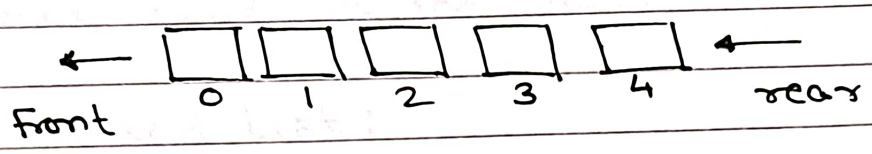
Queue Data Structure:

A Queue is a linear data structure that stores the dements sequentially. It inserts elements from one end & deletes elements from another end. Thus, we can say FIFO approach (first in First out)

Insertion happens at rear end whereas deletion happens at front end of the queue.



Queue operations:

1.) Enqueue()
2.) Dequeue()
3.) Peek()
4). isfull()
5.) isEmpty()

Queue may be represented ~~in~~ using lists (one o
or ~~a~~ linear arrays. Each queue will be maintaine
by a linear array Queue & two pointer variables:
Front, containing location of front element of
queue & rear, containing location of rear
element of queue.

Condition Front = Null indicates empty
Queue

This inserts an Element item into Queue

EnQueue (Q, N, Front, Rear, Item)

1. [Queue already Filled?]
   if ~~for~~ front = 1 & Rear = N or if front = Rear + 1
   then:
   write: overflow, & Return
2. [Find new value of Rear]
   if front = Null then [Queue initially empty]
   set front = 1 & Rear = 1
   else if Rear = N then
   set Rear = 1
   else
   set Rear = Rear + 1

3. Set Queue [Rear] = item
4. Return

Deleting an element from Queue

1.) [Queue already empty]
   if front = Null then write: Underflow &
   return
2.) Set item = Queue [front]
3. [Find New value of front]:
   if front = Rear, then [Queue has one elem
   set front = Null; Rear = Null
   else if front = N then
   set front = 1
   else
   set front = front + 1

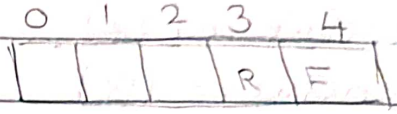
4. Return

Date : / /
Page No.

Algorithm to insert an element in a circular Queue:

Queue is full

1.) If (Rear+1) % Max = Front
    write overflow
    ~~else~~ return



Rear = 3 max = 5
(Rear + 1) % max = front
check for other values

Queue is empty

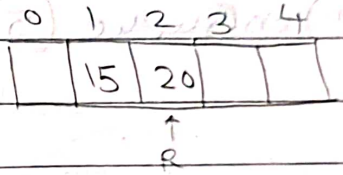
2.) If front = -1 & Rear = -1
    set front = Rear = 0

rear is at end

elif rear = max - 1 & front != 0
    set rear = 0

general condition

else
    set rear = (rear + 1) % max



3.) set Queue[rear] = val

Algorithm to delete an element from circular Q

Queue is empty

1.) if front = -1
    write (underflow)
    return

return value

2) set val = Queue[front]

only one element in Queue

3.) If front = rear
    set front = Rear = -1

Queue element deleted at the last

else if front = max - 1
    set front = 0

general condition

else
    front = front + 1

4.) Exit

1.) enQueue (14)
2.) enQueue (22)
3.) enQueue (13)
4) enQueue (6)
5) deQueue ()
6.) deQueue()
7.) enQueue (9)
8.) enQueue (20)
9.) enQueue (5)