Queue

Marwadi University

Department of CE

Unit No. 2
Linear Data Structures and their representation
Data Structure (01CE1301)

Unit-2 Linear Data Structures & their representation

In this part of unit 2 you will be able to understand the following concepts:

- Queue Concept
- Operations on Queue (insert, delete)
- Types of Queues -Simple Queue
- Circular Queue
- Double Ended Queue
- Priority Queues
- Applications of Queue.

Simple Queue

A ***queue*** is a linear data structure which can be implemented using an array or a linked list.

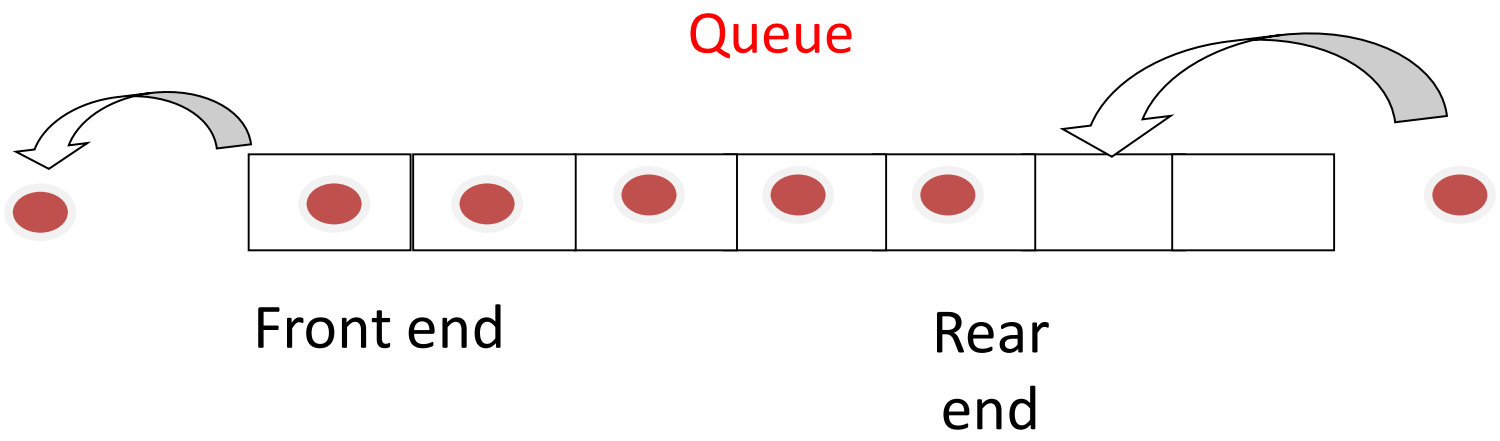
The elements in a queue are added at one end called the ***rear*** and removed from the other end called the ***front.***

Hence, queue is called a ***FIFO*** (First In First Out)

Queue

Ordered homogeneous group of items in which the items are added at one end (rear) and are removed from the other end (front).

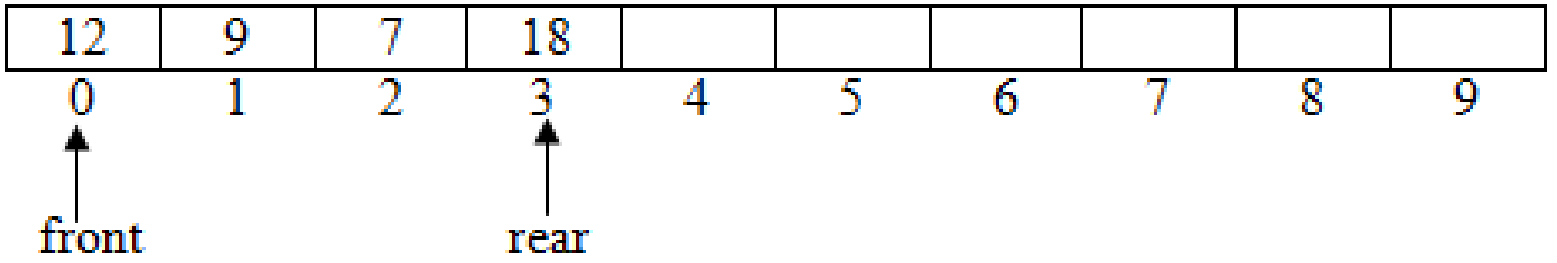
First In, First Out (FIFO)



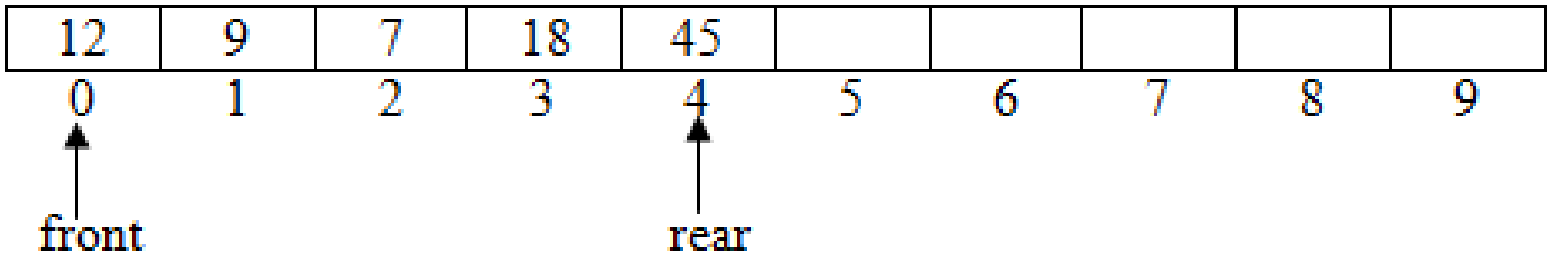
Queue Example:

The real-world example of a queue is the ticket queue outside a cinema hall, where the person who enters first in the queue gets the ticket first, and the last person enters in the queue gets the ticket at last. Similar approach is followed in the queue in data structure.

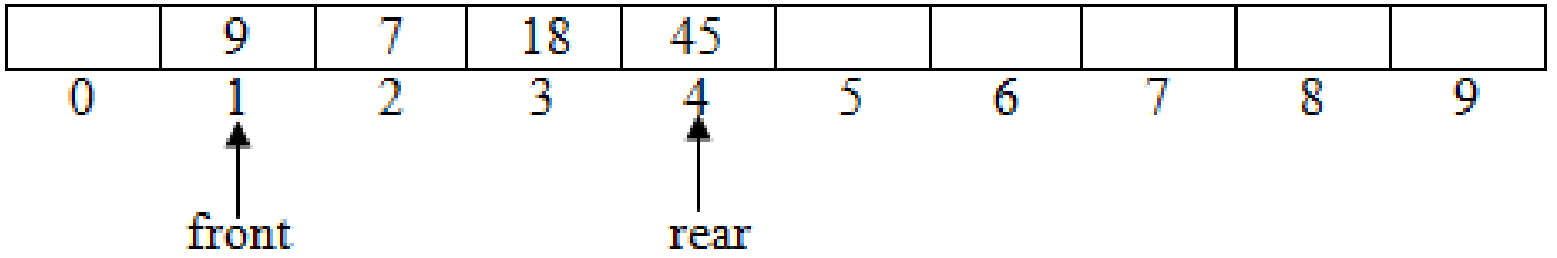
Array representation of Simple Queue



Add 45:



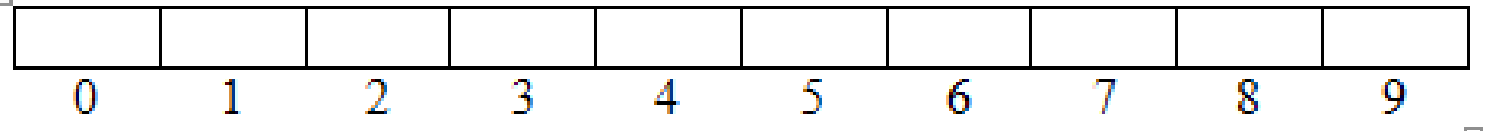
Delete:



