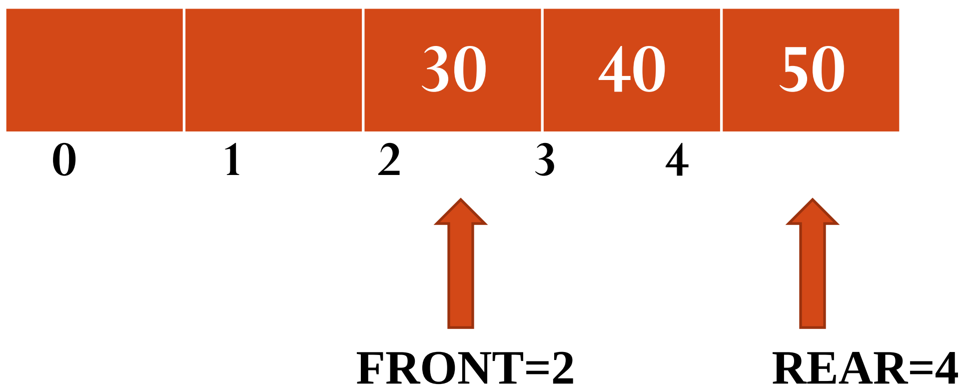
Data Structures – CST 201
Module - 2

Syllabus

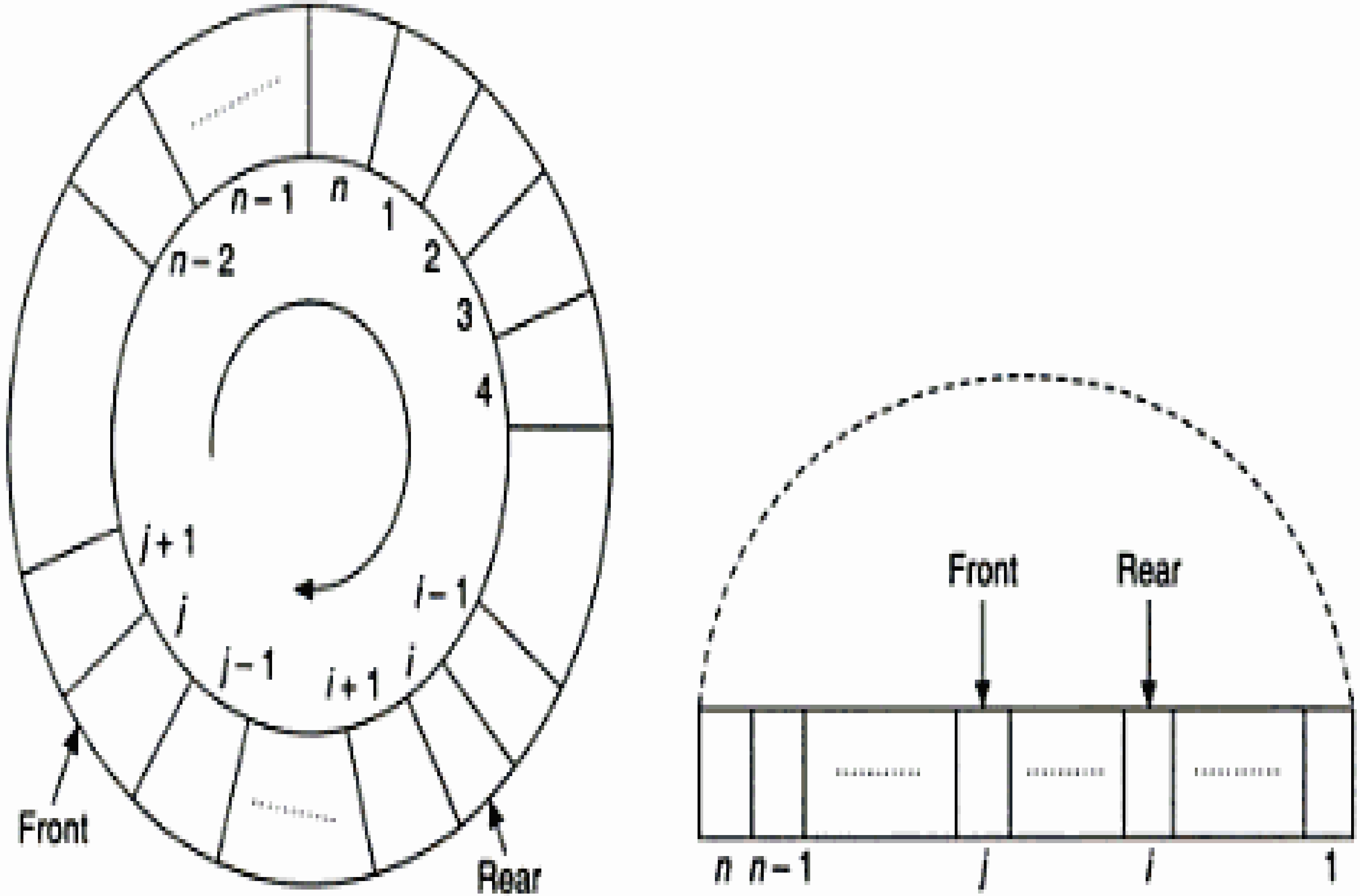
- Polynomial representation using Arrays
- Sparse matrix
- Stacks
  - Evaluation of Expressions
- Queues
  - **Circular Queues**
  - Priority Queues
  - Double Ended Queues,
- Linear Search
- Binary Search

