Ordinary Queue:

0	1	2	3	4	5
10	20	30	40	50	60

F=0 R = 0 ++R : R = 1 : R = 2 : R=3: R = 4 : R=5 (Full: R = size-1)

0	1	2	3	4	5

F=0 R = 0 ++R : R = 1 : R = 2 : R=3: R = 4 : R=5 (Full: R = size-1)

F = 1 F++ : F=2 : F = 3 …… F = 6 (if F>R) **F = 0 R=-1**

0	1	2	3	4	5
	30	40	50	60	

F=0 ; R = 5

Delete 3 items: 10 20 30 F = 3 R = 5

Double Ended Queue: DQueue

InsertFront()

InsertRear()

DeleteFront()

DeleteRear()

InsertFront()

Case 1:

0	1	2	3	4	5

F = 0

R = -1 Q.items[++Q.R] = item

Case 2:

0	1	2	3	4	5
	11	10	20	30	

F = 2

R = 4 if F>0 Q.items[++Q.F] = item --F F = 1 : F=0

Case 3:

0	1	2	3	4	5
11	12	10	20	30	

F = 0

R = 4 **Insertion not possible**

Delete Rear:

0	1	2	3	4	5
		10	20	30	

F = 2

R = 4 delte rare and decrement R (30 is deleted and R = 3) R =1 ;

0	1	2	3	4	5
		10	20		

F = 2

R =   3

```
Void InsertFront(QUE *Q)

{

    //Check for Full

   // Read item

  If(Q->f>Q->r)// F=0 and r=-1

  Q->items[++Q->r] = item;

Else

  If(Q->f >0)

      Q->items[--Q->f] = item;

Else

 // Not possible

}

Void DeleteRear((QUE *Q)

{

    //Check for empty

  Pf(deleted: Q->items[Q->r--])

}
```

Circular Queue :

0	1	2	3	4	5
11	12	30	40	50	60

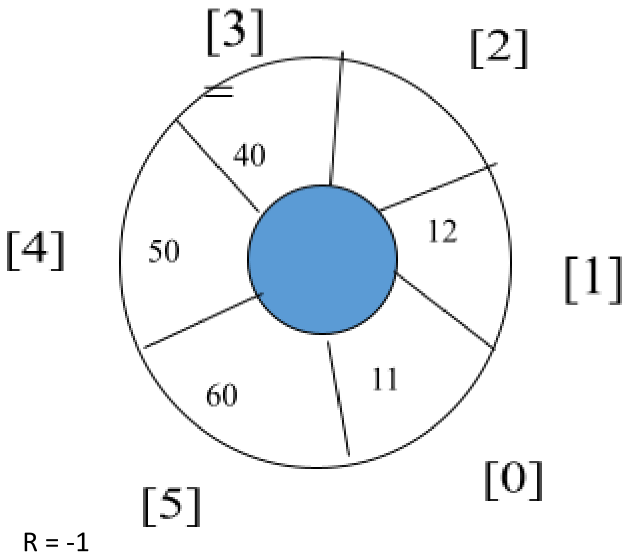
F=0 F = (F+1)%size (0+1)%6 = 1 : (1+1)%6 = 2

R = -1 R = (R+1) % Size : 0%6 = 0 (0+1)%6 = 1 : 2%6 = 2….. (4+1)%6 : 6%6 = 0 : (0+1)%6 = 1

