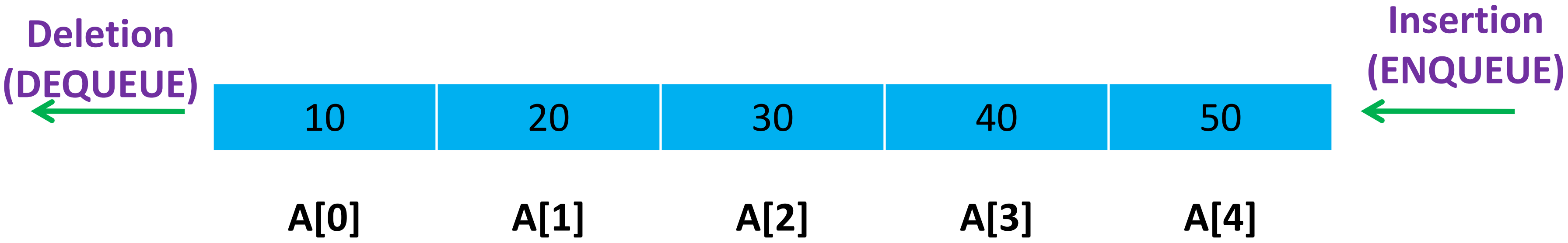
Subject : Data Structures
Topic : Queue

Queue

- Queue is Linear Data Structure
- It follows First In First Out(FIFO) principal
- It has two pointers front and rear

e.g.:



Deletion
(DEQUEUE)

Insertion
(ENQUEUE)

10	20	30	40	50
A[0]	**A[1]**	**A[2]**	**A[3]**	**A[4]**

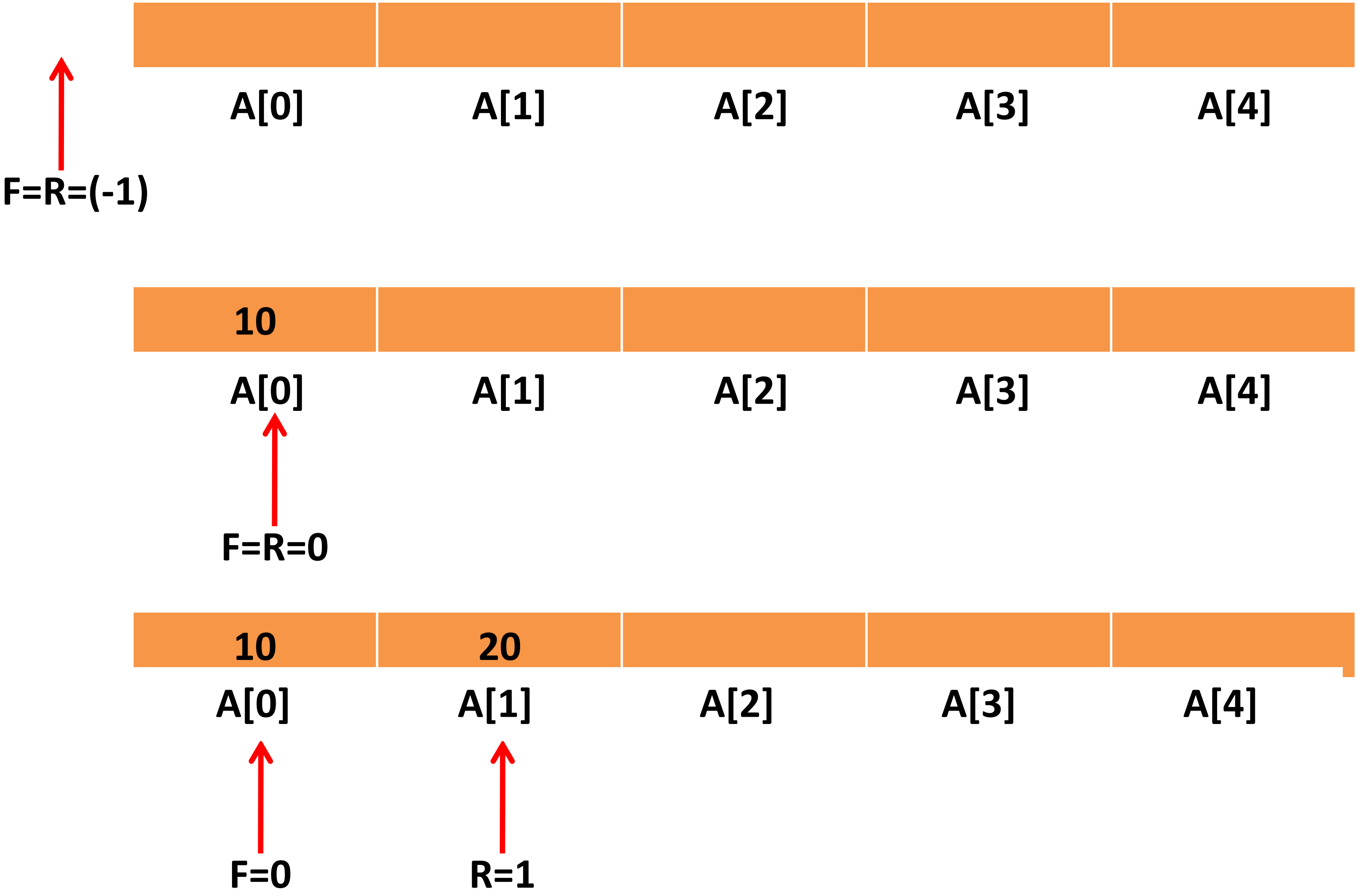
Operations on Queue

Insertion:

Algorithm:

```
Step 1: If REAR = MAX – 1 then
          Write “Queue is Overflow”
          Goto step 4
          [End of IF]
Step 2: IF FRONT=-1 and REAR=-1
                   SET FRONT=REAR=0
         ELSE
                   SET REAR=REAR+1
         [END OF IF]
Step 3: SET QUEUE [REAR] = NUM
Step 4: EXIT
```

Example of Insertion in Queue



A[0] A[1] A[2] A[3] A[4]

F=R=(-1)

10

A[0] A[1] A[2] A[3] A[4]

F=R=0

10 20

A[0] A[1] A[2] A[3] A[4]