CIRCULAR QUEUE

- In queue(represented using array) when the rear pointer reaches at the end, insertion is denied even if there is room at the front



- To avoid this problem we can use circular queue
- Physically Circular Array is same as ordinary array



(a) Circular queue (logical)

(b) Circular array (physical)

Figure 5.7 Logical and physical views of a circular queue.

CIRCULAR QUEUE - PRINCIPLE

- Both pointers will move in clockwise direction
- This is controlled by the MOD operation
- For example if the current location is i then move to the next location by **(i mod LENGTH) +1** where **1<=i<=LENGTH**

CIRCULAR QUEUE- Operations

- **ENQUEUE**: Insert an element into Circular Queue
- **DEQUEUE**: Delete an element from the Circular Queue
- **DISPLAY**: Display the contents of the Circular Queue

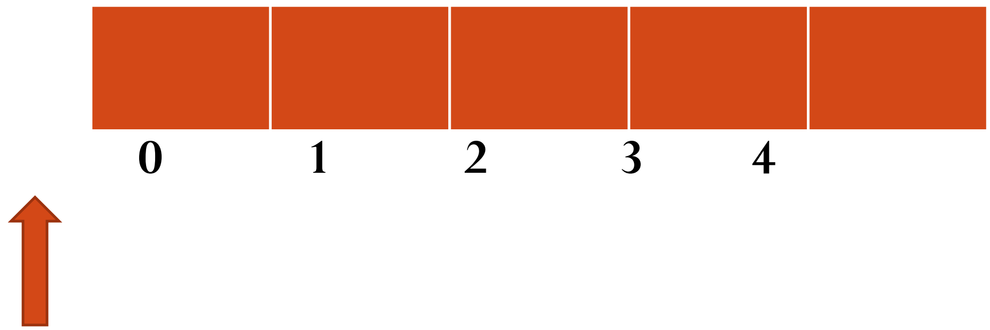
CIRCULAR QUEUE- Representations

- Two Representations
  - Array Representation
  - Linked List Representation

int A[5];