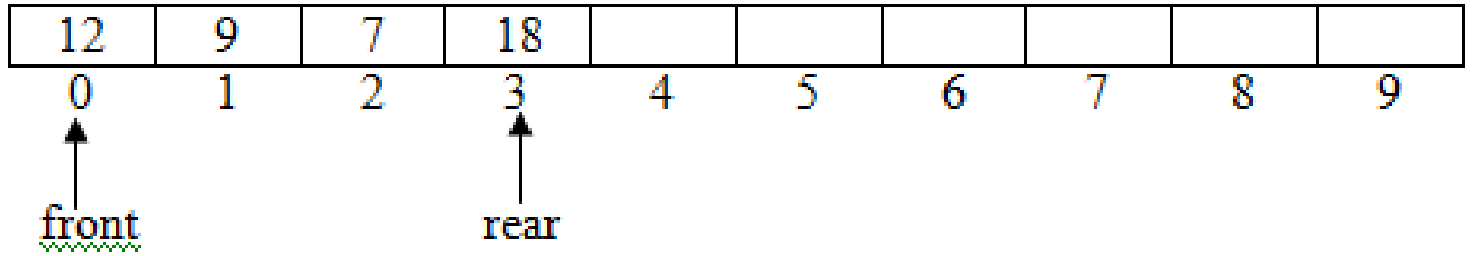
Operations on Simple Queue

Insert Operation

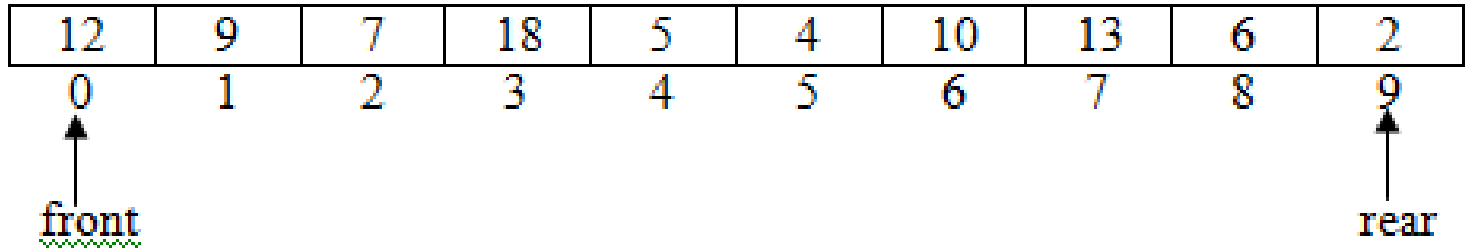
Initially Case: front=-1, rear=-1



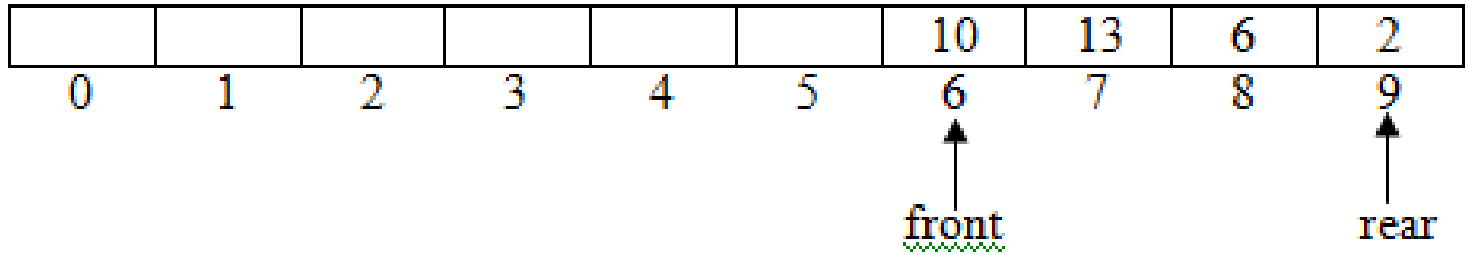
Normal Case:



Overflow Case:



Overflow Case:



Insertion in queue

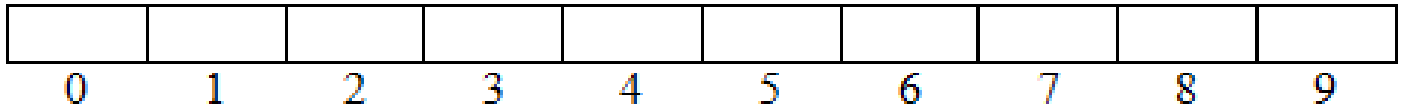
QINSERT (Q,F,R,N,Y) : Given F and R , pointers to the front and rear elements of queue , a queue Q consisting of N elements and y is element which is inserted by this procedure at rear of queue . Initially F and R are set to -1.

1. [checking for overflow]
   If (R= =N-1)
      then write (" overflow")
         return
2. [Increment rear pointer]
   R←R+1
3. [Insert element]
   Q[R] ← Y
4. [Is front pointer properly set]
   if F=-1
      Then F← 0
      Return

