QUEUES

Def:- It is a linear list in which all limitations are made at one end of the list "rear" of queue.

All deletions are made at other end "Front" of queue.

⟹ Insertion Operation = Enqueuing a Queue

⟹ Deletion Operation = Dequeuing a Queue

```
        Front   rear
          ↓      ↓
        _____
        a  b  c
        ‾‾‾‾‾‾‾‾‾‾‾
```

Drawbacks of Linear Queues-

when you delete some data from full queue, then you are not able to insert data at that place again.

* This is a gross limitation of a linear queue since Queue-Full condition does not check whether Q is physically Full.

```java
import java.util.*;

class que
{
    int array[] = new int [5];
    int    rear, front;
    int   size;

    que()
    {
        rear = -1;      front = -1;
        size = array.length;
    }
    Scanner  in = new  Scanner(System.in);
    void  qAdd()
    {    if (rear == size-1)
        {
            System.out.println("Out of Memory");
            return;
        }
        else
        {
            rear++;
            S.o.p(" enter value");
            array[rear] = in.nextInt();
        }return;
```

```java
void       qvDel()
{
    if (front == rear)
    {
        S.o.p(" Nothing found ");
        return;
    }
    else
    {
        front ++;
        sop(" the value is:" + array(front));
    }}}

class Apque
{   public static void main( Strings args[])
    {   qvue    qv = new    qvue();
        qv. qvDel();
        for (int  i=0; i<qv.array.length; i++)
        qv.qvAdd();

        for (int i=0; i<qv.array.length; i++)
        qv.qvdel();
    }
}
```
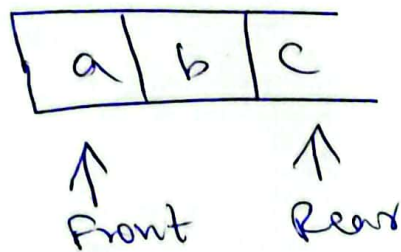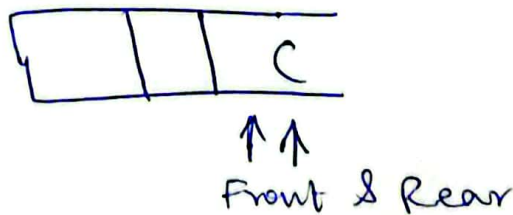
Circular Queue:–

They serve to rectify the limitation of linear queues.

Front and rear variables which displayed a linear movement over a queue, display a circular movement over the queue data structure.
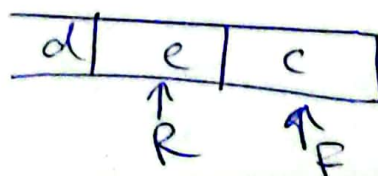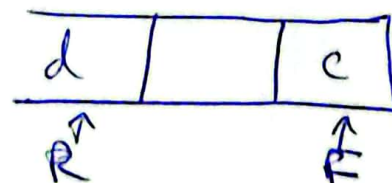
(a) Initial circular queue

a	b	c	

↑ Front   ↑ Rear

(b) Circular queue after two deletions

		c	

↑↑ Front & Rear

(C) Circular Queue after insertion of d, e

insert 'd'

d		c	

R↑   ↑F

d	e	c	

↑R   ↑F