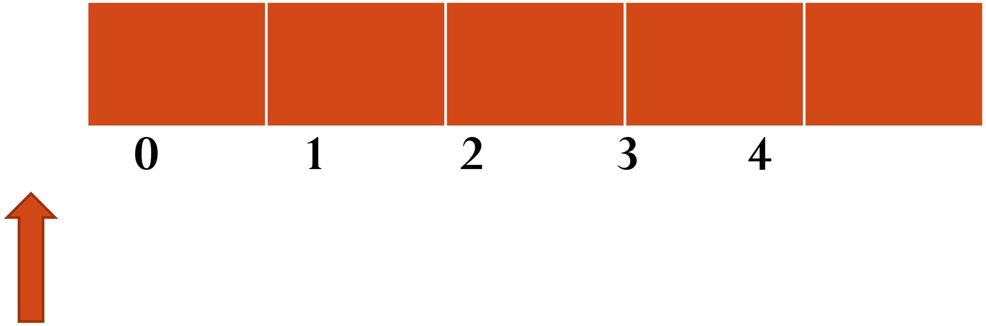
If FRONT=-1 Or REAR=-1 then

Queue is EMPTY



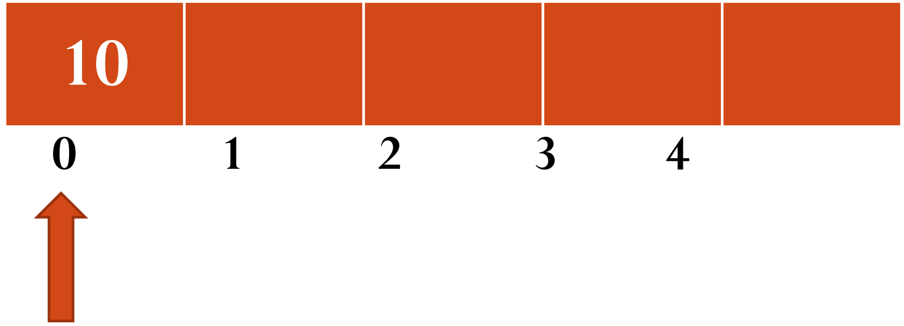
FRONT=-1
REAR=-1

ENQUEUE 10



FRONT=-1

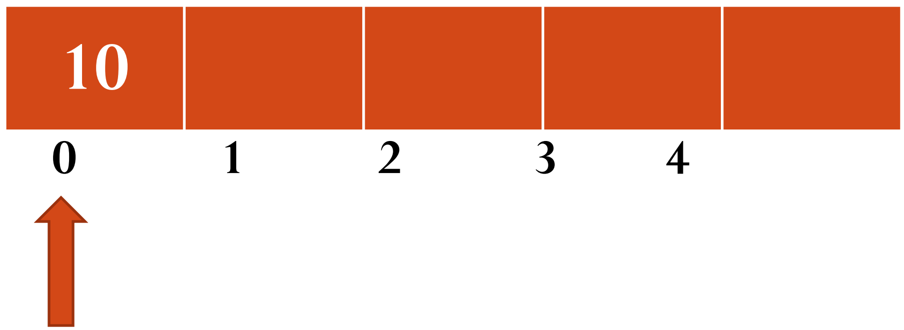
REAR=-1



ENQUEUE 10

FRONT=0

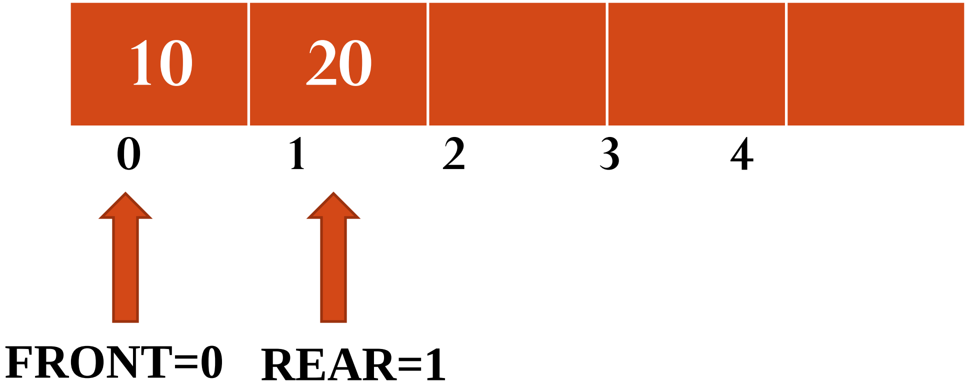
REAR=0



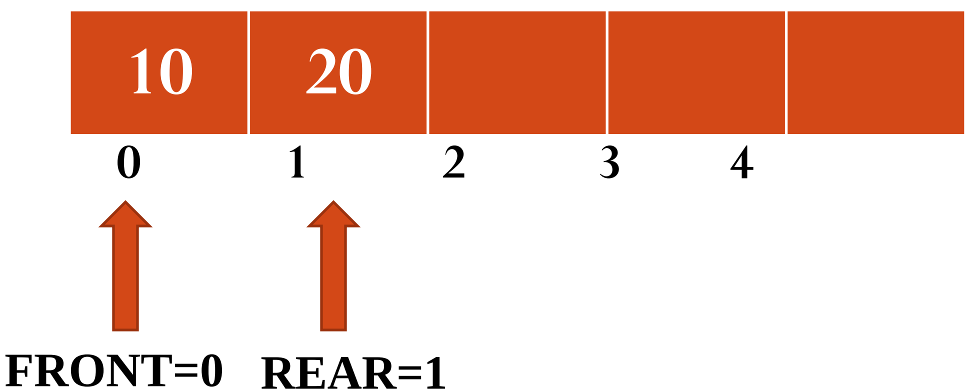
ENQUEUE 20

FRONT=0

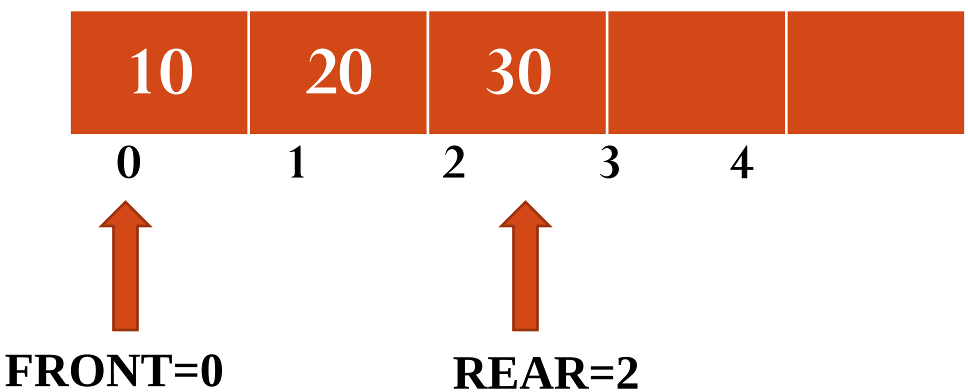
REAR=0



ENQUEUE 20

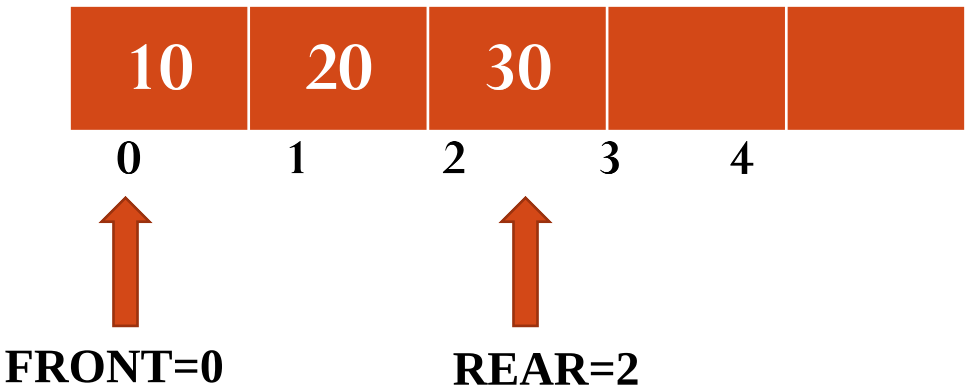


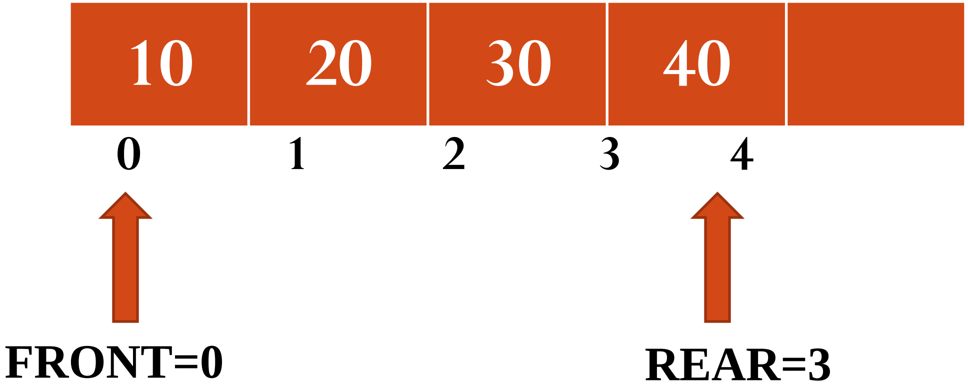
ENQUEUE 30



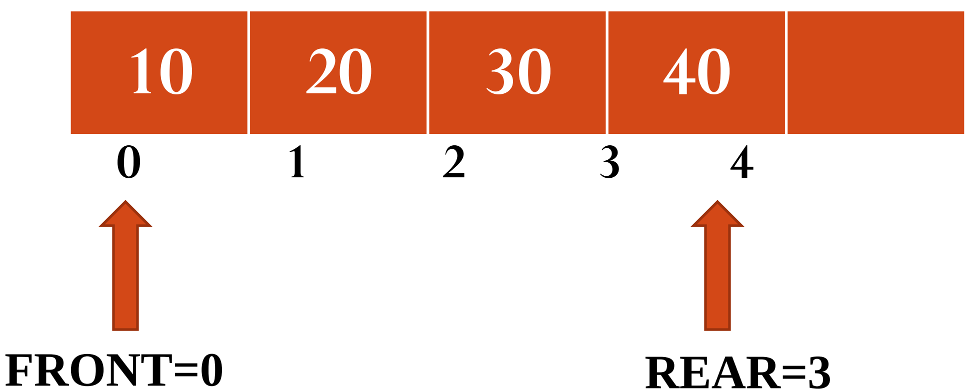
ENQUEUE 30

ENQUEUE 40