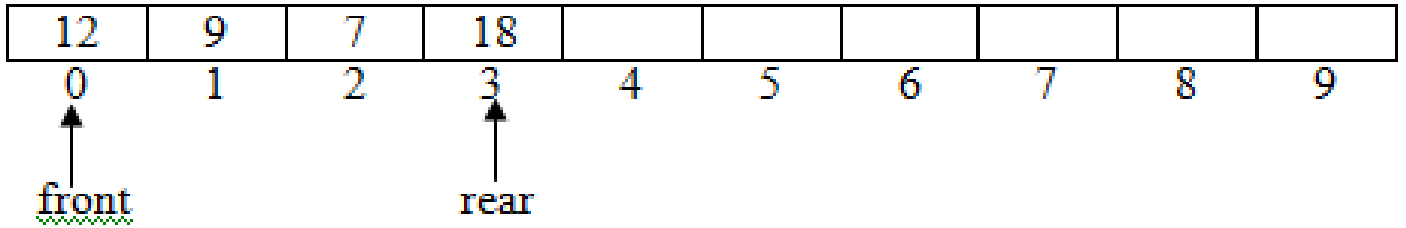
Operations on Queue

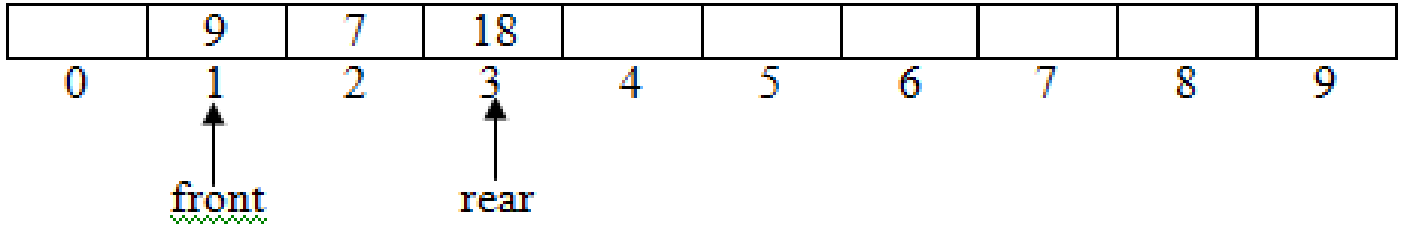
Delete

Underflow Case: front=-1, rear=-1

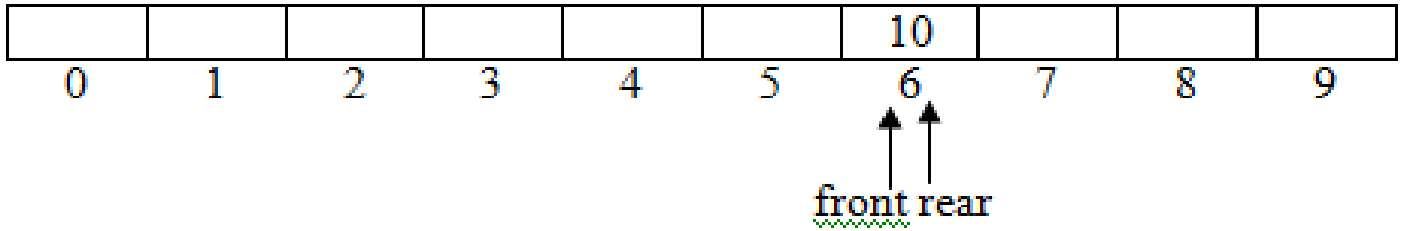


Normal Case:





Special Case:



front = -1, rear = -1

Deletion from queue

QDELETE (Q,F,R) : Given F and R , pointers to the front and rear elements of queue , This procedure delete element at front of the queue . Y is temporary variable .

1. [checking for underflow]
   If (F==-1)
   then write (" underflow")
   return
2. [Delete element]
   Y← Q[F]
3. [Queue empty]
   if F=R
   Then F← R ← -1
   Else F← F + 1 (increment front pointer)
4. [Return element]
   Return(Y)

Example on Simple Queue

- Ex 1. Following operations are performed on an empty queue. Give me front and rear after each operation.