F=0

R=1

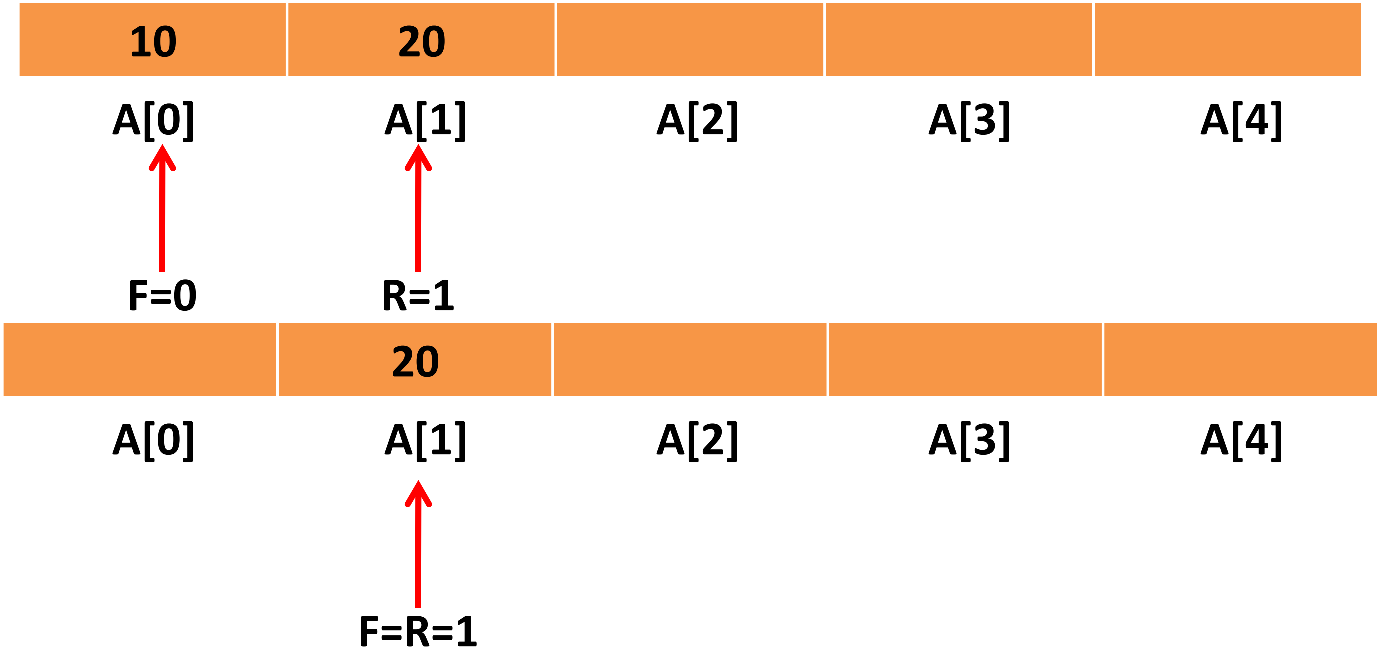
Operations on Queue

Deletion:

Algorithm:

```
Step 1: IF FRONT = -1 OR FRONT>REAR
            Write "Queue is Underflow"
        ELSE
            SET VAL=QUEUE [FRONT]
            FRONT = FRONT + 1
        [END OF IF]
Step 2: EXIT
```

Example of Deletion in Queue



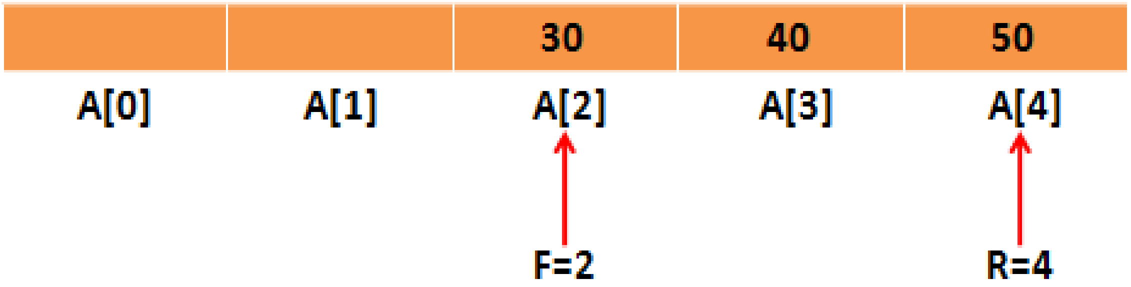
Types Of Queue

1. Circular Queue
2. Priority Queue
3. Deque
4. Multiple Queue

Circular Queue

Why Circular Queue is needed?

- Problem:
  - Wastage of memory in standard queue in DEQUEUE operation

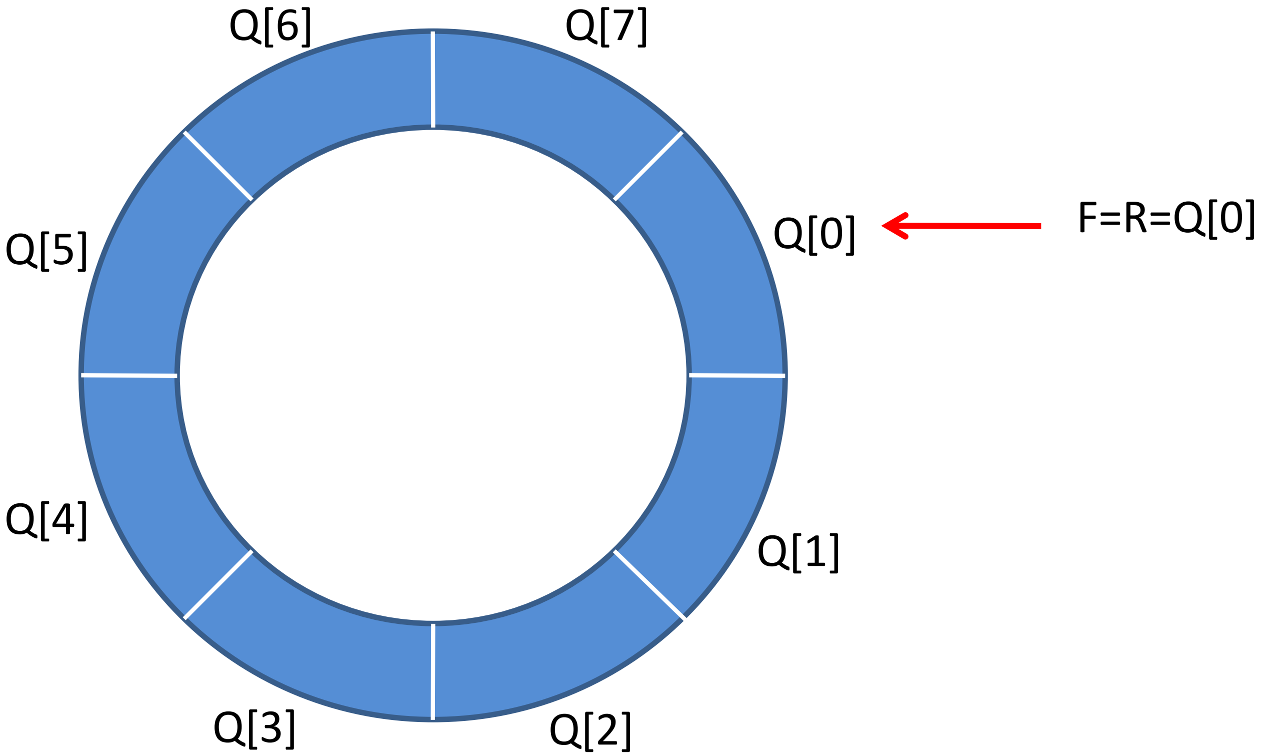


		30	40	50
A[0]	A[1]	A[2]	A[3]	A[4]

What is Circular Queue?

