

Data Structures and Algorithms

Prepared by Prof. Shereen Taie

IT251
Information Technology Department

Lecture 7



Data Structures and Algorithms

Lecture 7

IT251

Information Technology Department

Prepared by
Prof. Shereen Taie

Learning Objectives:

- Representation of Queue (or) Implementation of Queue
- Queue using Array
- Queue operations using array
- Queue using Linked list
- Applications of Queue
- CIRCULAR QUEUE
- Representation of Circular Queue
- Operations on Circular queue

DATA STRUCTURES

Chapter

03

Representation of Queue (or) Implementation of Queue

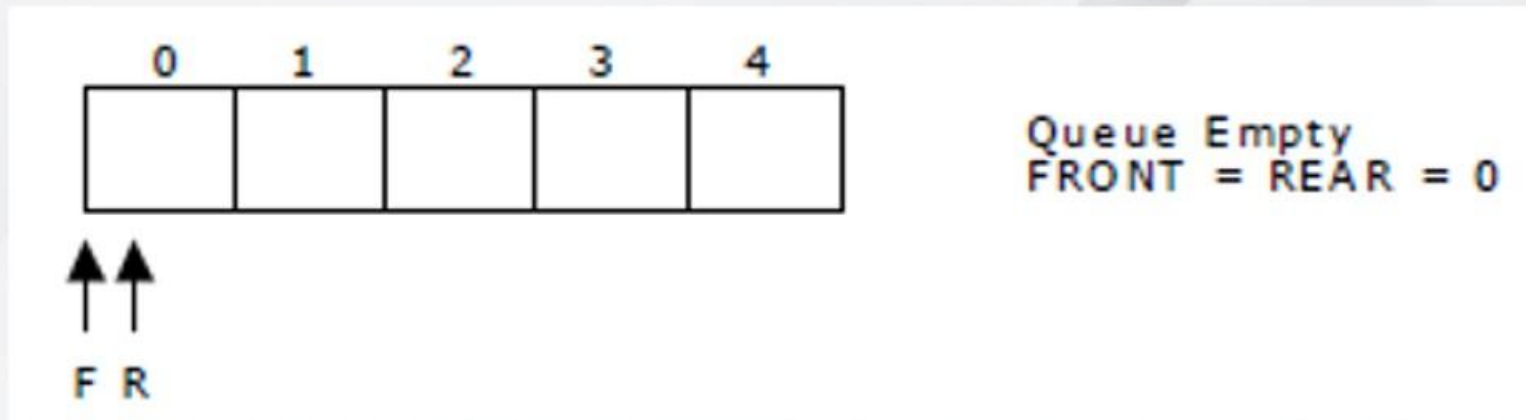
The queue can be represented in two ways:

1. Queue using Array
2. Queue using Linked List

1.Queue using Array:

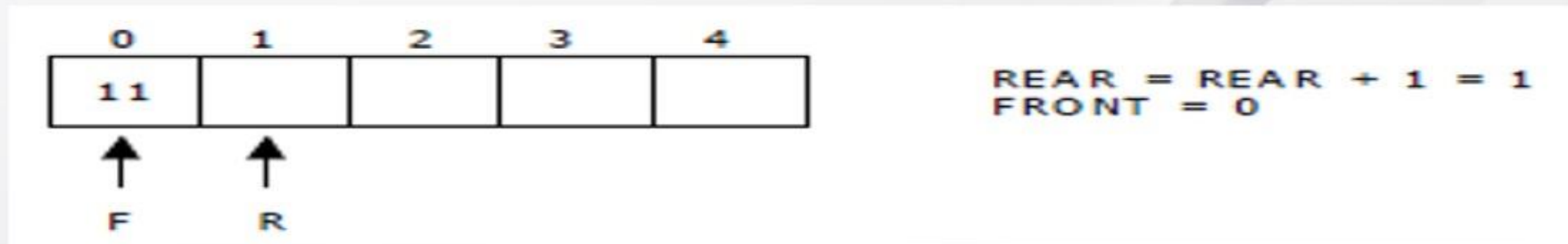
- Let us consider a queue, which can hold maximum of five elements.

Initially the queue is empty.

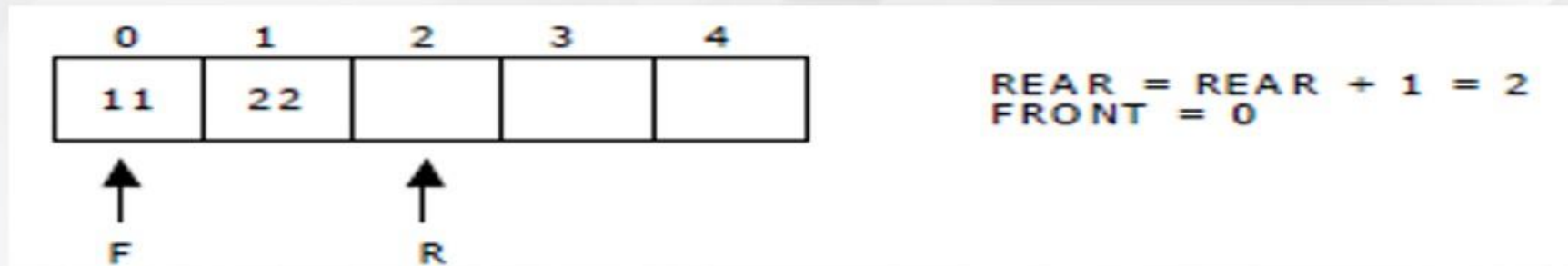


1.Queue using Array:

Now, insert 11 to the queue. Then queue status will be:

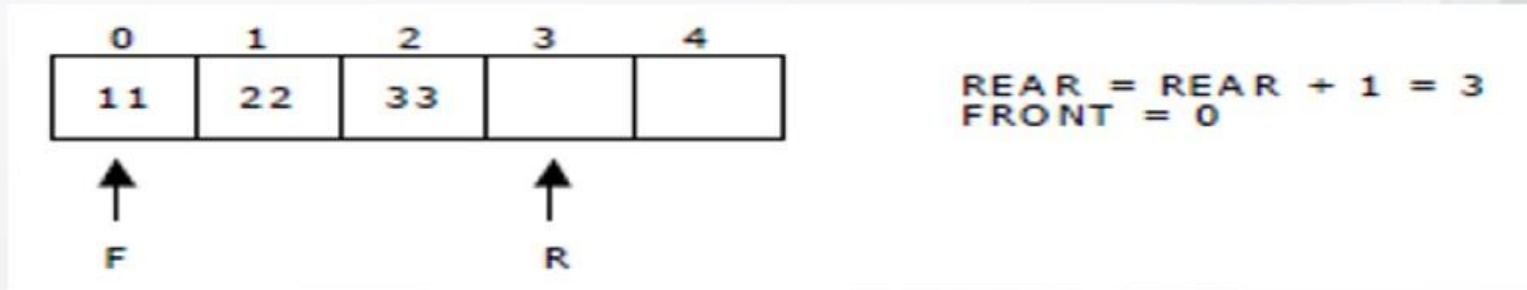


Next, insert 22 to the queue. Then the queue status is:



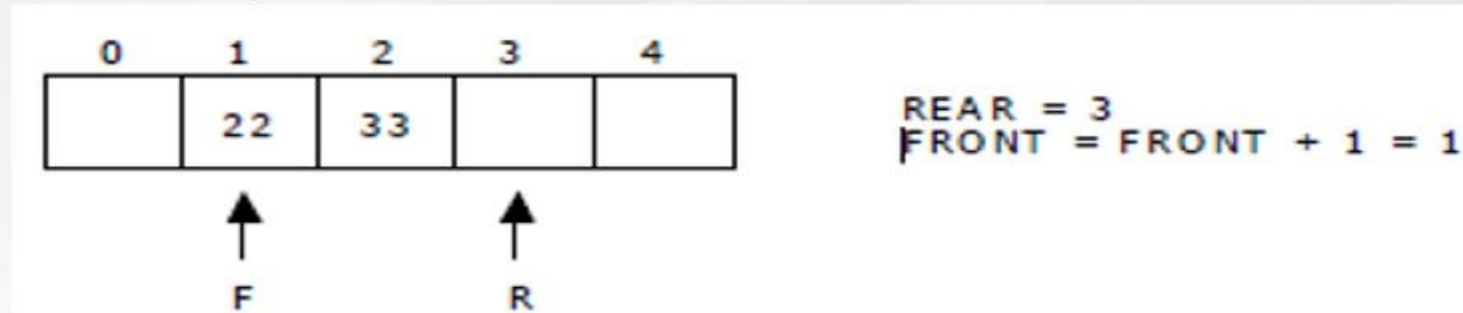
1.Queue using Array:

Again insert another element 33 to the queue. The status of the queue is:



Now, delete an element. The element deleted is the element at the front of the queue.

So the status of the queue is:

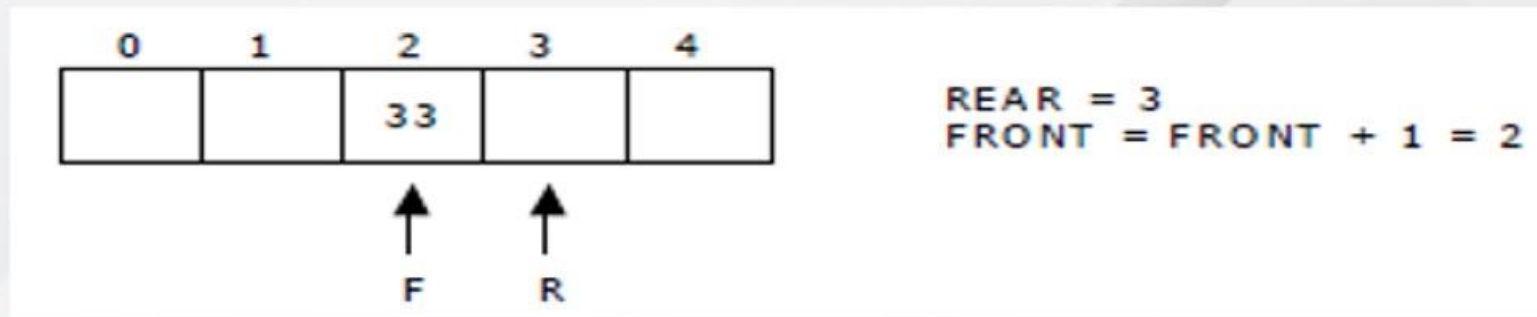


1.Queue using Array:

Again, delete an element.

The element to be deleted is always pointed to by the FRONT pointer.

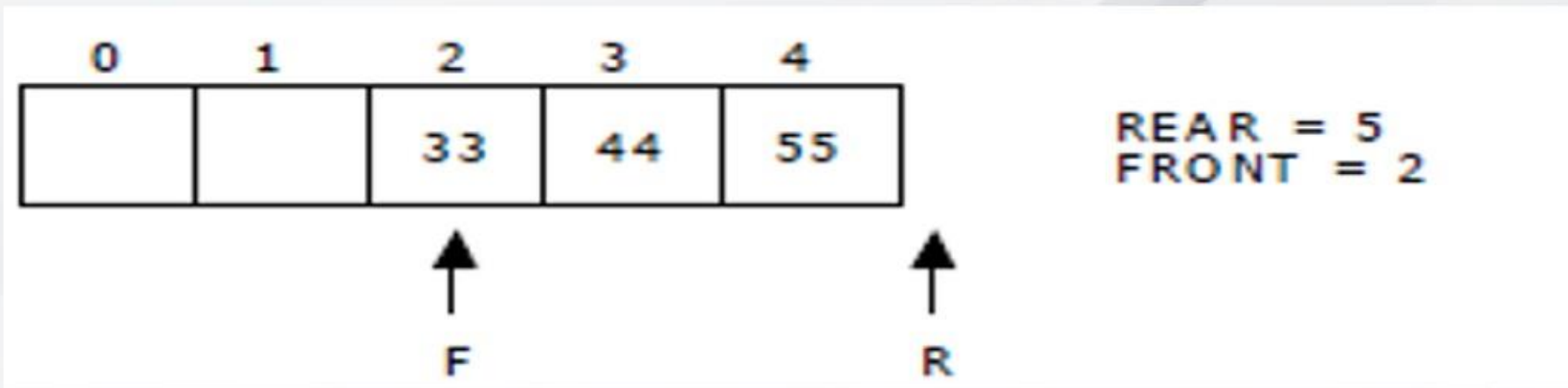
So, 22 is deleted. The queue status is as follows:



1.Queue using Array:

Now, insert new elements 44 and 55 into the queue.

The queue status is:

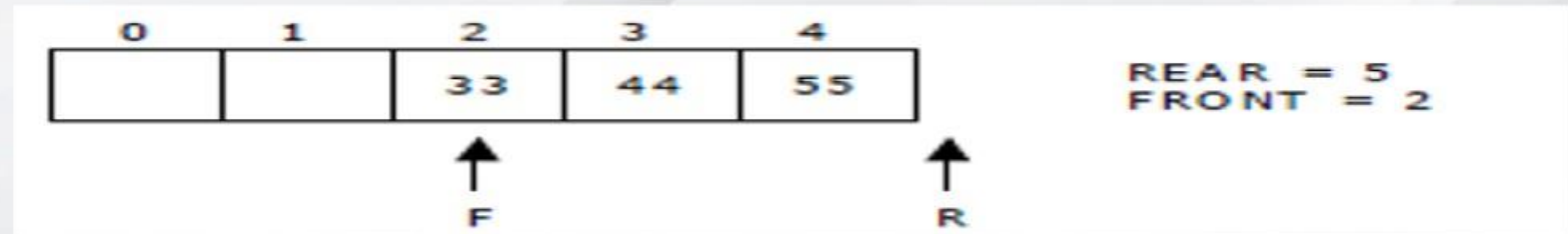


1.Queue using Array:

Next insert another element, say 66 to the queue.