1 Add A, B, C, D, E, F

2Delete two

alphabets 3 Add G

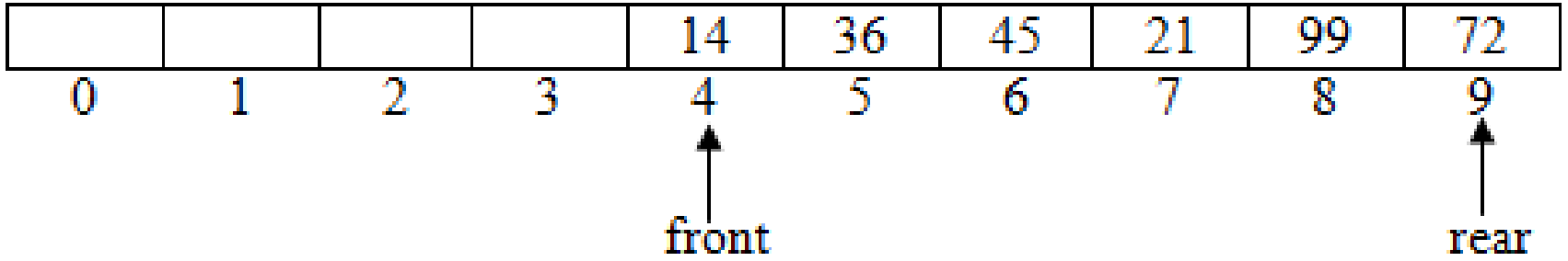
4 Add H

5Delete four

alphabets 6 Add I

Drawback of Simple Queue

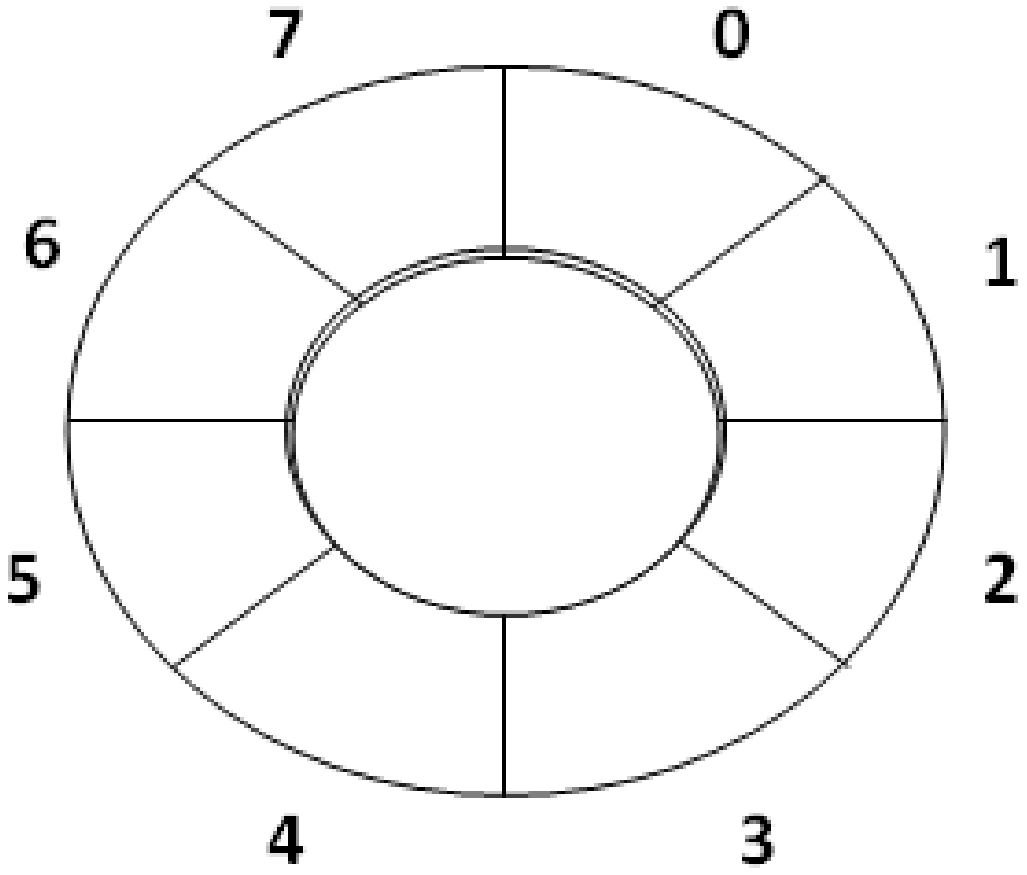
Waste of Memory Space



- front = 4, rear = 9
- Even though there is a space available, the overflow condition still exists and we can't insert an element.
- **Two solutions:**
  1) Shift all elements to the left so that we can insert element; But it is time consuming.
  2) Circular Queue

2. Circular Queue

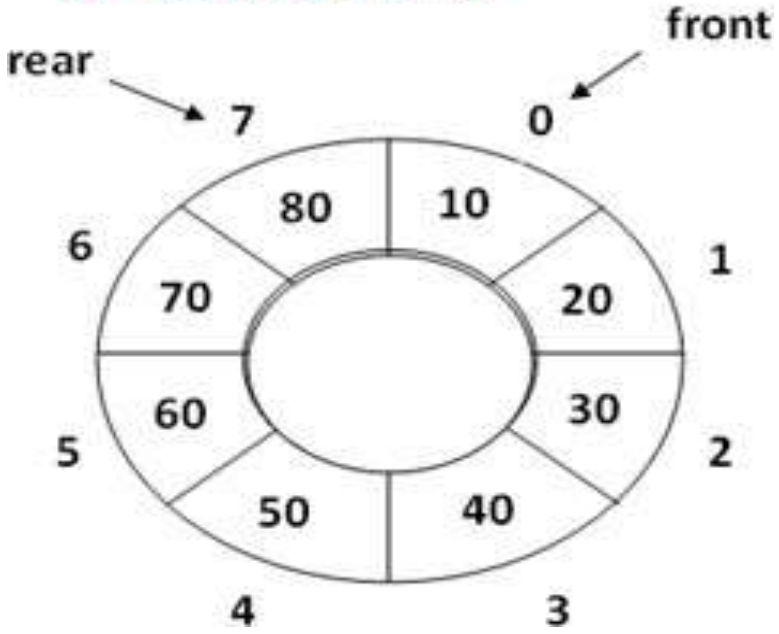
- The first index comes right after the last index