- The Arrangement of the elements Q[0], Q[1], ...,Q[n] in a circular fashion with Q[1] following Q[n] is called Circular Queue.
- The **last node is connected to first node** to make a circle.
- Initially, Both Front and Rear pointers points to the beginning of the array.
- It is also known as "<u>Ring Buffer</u>".

- e.g.:



F=R=Q[0]

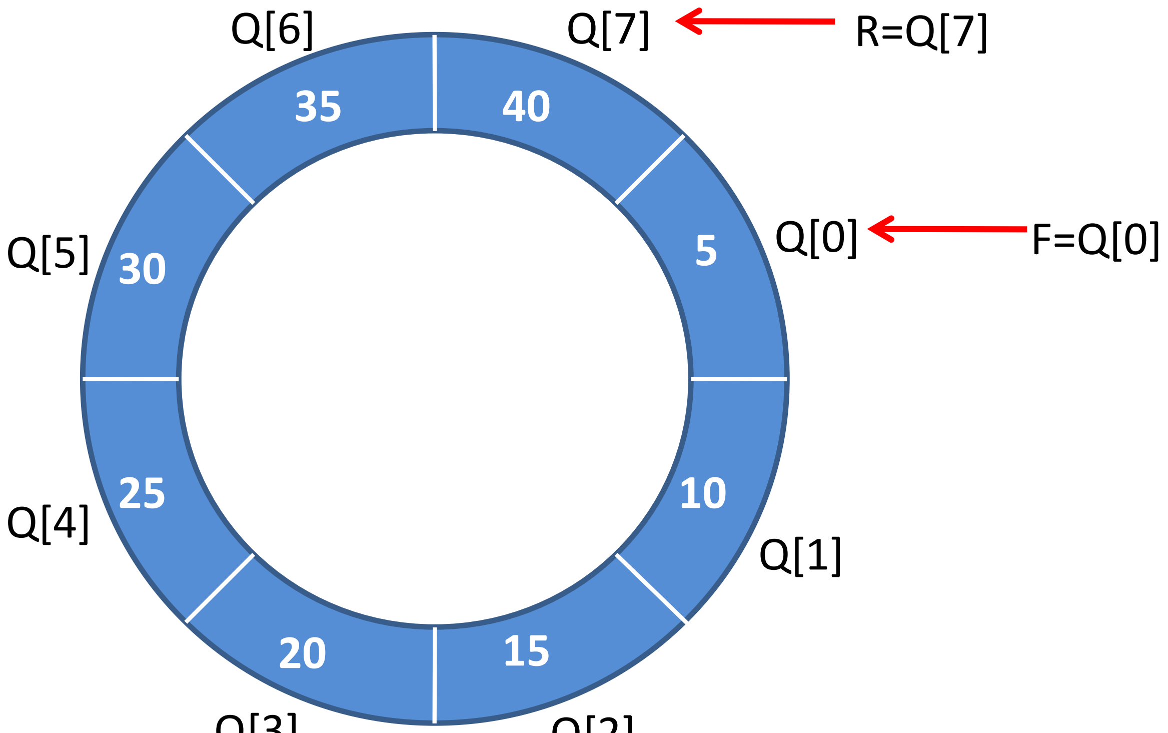
Operations on Circular Queue

Insertion:

Algorithm:

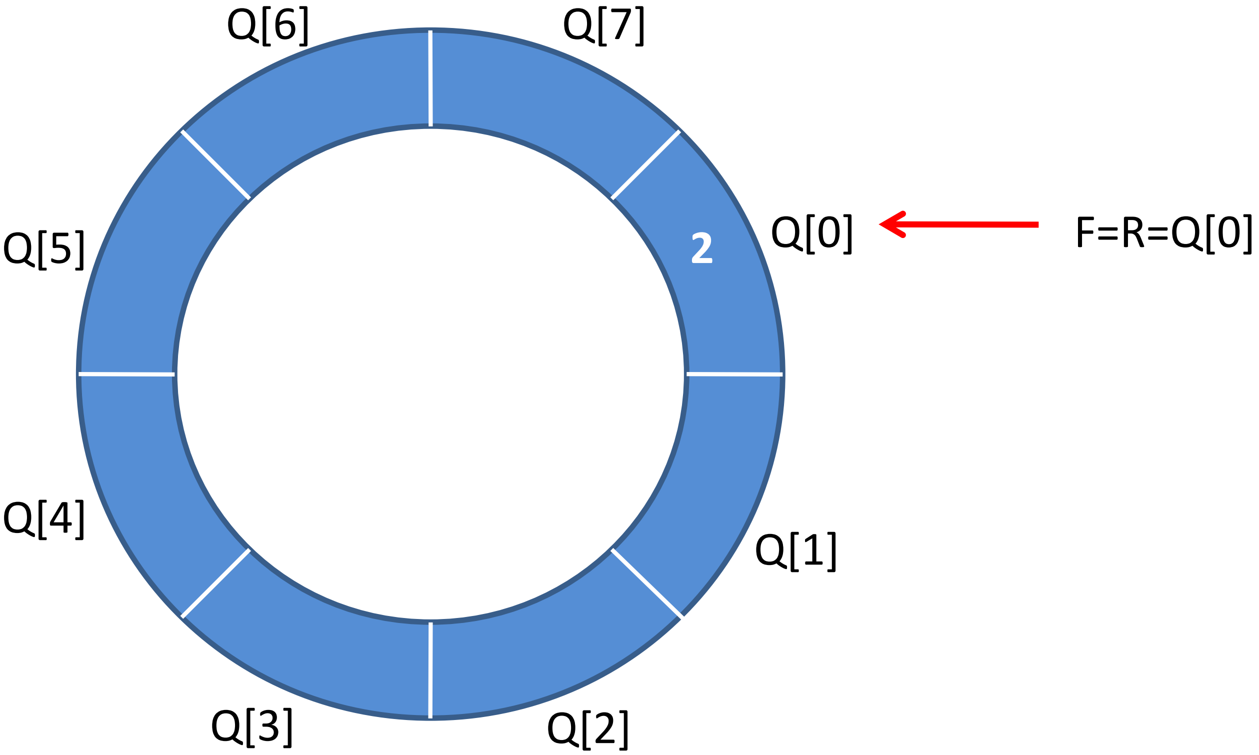
```
Step 1: IF FRONT=0 and REAR=MAX-1 OR REAR=FRONT-1 Then
                        Write “Overflow”
                        Goto Step 4
        [END OF IF]
Step 2: IF FRONT = REAR=-1 then
                        SET FRONT=REAR=0
        ELSE IF REAR=MAX-1 and FRONT!=0
                        SET REAR=0
        ELSE
                        SET REAR=REAR+1
        [END OF IF]
Step 3: SET QUEUE[REAR]=VAL
Step 4: EXIT
```

e.g.:



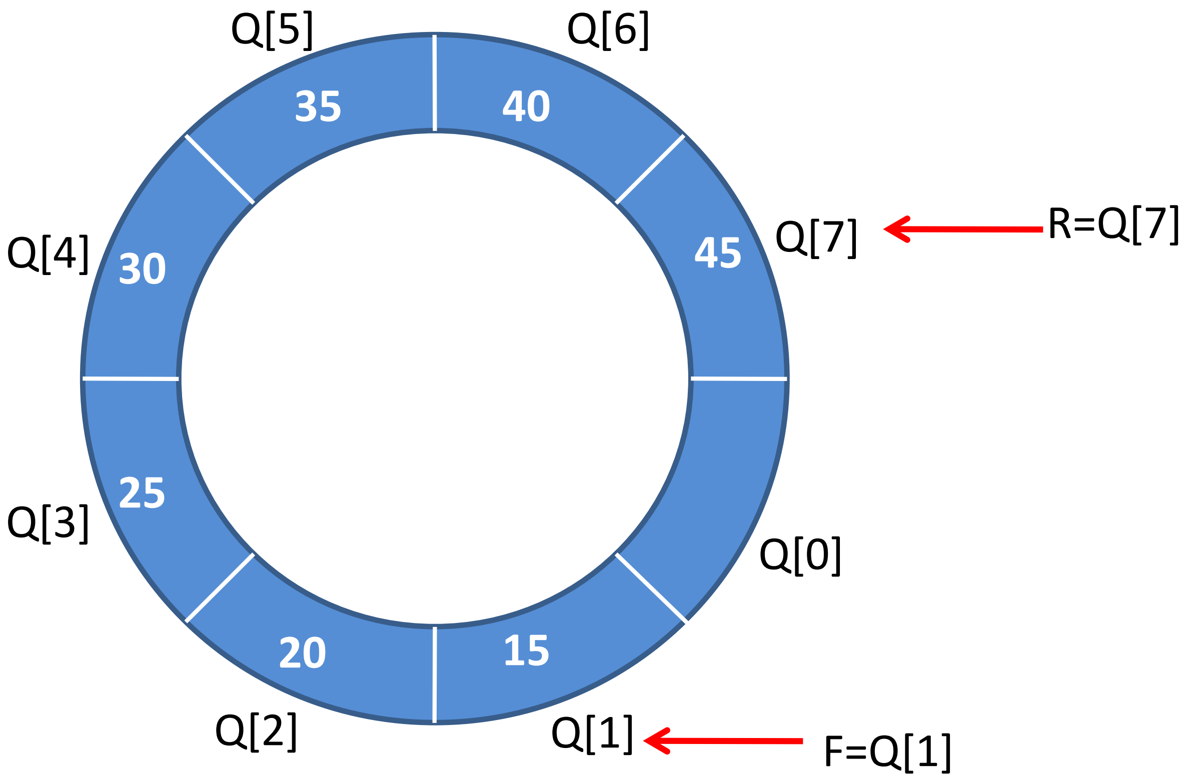
Queue is full(Overflow)

e.g.: (cond..)



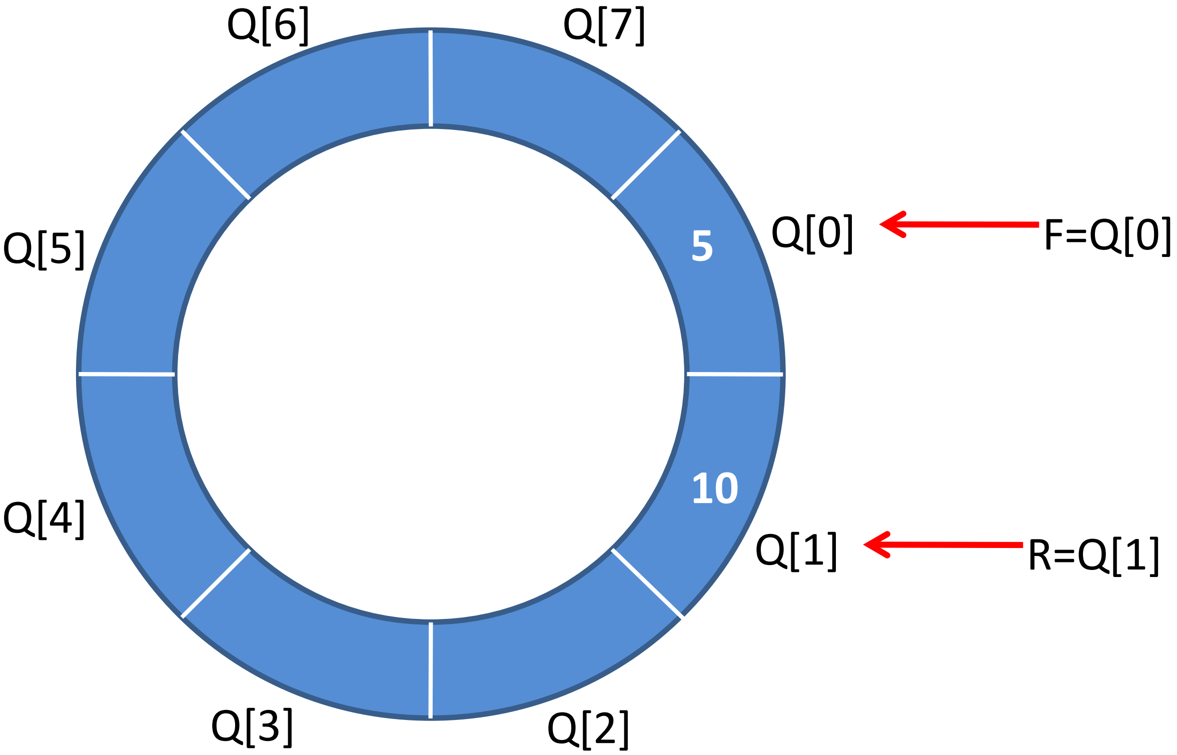
If F=R=-1 then F=R=0

e.g.: (cond..)



If REAR=MAX-1 and FRONT!=0

e.g.: (cond..)