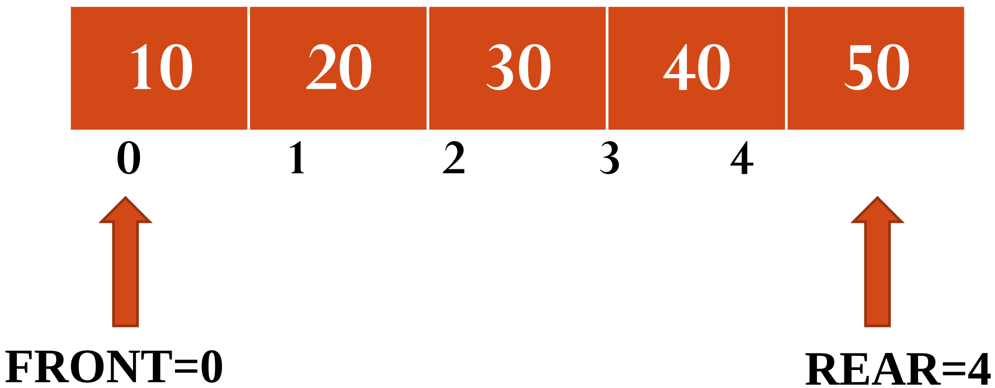




ENQUEUE 40

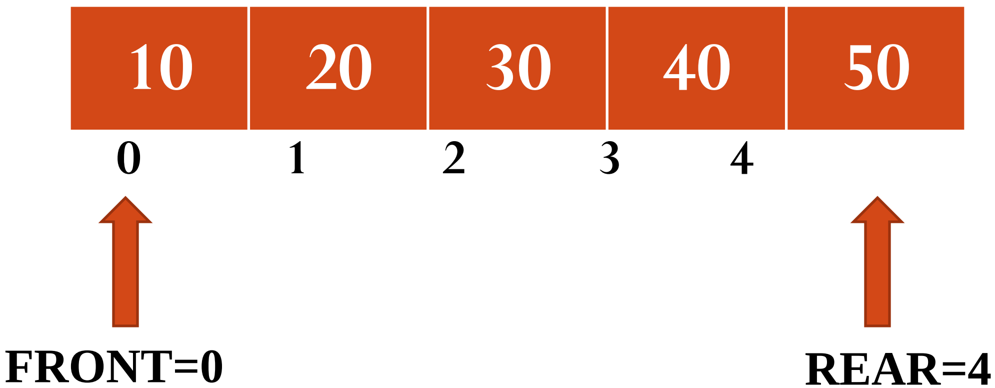


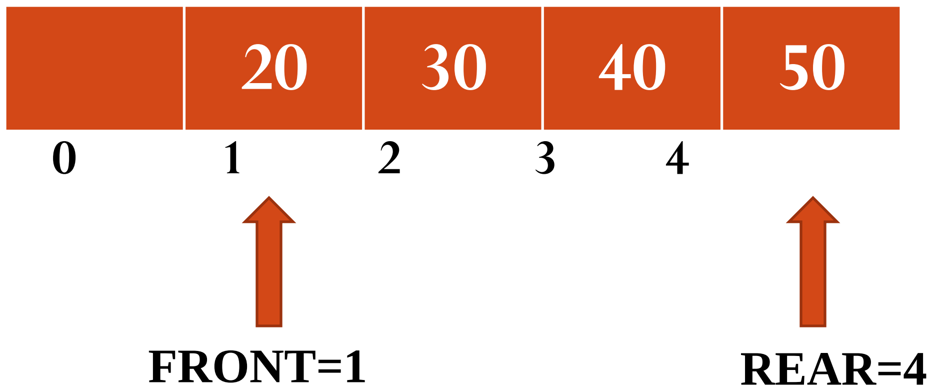
ENQUEUE 50

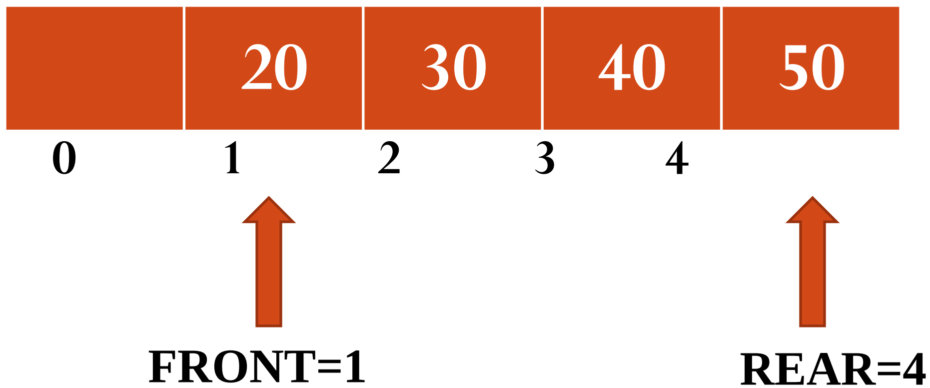


ENQUEUE 50

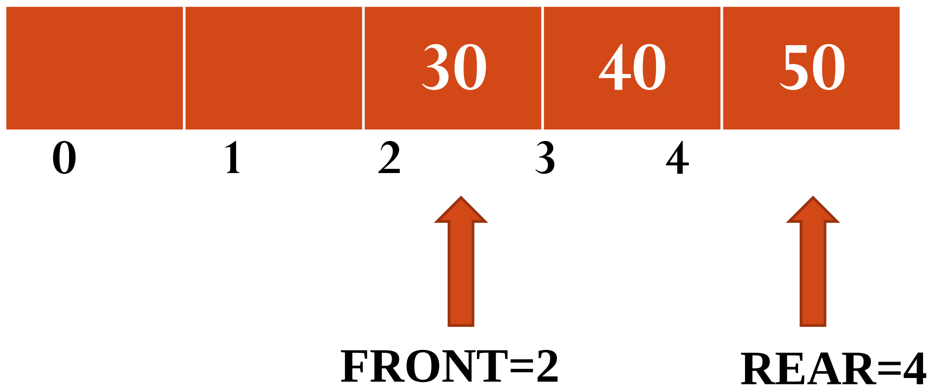
DEQUEUE



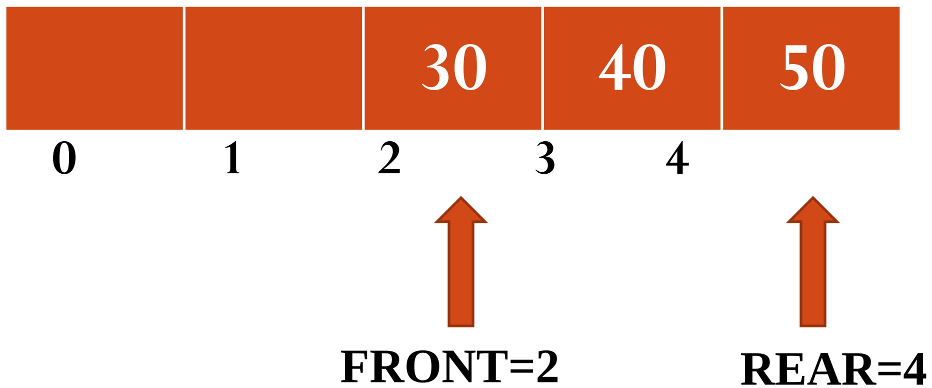

20
30
40
50
0
1
2
3
4
FRONT=1
REAR=4



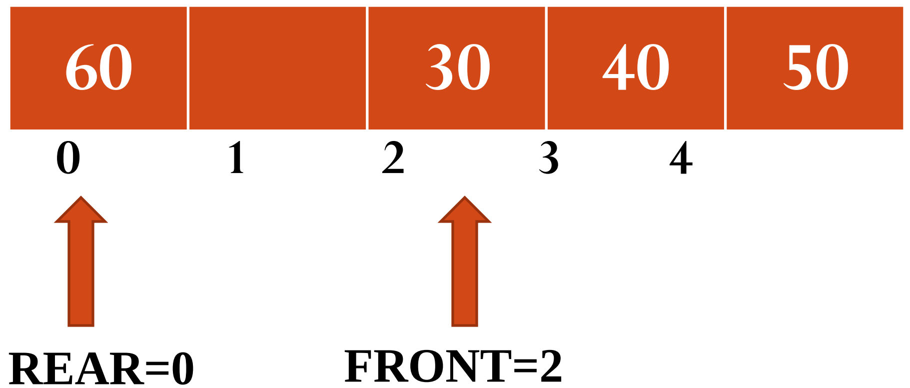
DEQUEUE

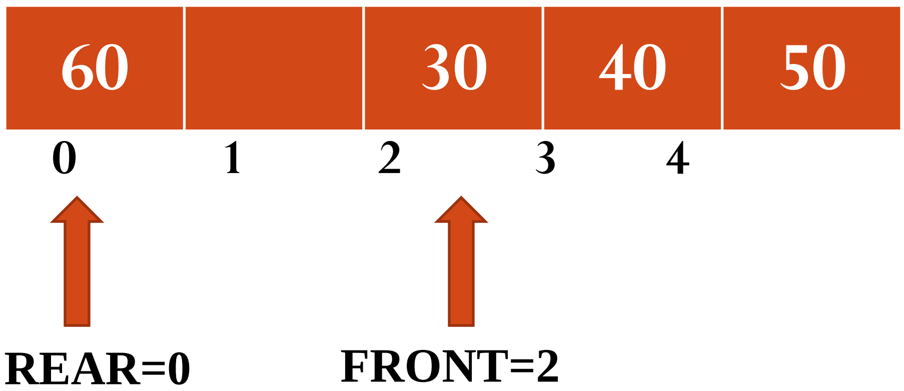


30
40
50
0
1
2
3
4
FRONT=2
REAR=4

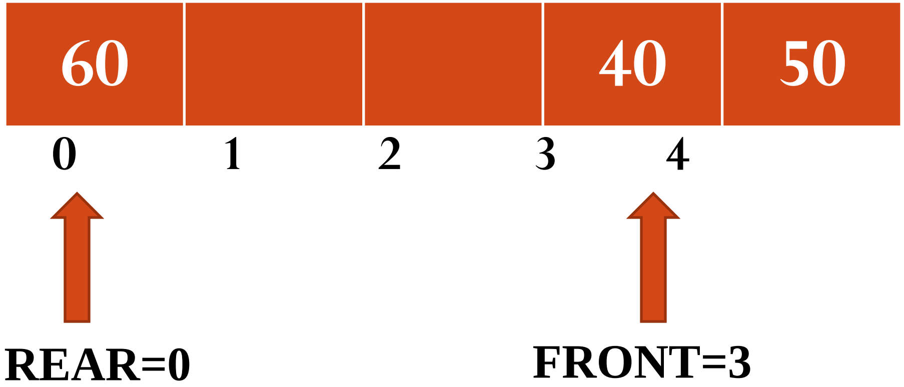


ENQUEUE 60

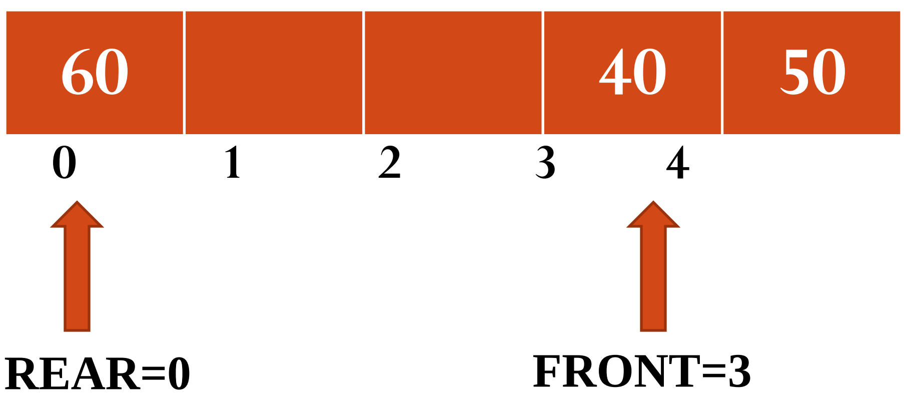

60
30
40
50
0
1
2
3
4
REAR=0
FRONT=2