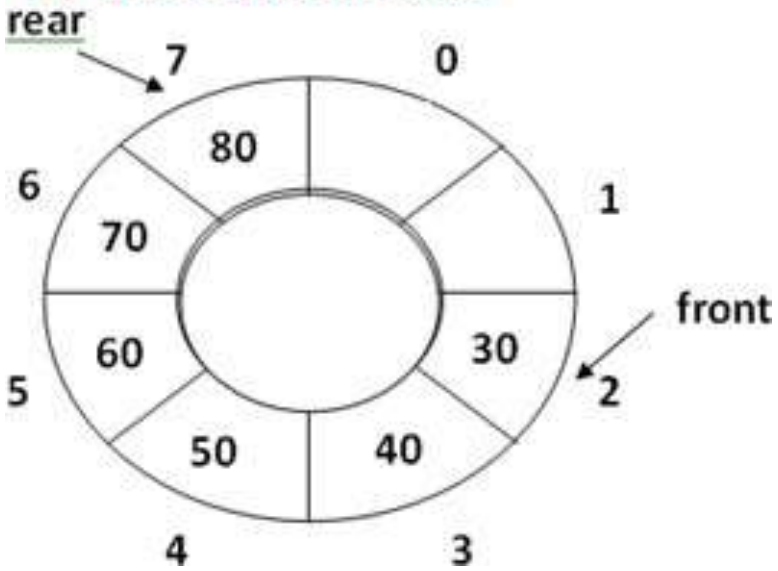
Operations on a Circular Queue

Insert in Circular Queue

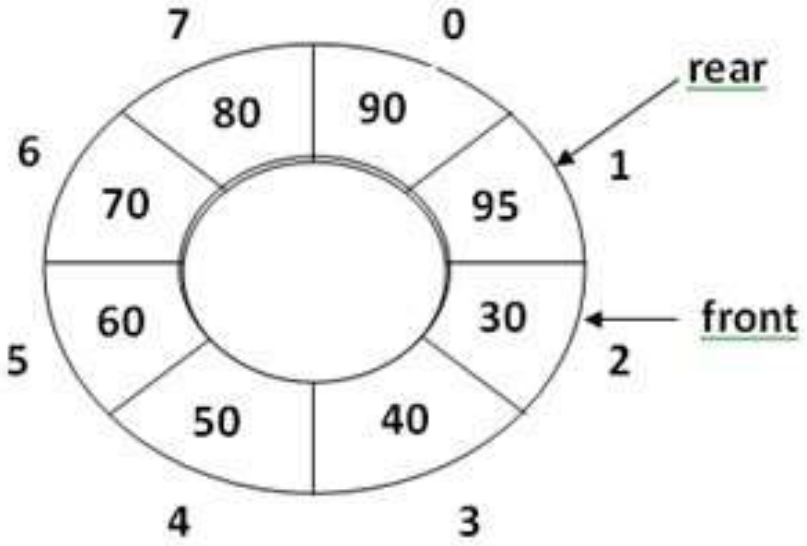
1) Insert 8 elements



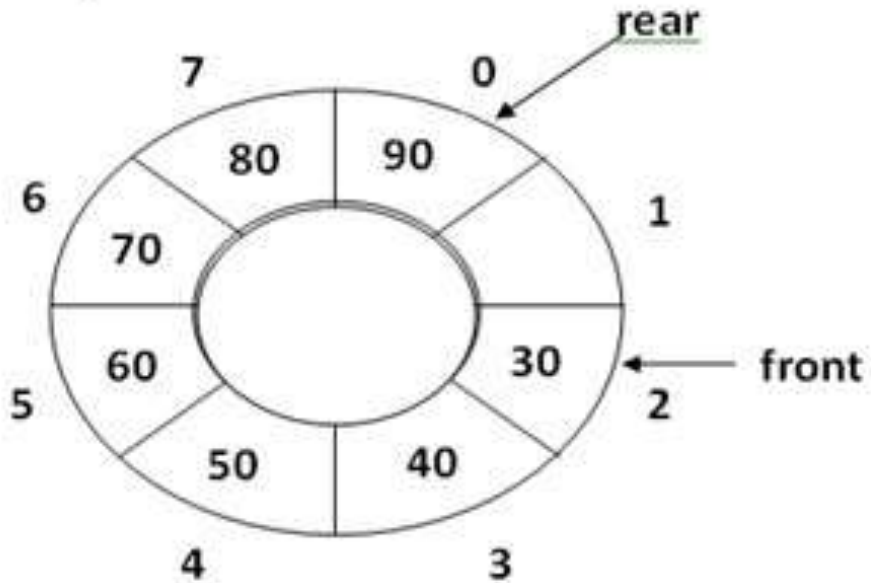
2) Delete two elements



4) Insert one element



3) Insert one element



Operations on a Circular Queue

Insert in CQueue:

CQinsert(Queue, front, rear, MAX, New_Value)

Step 1: [Overflow]
if (front = 0 and rear = MAX – 1) OR (rear = front – 1)
then
write "Overflow"
Exit
[End of If]

Step 2: [Reset Rear Pointer]
Set rear = (rear + 1) % MAX

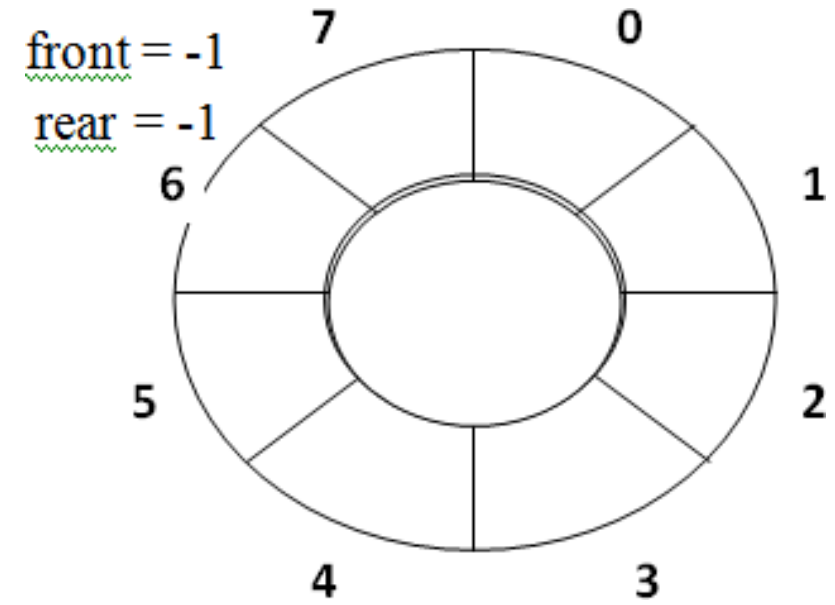
Step 3: [Insert an element]
Set Queue[rear] = New_Value

