text{FRONT} + 1$
3.  $\text{REAR} = (\text{REAR} \% N + 1)$