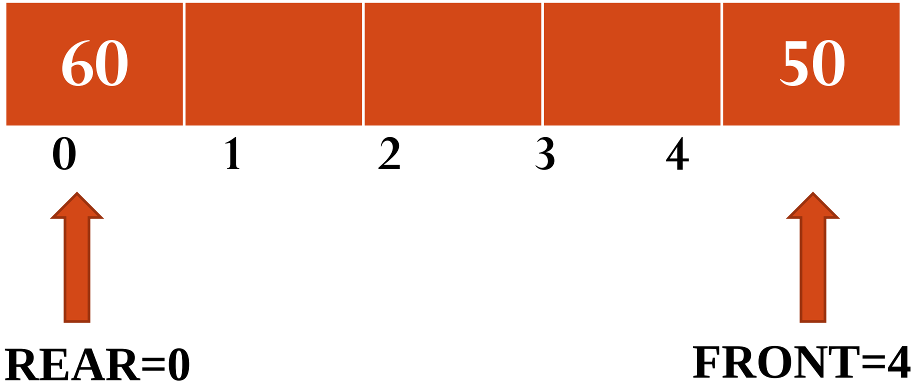


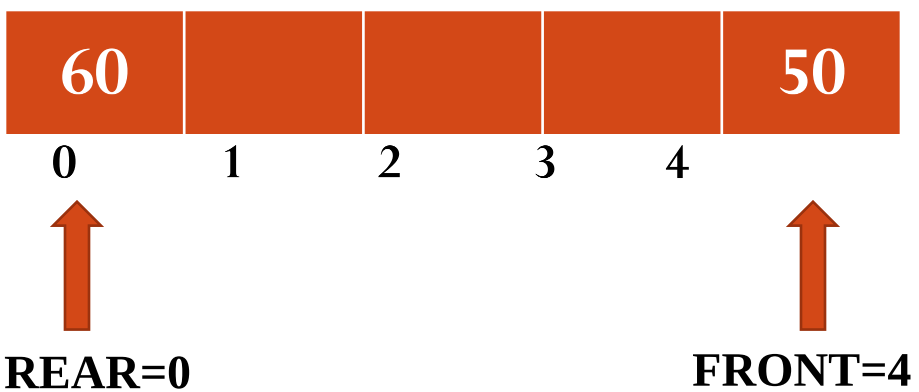
DEQUEUE


60
40
50
0
1
2
3
4
REAR=0
FRONT=3

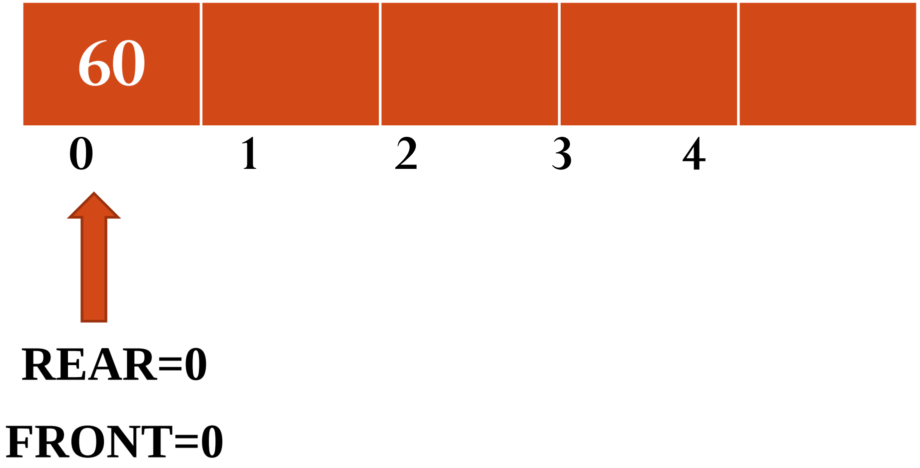
DEQUEUE



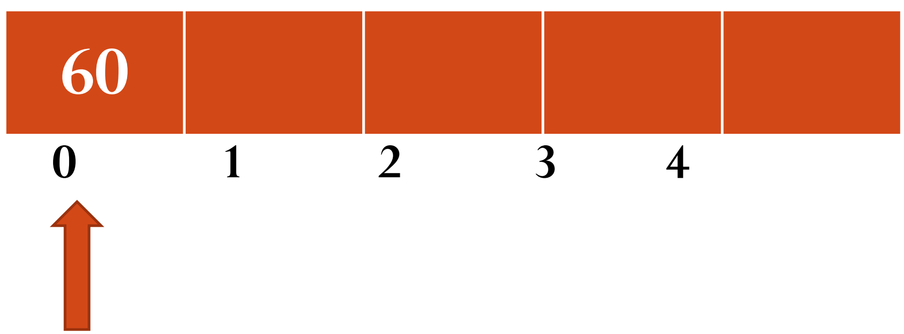

60
50
0
1
2
3
4
REAR=0
FRONT=4



DEQUEUE

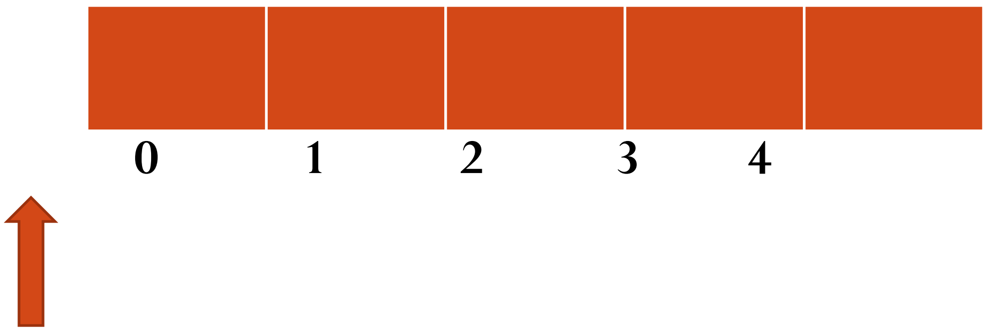

60
0
1
2
3
4
REAR=0
FRONT=0

DEQUEUE



REAR=0

FRONT=0



REAR=-1

FRONT=-1

CIRCULAR QUEUE- Various States

1. Queue is Empty: FRONT=-1 & REAR=-1
2. Queue is Full: FRONT = (REAR+1)%SIZE
3. Queue contains only one element: FRONT=REAR
4. Total elements in the queue
   - FRONT <= REAR: REAR-FRONT+1