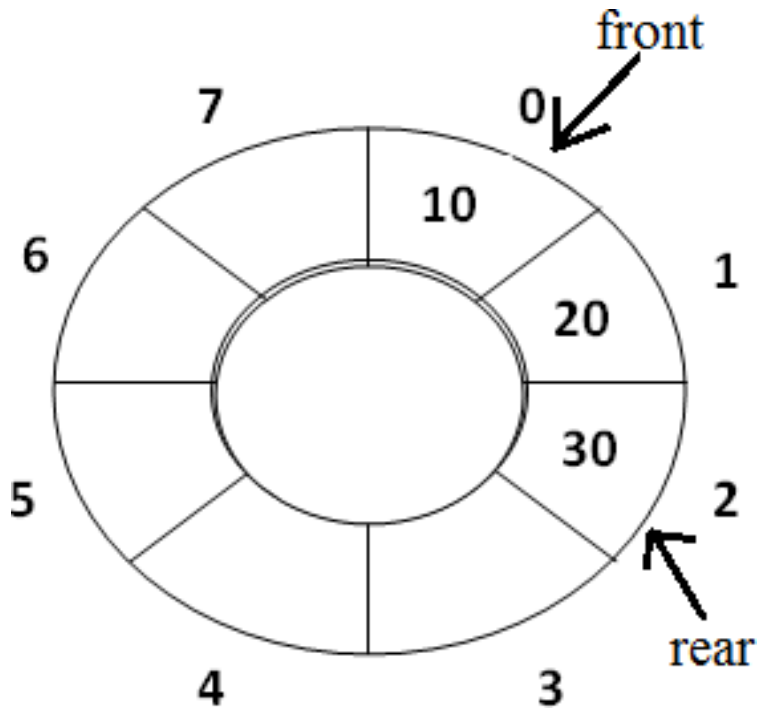
Step 4: [Queue is empty then set front pointer]
IF front = -1 then
set front = 0
[End of IF]

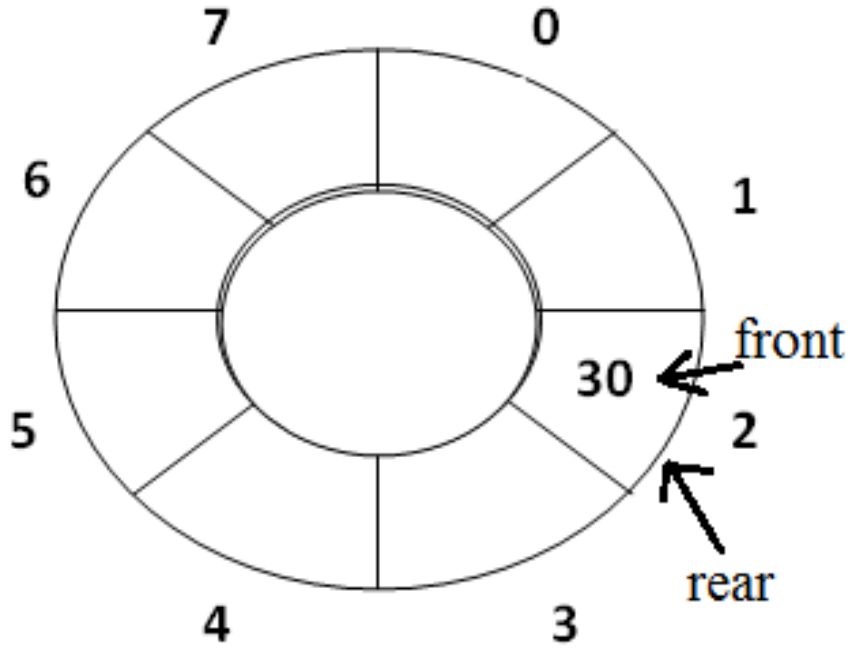
Step 5: Exit

Delete from Circular Queue

Operations on a Circular Queue







Operations on a Circular Queue

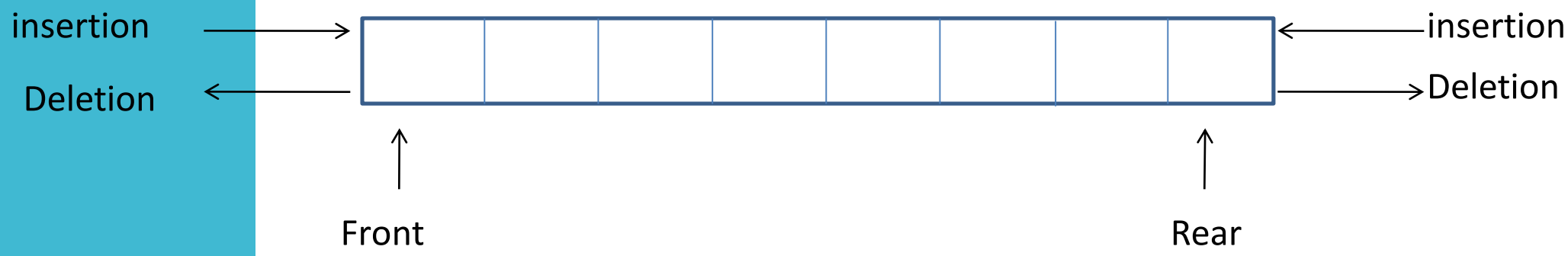
Delete from CQueue:

CQdelete (Queue, front, rear)

Step 1: [Underflow]
if front = - 1 then
write "Underflow"
Exit [End of If]
Step 2: [Delete an element]
Set value = Queue[front]
Step 3: [increment front pointer]
[Only one element in Queue]
IF front = rear then
set front = rear = -1
Else
set front = (front + 1) % MAX [End of IF]
Step 4: return (value)

Deque

Double-ended queue - Elements can be added to or removed from the front or back.