We cannot insert 66 to the queue as the rear crossed the maximum size of the queue (i.e., 5). **There will be queue full signal, when the rear crossed the maximum size of the queue.**

The queue status is as follows:



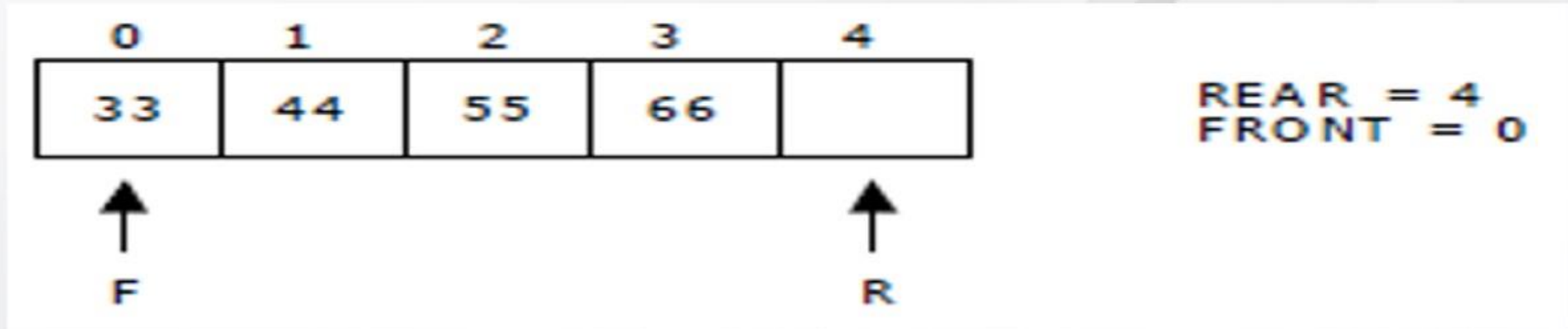
Now it is not possible to insert an element 66 even though there are two vacant positions in the linear queue.

1.Queue using Array:

- To overcome the problem of it is not possible to insert a new element even though there are two vacant positions in the linear queue.
- the elements of the queue are to be shifted towards the beginning of the queue so that it creates vacant position at the rear end.
- Then the FRONT and REAR are to be adjusted properly.

1.Queue using Array:

The element 66 can be inserted at the rear end. After this operation, the queue status is as follows:



This difficulty can overcome if we treat queue position with index 0 as a position that comes after position with index 4 i.e., we treat the queue as a **circular queue**.

Chapter
03

Queue operations using array



Queue operations using array:

1. enqueue() or insertion():which inserts an element at the end of the queue.

Algorithm: Procedure for insertion():

Step-1:START

Step-2: if rear==max then

Write 'Queue is full'

Step-3: otherwise

3.1: read element 'queue[rear]'

Step-4:STOP

Queue operations using array:

2. dequeue() or deletion(): which deletes an element at the start of the queue.

Algorithm: procedure for deletion():

Step-1:START

Step-2: if $\text{front} == \text{rear}$ then

 Write 'Queue is empty'

Step-3: otherwise

 3.1: print deleted element

Step-4:STOP

Queue operations using array:

3. display(): which displays an elements in the queue.

Algorithm: procedure for display():

Step-1:START

Step-2: if front==rear then

Write' Queue is empty'

Step-3: otherwise

3.1: for i=front to rear then

3.2: print 'queue[i]'

Step-4:STOP

Chapter
03

Queue using Linked list



2. Queue using Linked list:

- We can represent a queue as a linked list.
- In a queue data is deleted from the front end and inserted at the rear end.
- We can perform similar operations on the two ends of a list.
- We use two pointers front and rear for our linked queue implementation.

2. Queue using Linked list:

The linked queue looks as shown in figure:

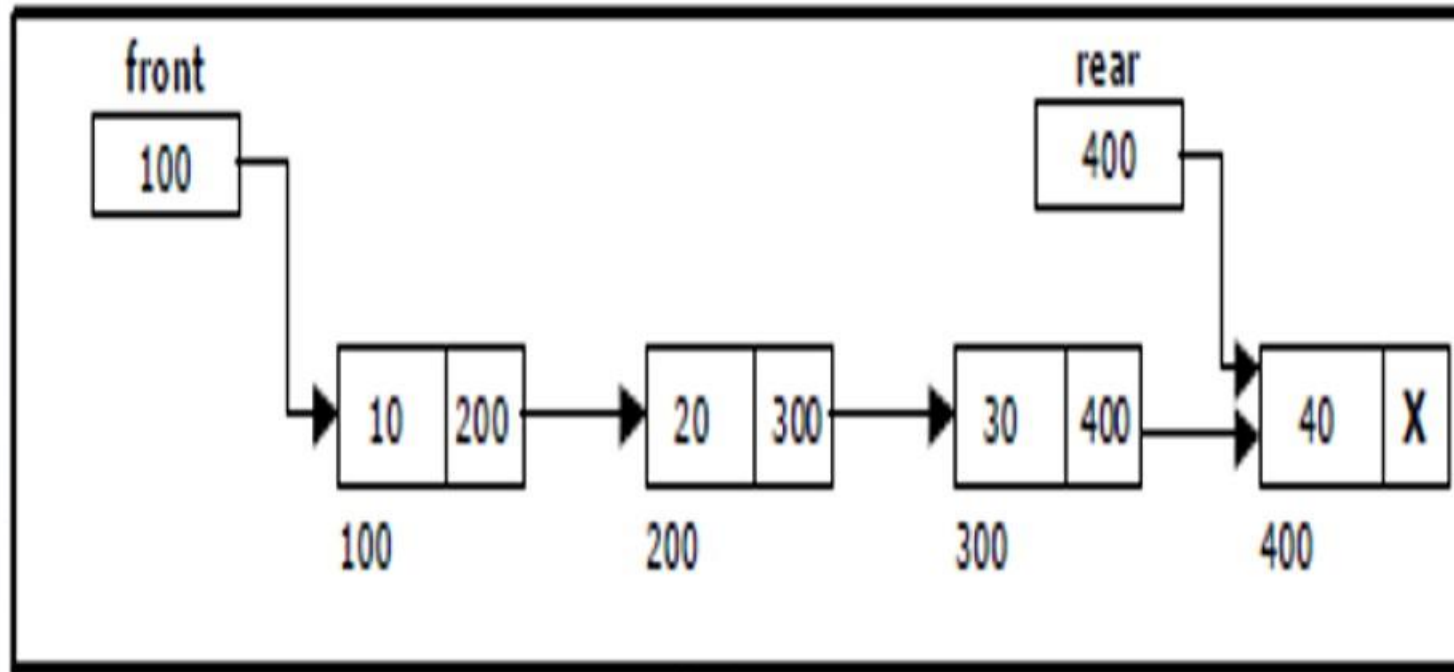


Figure : Linked Queue representation

Applications of Queue

Applications of Queue:

1. It is used to schedule the jobs to be processed by the CPU.
2. When multiple users send print jobs to a printer, each printing job is kept in the printing queue. Then the printer prints those jobs according to first in first out (FIFO) basis.
3. Breadth first search uses a queue data structure to find an element from a graph.

Chapter
03

CIRCULAR QUEUE



Circular Queue:

- A more efficient queue representation is obtained by regarding the array $Q[\text{MAX}]$ as circular.
- Any number of items could be placed on the queue.
- This implementation of a queue is called a circular queue because it uses its storage array as if it were a circle instead of a linear list.

Circular Queue:

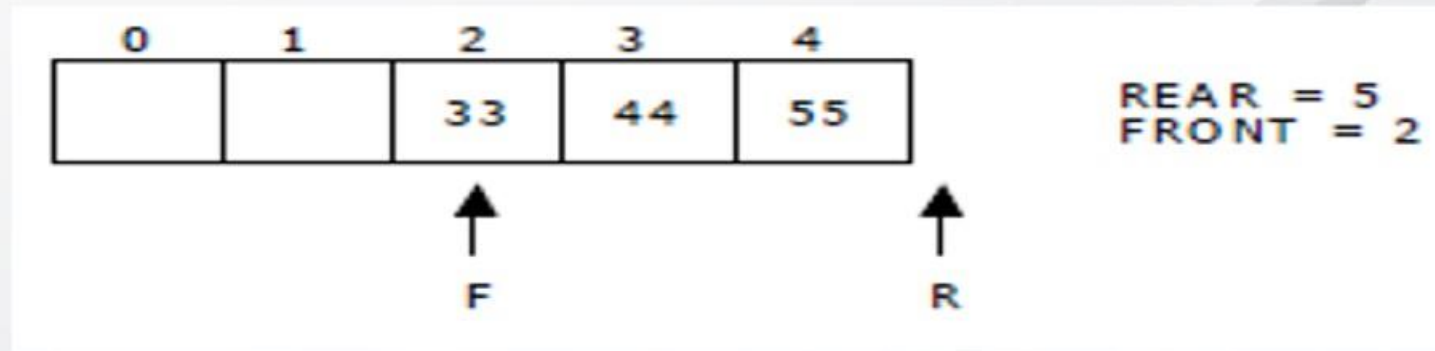
There are two problems associated with linear queue.

They are:

- Time consuming: linear time to be spent in shifting the elements to the beginning of the queue.
- Signaling queue full: even if the queue is having vacant position.

Circular Queue:

For example, let us consider a linear queue status as follows:

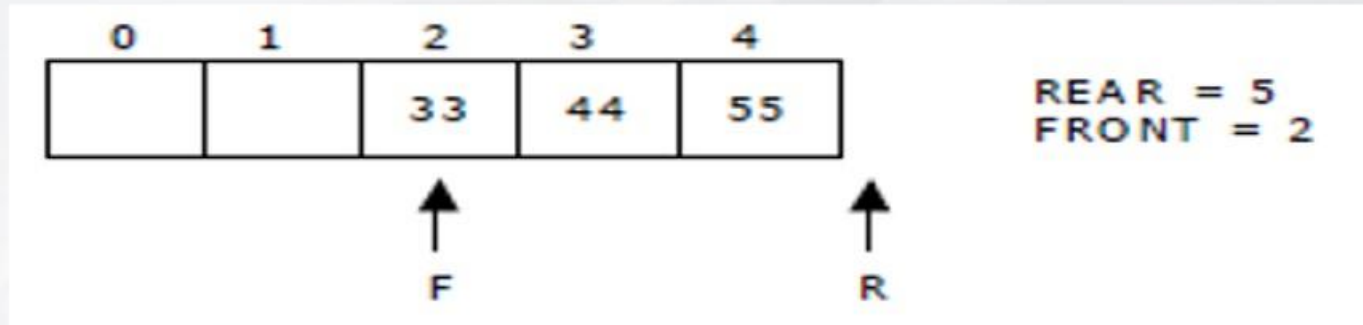


Next insert another element, say 66 to the queue.