   - FRONT>REAR: SIZE-FRONT+REAR+1
5. Increment FRONT by one: FRONT=(FRONT+1)%SIZE
6. Increment REAR by one: REAR=(REAR+1)%SIZE

CIRCULAR QUEUE – ENQUEUE

Algorithm ENQUEUE(ITEM)

```
{       if  (REAR + 1) % SIZE = FRONT then
                Print “Queue is FULL”
        else if FRONT=-1 then   // Presently Queue is empty
        {       FRONT=REAR=0
                A[REAR]=ITEM
        }
        else
        {       REAR = (REAR + 1) % SIZE
                A[REAR] = ITEM
        }
}
```

CIRCULAR QUEUE – DEQUEUE

Algorithm DEQUEUE()

```
{
        if  FRONT = -1 then
                Print "Queue is EMPTY"
        else if FRONT = REAR then  //Queue contains only one element
        {       Print "Dequeued item is " A[FRONT]
                FRONT = REAR = -1
        }
        else
        {       Print "Dequeued item is " A[FRONT]
                FRONT = (FRONT+1)%SIZE
        }
}
```

CIRCULAR QUEUE – DISPLAY

Algorithm DISPLAY()

```
{       if  FRONT = -1 then
                Print "Queue is EMPTY"
        else
        {       if FRONT <= REAR then
                {       for i=FRONT to REAR do
                                Print A[i]
                }
                else
                {       for i=FRONT to SIZE-1 do
                                Print A[i]
                        for i=0 to REAR do
                                Print A[i]
                }
        }
}
```

In order to trace these two algorithms, let us consider a circular queue of LENGTH = 4. The following operations are requested. Different states of the queue while processing these requests are illustrated in Figure 5.8.

1. ENCQUEUE (A)
2. ENCQUEUE (B)
3. ENCQUEUE (C)
4. ENCQUEUE (D)
5. DECQUEUE
6. ENCQUEUE (E)
7. DECQUEUE
8. ENCQUEUE (F)
9. DECQUEUE
10. DECQUEUE
11. DECQUEUE
12. DECQUEUE