Rear=Rear+1

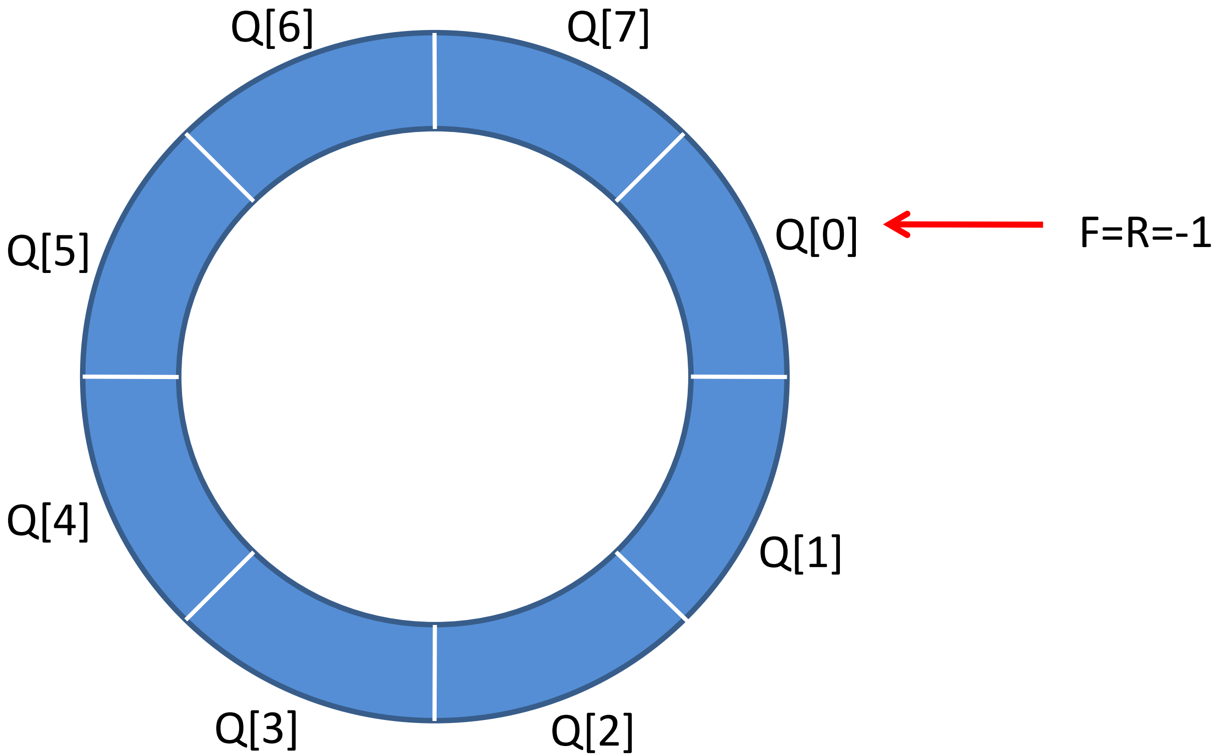
Operations on Circular Queue

Deletion:

Algorithm:

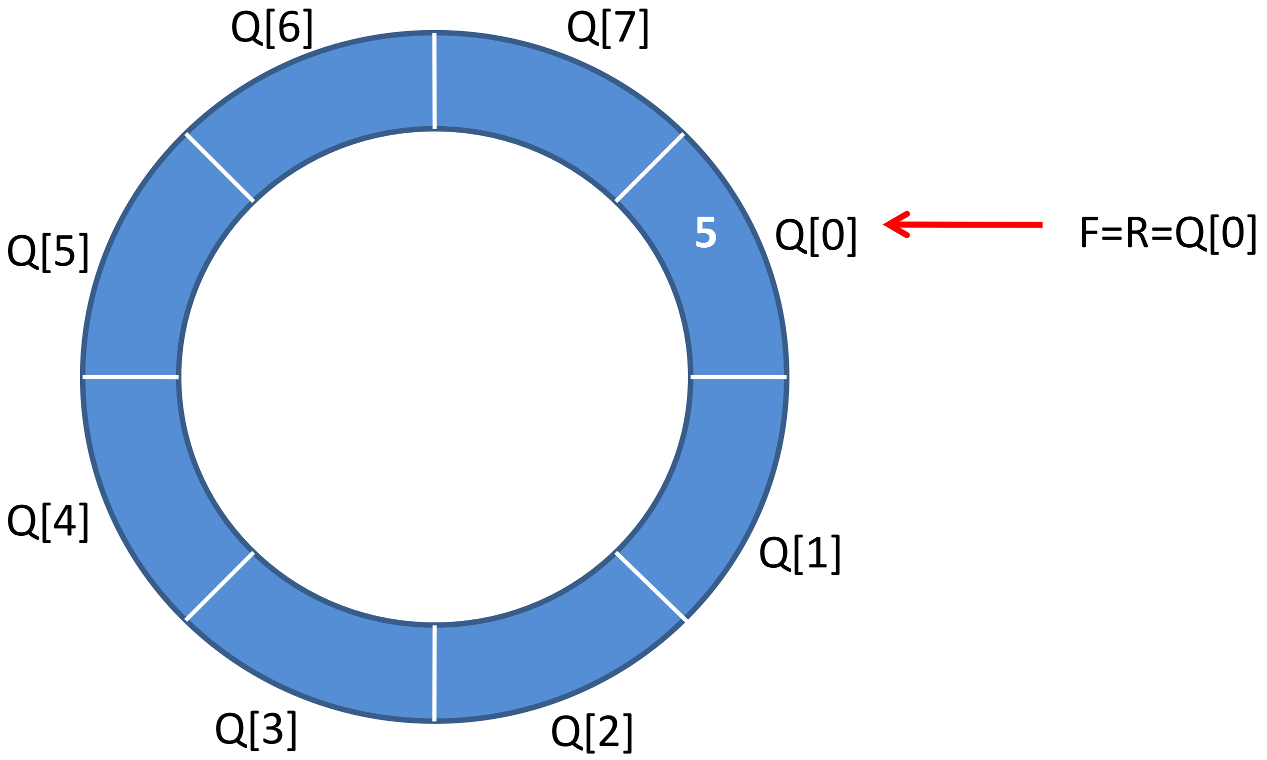
```
Step 1: If FRONT = -1 then
                Write ("Circular Queue Underflow“)
                   GOTO Step 4
Step 2: SET VAL=QUEUE[FRONT]
Step 3: If FRONT = REAR then
                SET FRONT=REAR=-1
          ELSE
                IF FRONT=MAX-1
                        SET FRONT=0
                ELSE
                        SET FRONT=FRONT+1
                [END OF IF]
          [END OF IF]
Step 4: EXIT
```

e.g.:



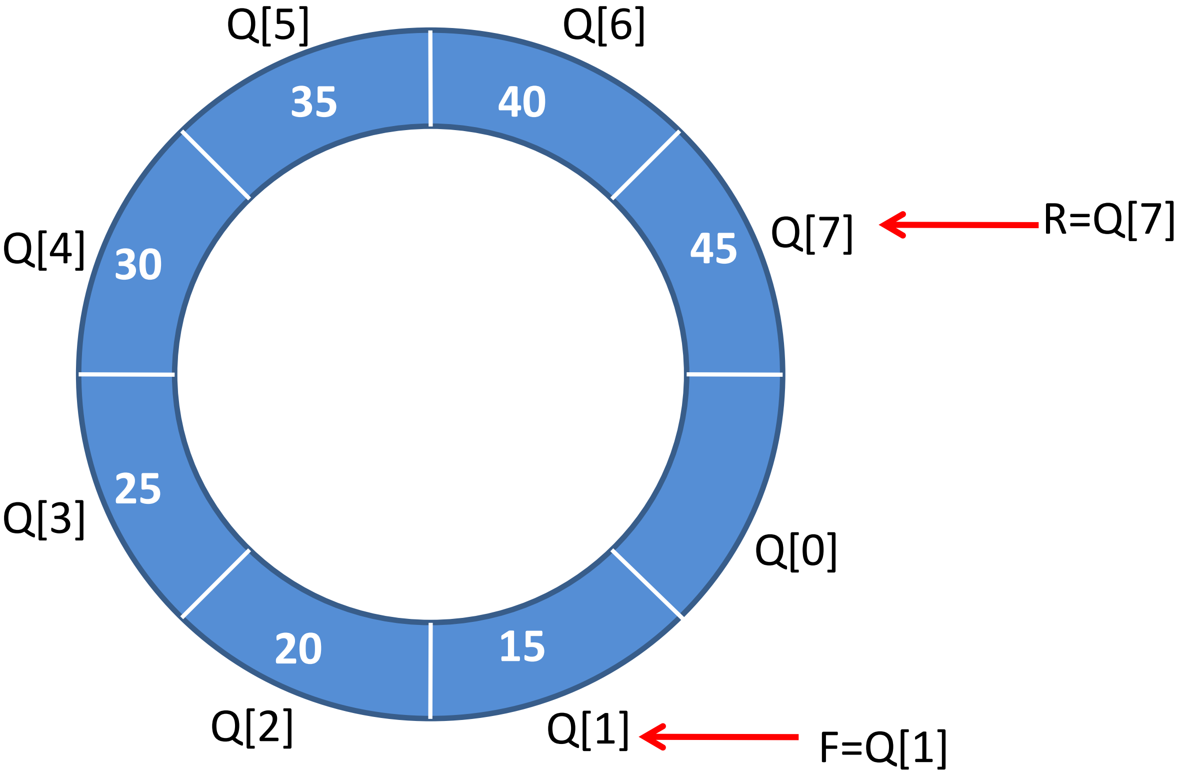
Queue is Empty(Underflow)

e.g.:



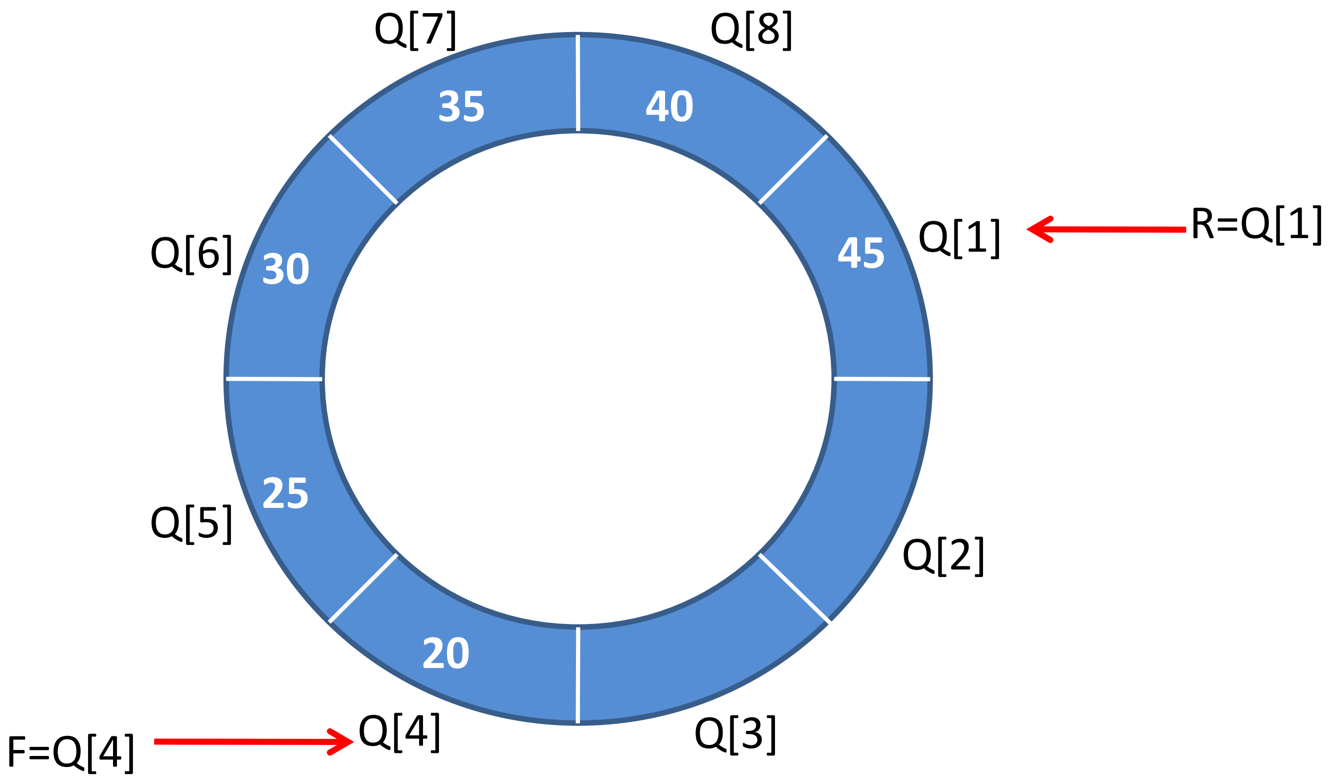
If F=R=0 then F=R=-1

e.g.: (cond..)



If Front=MAX-1 then Front=0

e.g.: (cond..)



Front=Front+1

Priority Queue

Priority Queue

- In priority queue, each element is assigned a priority.
- Priority of an element determines the order in which the elements will be processed.
- Rules:
  1. An element with higher priority will processed before an element with a lower priority.
  2. Two elements with the same priority are processed on a First Come First Serve basis.

Types of Priority Queue

1. Ascending Priority Queue

   In this type of priority queue, elements can be inserted into any order but only the smallest element can be removed.