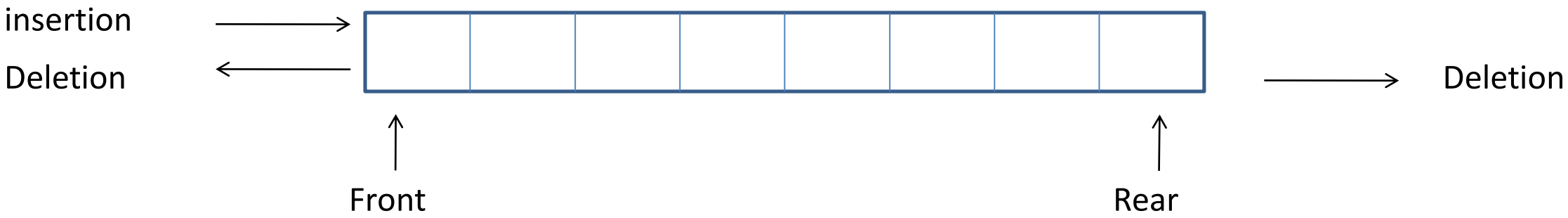
Types:

1. Output-restricted deque
2. Input-restricted deque

Deque

Output-restricted deque - Insertion can be made at both ends, but output can be made from one end only.

Input-restricted deque - Deletion can be made from both ends, but input can only be made at one end.



Double Ended Queue

Algorithm for Insertion at front end

Step-1: If (front=0 AND (rear=MAX-1 OR front=rear+1))
"Queue Overflow"
Step-2: If (front=-1)
set front=rear=0
else if (front=0)
set front=MAX-1
else
front=front-1
Step-3: deque[front] = value

Double Ended Queue

Algorithm for Insertion at rear end

Step-1: If (front=0 AND (rear=MAX-1 OR front=rear+1))
"Queue Overflow"
Step-2: If (front=-1)
set front=rear=0
else if (rear=MAX-1)
set rear=0
else
rear=rear+1
Step-3: enque[rear] = value

Double Ended Queue

Algorithm for Deletion at rear end

Step-1: If (front=-1)
“Queue Underflow”
Step-2: If (front=rear)
set front=rear=-1
else if (rear=0)
set rear=MAX-1
else
rear=rear-1

Double Ended Queue

Algorithm for Deletion at front end

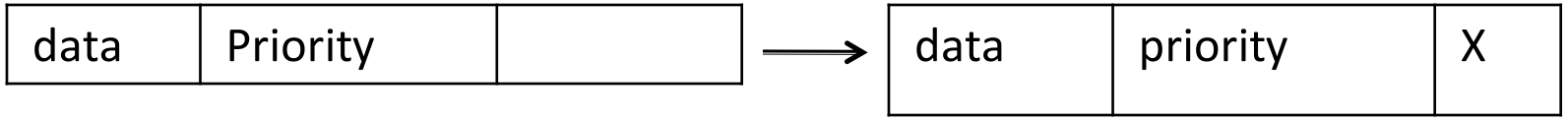
Step-1: If (front=-1)
“Queue Underflow”
Step-2: If (front=rear)
set front=rear=-1
else if (front=MAX-1)
set front=0
else
front=front+1

Priority Queue

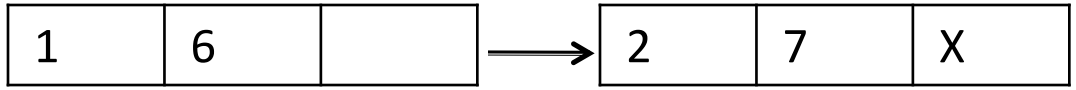
- A modified queue in which elements are inserted arbitrarily with an associated priority. On deletion, element with the highest priority is removed from the queue
- Order of returned elements is not always FIFO.

- If there are two elements with same priority then process them as per FIFO.

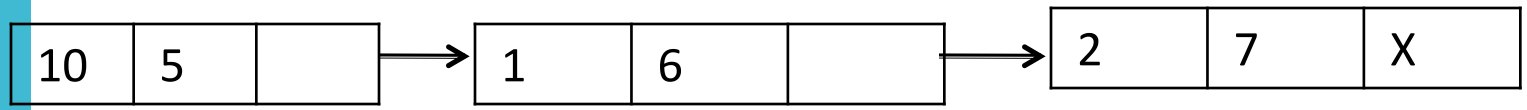
- Examples
  - Processes scheduled by CPU



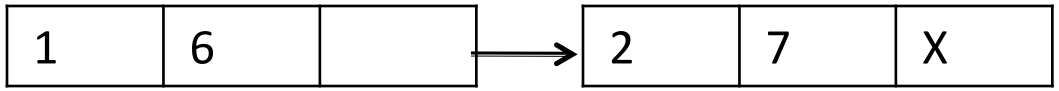
For example



- Insert (data = 10 , priority = 5)



- Delete



Application of Queue

- Queue used as waiting lists for a single shared resource like printer, disk, CPU, etc.
- It used as buffer on MP3 player, iPod playlist.
- It used in Operating System for interrupts handling
.

THANK YOU