Circular Queue:

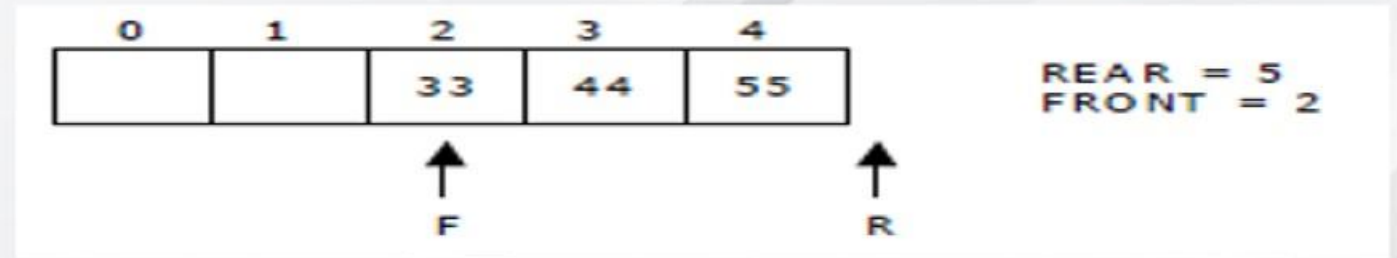
- We cannot insert 66 to the queue as the rear crossed the maximum size of the queue (i.e., 5).
- There will be queue full signal.

The queue status is as follows:



Circular Queue:

- This difficulty can be overcome if we treat queue position with index zero as a position that comes after position with index four then we treat the queue as a **circular queue**.

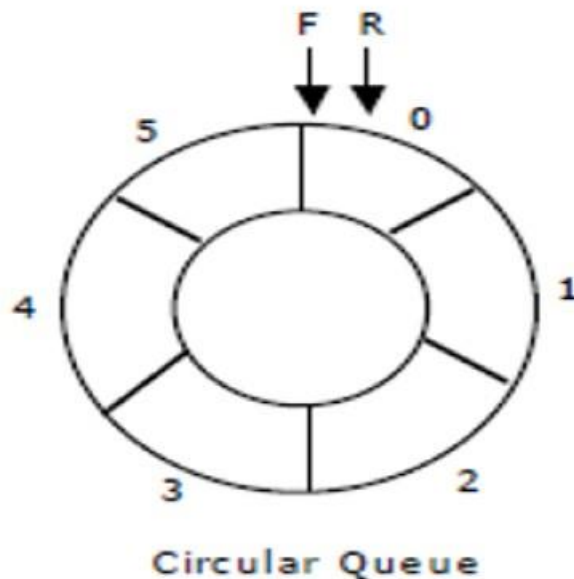


- In circular queue if we reach the end for inserting elements to it, it is possible to insert new elements if the slots at the beginning of the circular queue are empty.

Representation of Circular Queue

Representation of Circular Queue:

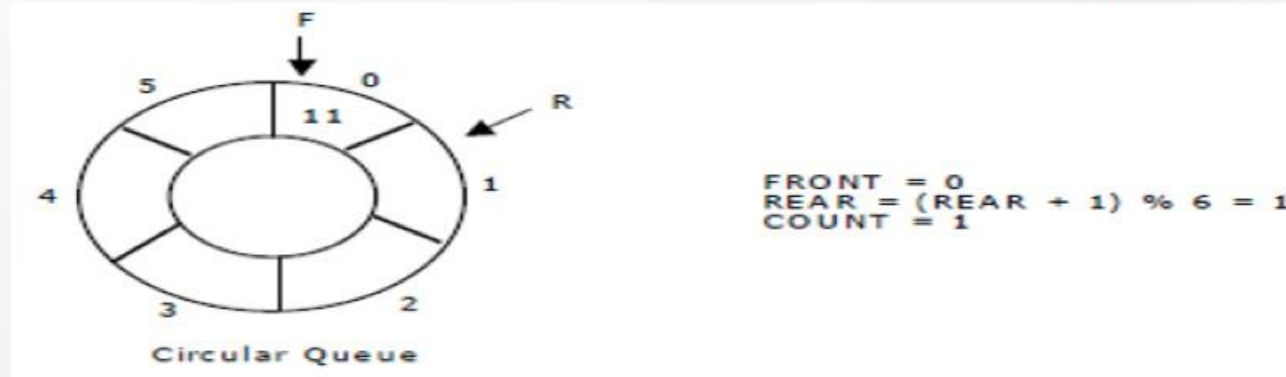
- Let us consider a circular queue, which can hold maximum (MAX) of six elements.
- Initially the queue is empty.