2. Descending Priority Queue

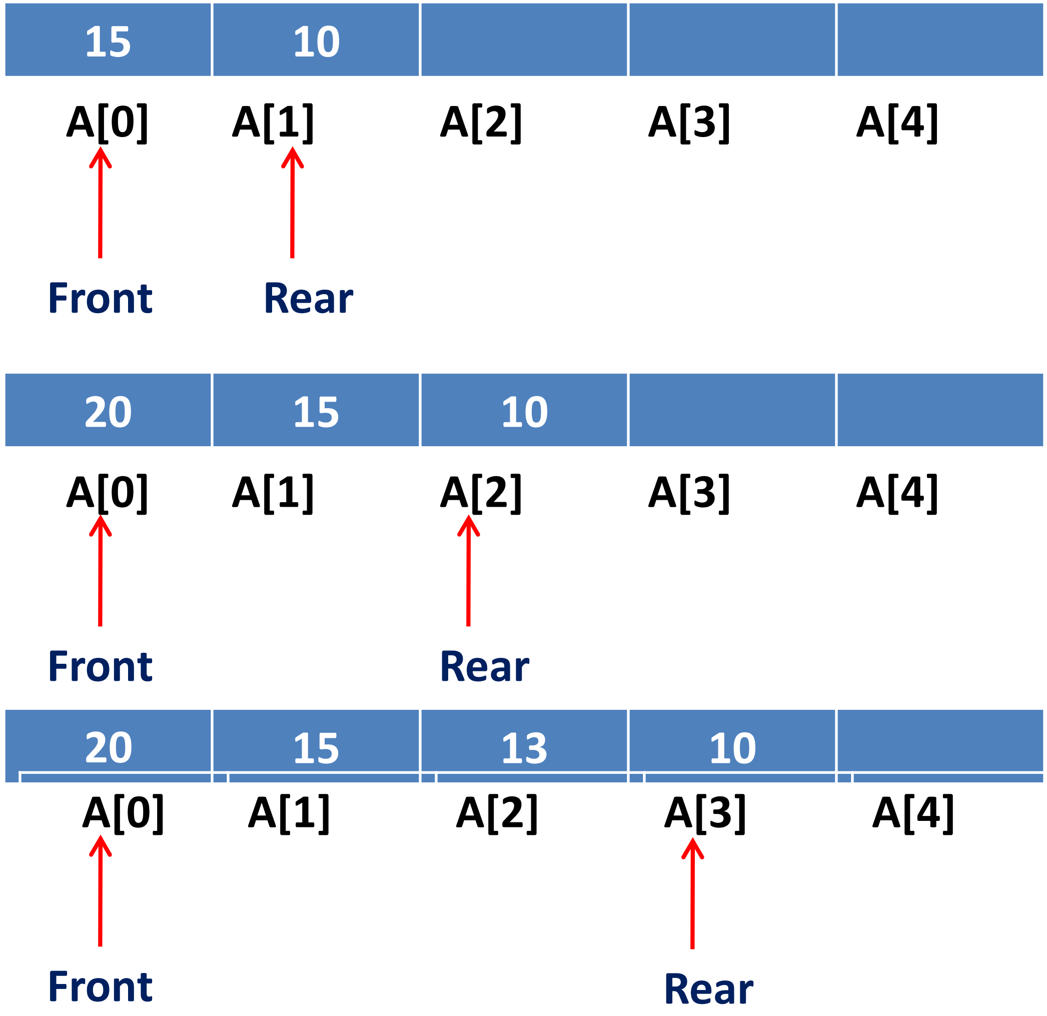
   In this type of priority queue, elements can be inserted into any order but only the largest element can be removed.

Array Representation of Priority Queue

Insertion Operation:

- While inserting elements in priority queue we will add it at the appropriate position depending on its priority
- It is inserted in such a way that the elements are always ordered either in Ascending or descending sequence

Array Representation of Priority Queue



15	10			
A[0]	A[1]	A[2]	A[3]	A[4]
Front	Rear			

20	15	10		
A[0]	A[1]	A[2]	A[3]	A[4]
Front		Rear		

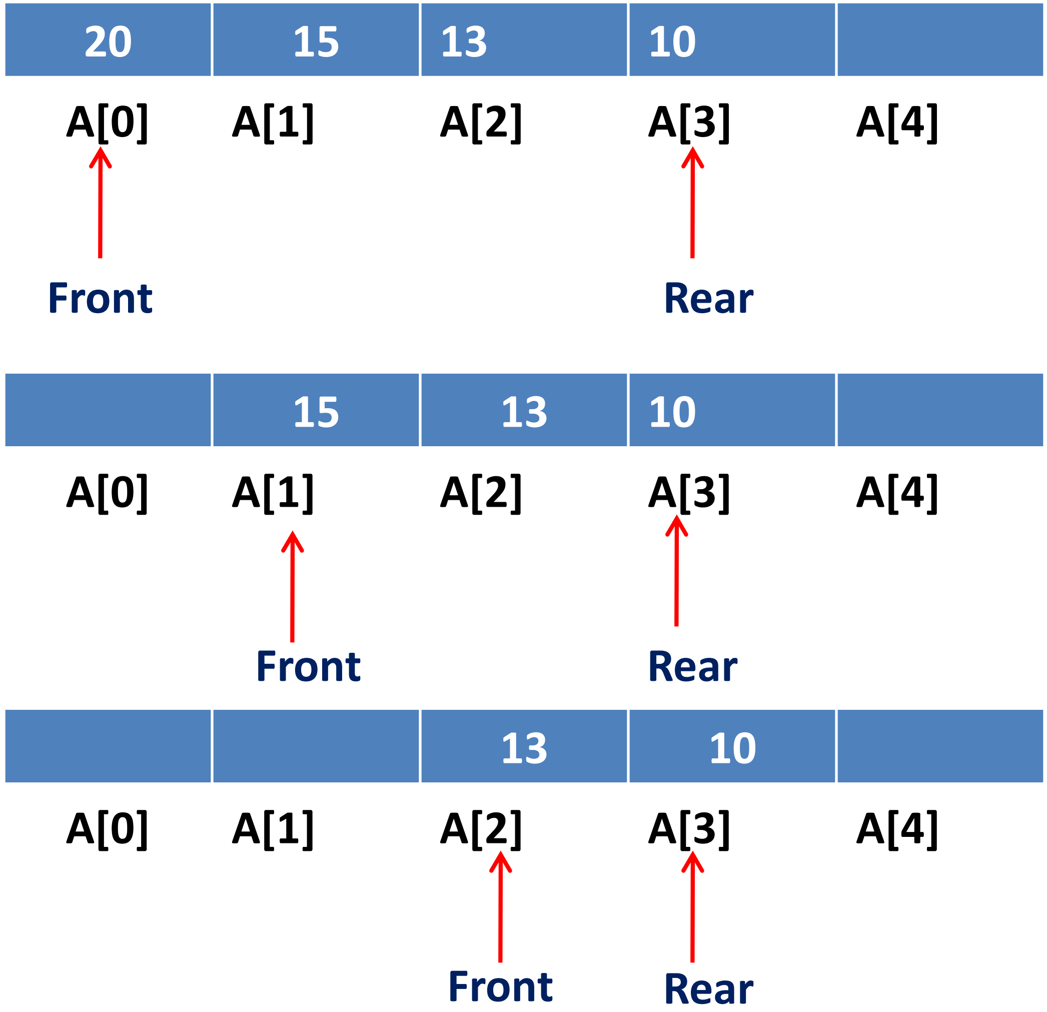
20	15	13	10	
A[0]	A[1]	A[2]	A[3]	A[4]
Front			Rear	

Array Representation of Priority Queue

Deletion Operation:

- While deletion, the element at the front is always deleted.