Queue Empty
MAX = 6
FRONT = REAR = 0
COUNT = 0

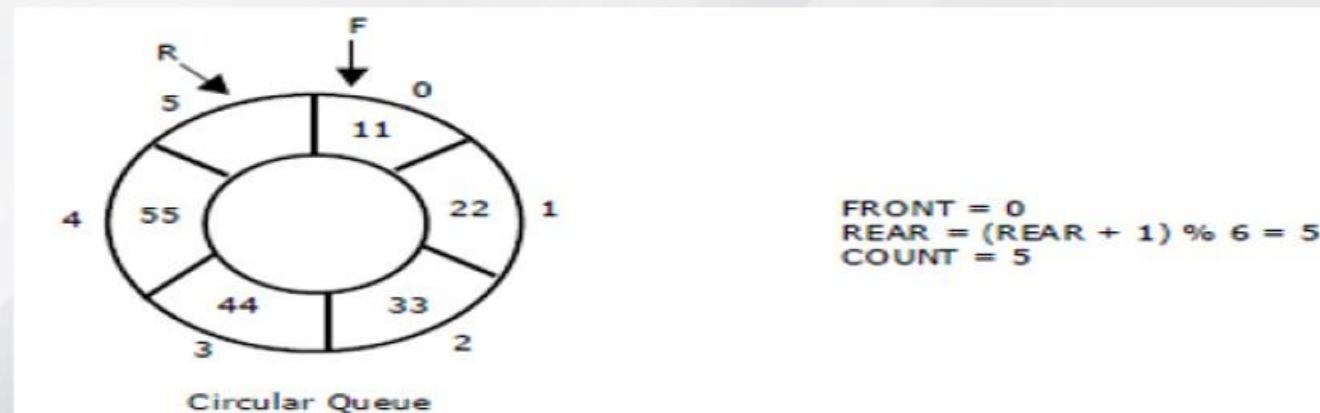
Representation of Circular Queue:

- Now, insert 11 to the circular queue. Then circular queue status will be:



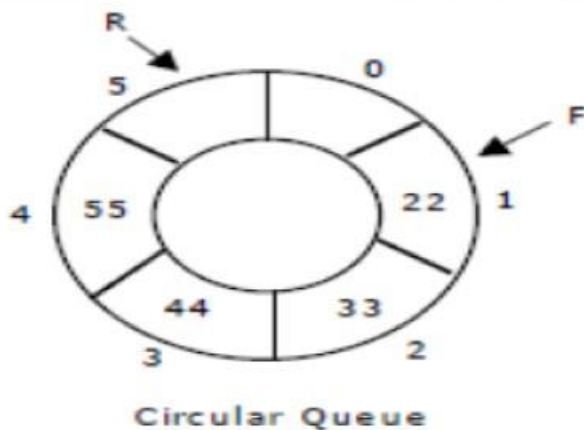
$6 \% 6 = 0$
 $12 \% 6 = 0$
 $7 \% 6 = 1$
 $1 \% 6 = 1$
 $5 \% 6 = 5$

- Insert new elements 22, 33, 44 and 55 into the circular queue. The circular queue status is:



Representation of Circular Queue:

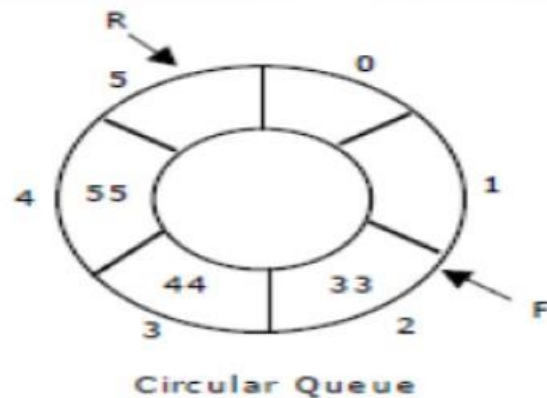
- **Now, delete an element.** The element deleted is the element at the front of the circular queue. So, 11 is deleted. The circular queue status is as follows:



```
FRONT = (FRONT + 1) % 6 = 1  
REAR = 5  
COUNT = COUNT - 1 = 4
```

Representation of Circular Queue:

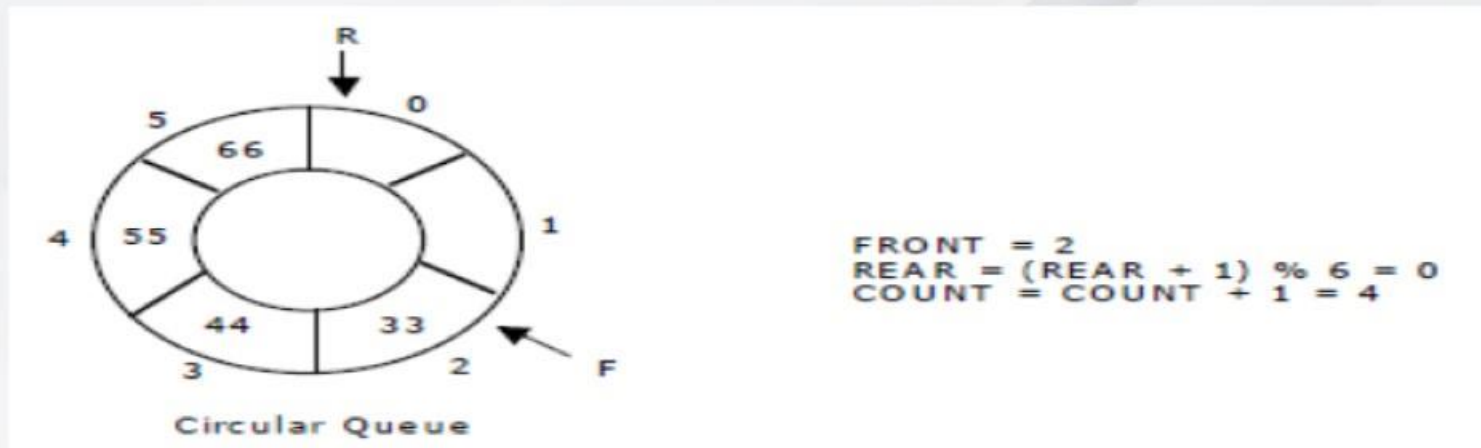
- Again, delete an element. The element to be deleted is always pointed to by the FRONT pointer. So, 22 is deleted. The circular queue status is as follows:



```
FRONT = (FRONT + 1) % 6 = 2
REAR = 5
COUNT = COUNT - 1 = 3
```

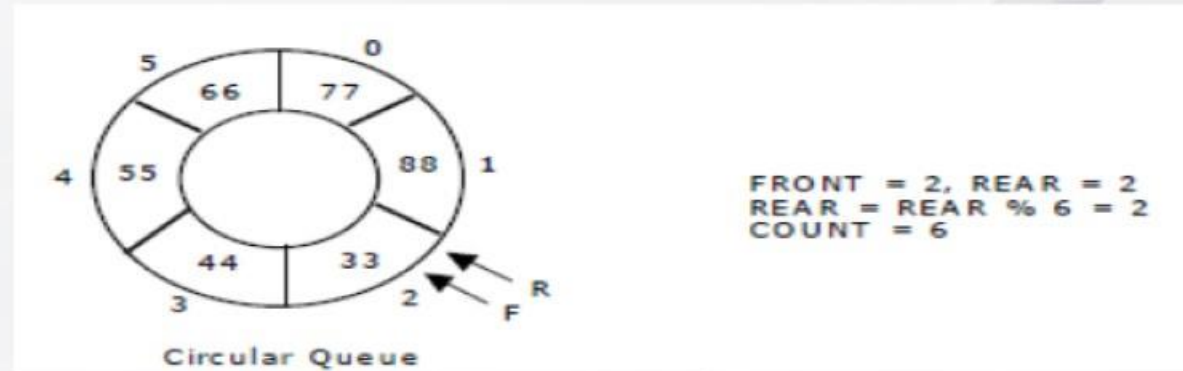
Representation of Circular Queue:

- Again, insert another element 66 to the circular queue. The status of the circular queue is:



Representation of Circular Queue:

- Now, insert new elements 77 and 88 into the circular queue. The circular queue status is:



- Now, if we insert an element to the circular queue, as $COUNT = MAX$ we cannot add the element to circular queue. So, the circular queue is *full*.

Operations on Circular queue

Operations on Circular queue:

1. **enqueue() or insertion():** This function is used to insert an element into the circular queue. In a circular queue, the new element is always inserted at Rear position.

Algorithm: procedure of insertCQ():

Step-1: START

Step-2: if count==MAX then

Write "Circular queue is full"

Step-3: otherwise

3.1: read the data element

3.2: $CQ[rear] = data$

3.3: $rear = (rear + 1) \% MAX$

3.4: count=count+1

Step-4: STOP

Operations on Circular queue:

2. **dequeue() or deletion():** This function is used to delete an element from the circular queue. In a circular queue, the element is always deleted from front position.

Algorithm: procedure of deleteCQ():

Step-1: START

Step-2: if count==0 then

Write "Circular queue is empty"

Step-3: otherwise

3.1: print the deleted element

3.2: $\text{front} = (\text{front} + 1) \% \text{MAX}$

3.3: count=count-1

Step-4: STOP

Operations on Circular queue:

3. **dispalY():**This function is used to display the list of elements in the circular queue.

Algorithm: procedure of display CQ():

Step-1:START

Step-2: if count==0 then

Write "Circular queue is empty"

Step-3:otherwise

3.1: print the list of elements

3.2: for i=front to j!=0

3.3: print CQ[i]

3.4: $i=(i+1)\%MAX$

Step-4:STOP

Thank You