Array Representation of Priority Queue



20	15	13	10	
A[0]	A[1]	A[2]	A[3]	A[4]
Front			Rear	

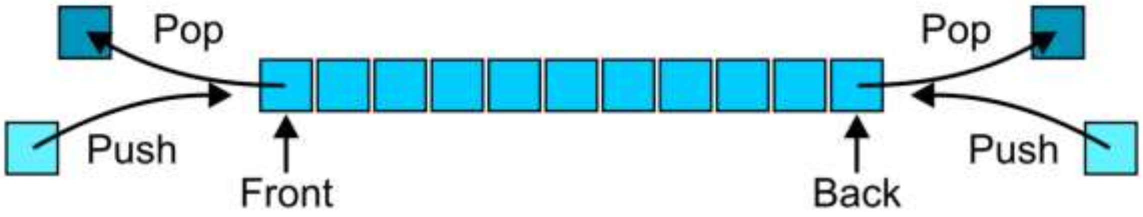
	15	13	10	
A[0]	A[1]	A[2]	A[3]	A[4]
	Front		Rear	

		13	10	
A[0]	A[1]	A[2]	A[3]	A[4]
		Front	Rear	

Double Ended Queue

What is deque ?

- ✓ A **double-ended queue** is an abstract data type that generalizes a queue, for which elements can be added to or removed from either the front or rear.
- ✓ It is also often called a **head-tail linked list**.



Types

Input-restricted deque

Deletion can be made from both ends , but

Insertion can be made at one end only.

Output-restricted deque

Insertion can be made at both ends , but

Deletion can be made from one end only.

Operations

pushRear() - Insert element at back

pushFront() - Insert element at front

popRear() - Remove last element

popFront() - Remove first element

isEmpty() – Checks whether the queue
is empty or not.

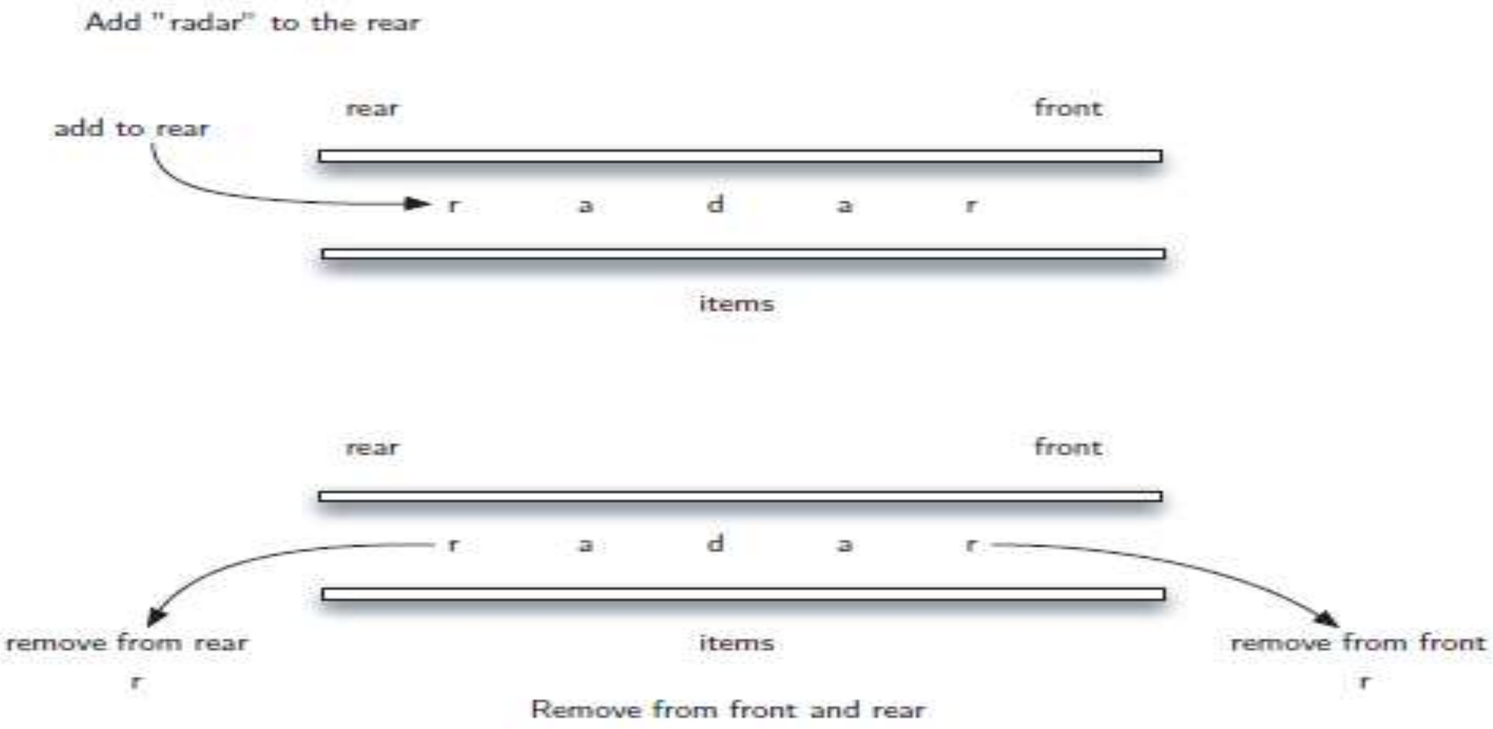
Example of deque Operation

Operation	deque Contents	Return Value
isEmpty()	[]	True
pushFront(‘a’)	[‘a’]	
pushFront(‘b’)	[‘b’ , ‘a’]	
pushRear(‘c’)	[‘b’ , ‘a’ , ‘c’]	
popFront()	[‘a’ , ‘c’]	‘b’
isEmpty()	[‘a’ , ‘c’]	False
popRear()	[‘a’]	‘c’

deque Applications

Palindrome Checker

Madam, Radar, Malayalam are some examples for palindrome



Thank You