0	1	2	3	4	5
	12	13			

F = 5: F = (F+1)%size : 4 %6 =4 : (4+1)%6 =5 : (5+1)%6 =0 : 1%6 = 1

R = R = (R+1) % Size = 6%6 =0 : (0+1)%6 = 1 : 2%6 = 2



F = 0 R = -1

F = 0 R = 5 Delete 3 items : 10 20 30 F = 3

F = 0 R = 1 Insert 2 items R = 1 F = 3

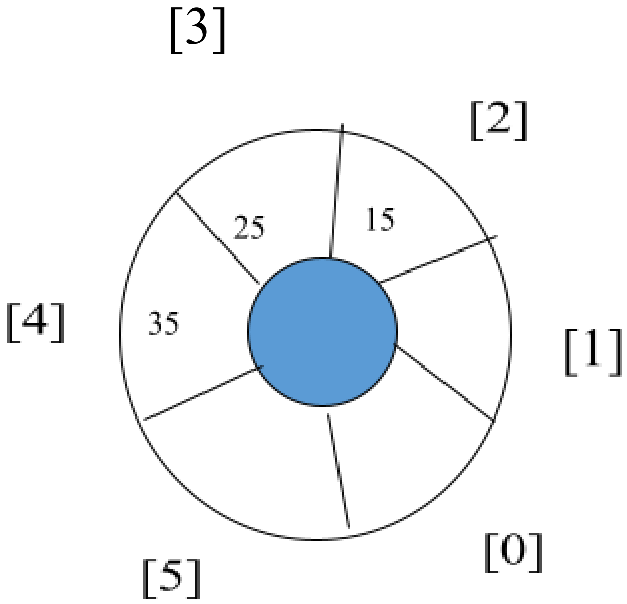
F = 0 R = 3

F = 0 R = 4

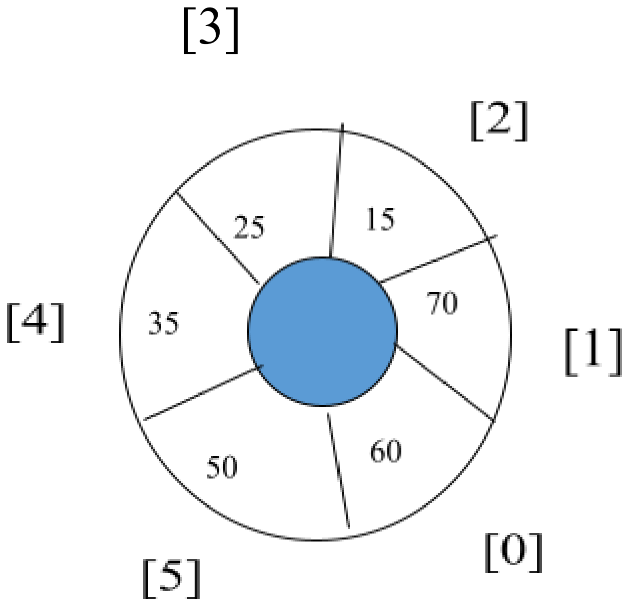
F = 0 R = 5

A circular queue of size 6 has three elements 15,25,35 which are inserted in the same sequence. Front value is 2 and Rear is 4. Show the content of the queue along with front and rear values with circular diagrammatic representation for the following sequence of operations. i)Insert 50 , 60 insert 70 ii) delete two elements

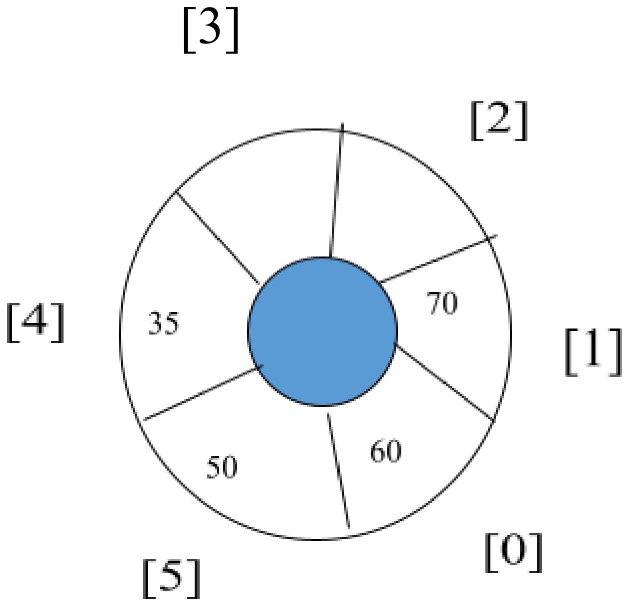
iii) Insert 100 & 200 V)Insert 300



F=2 R=4

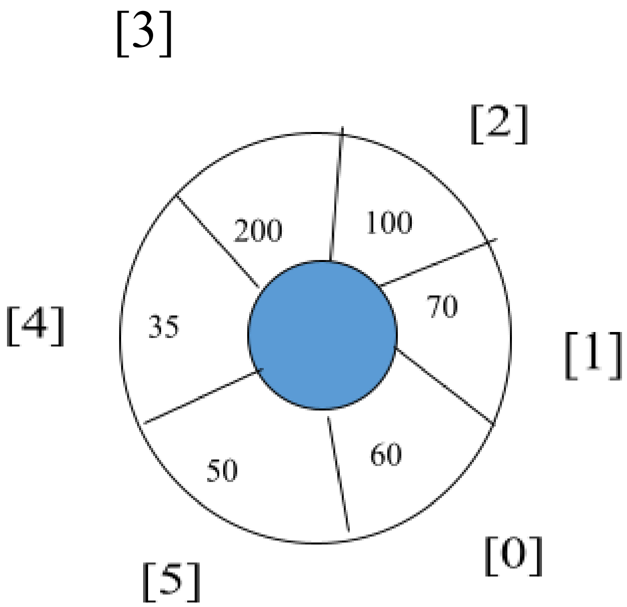


F=2 R=4 : F = 2 R =1



F = 2 R =1

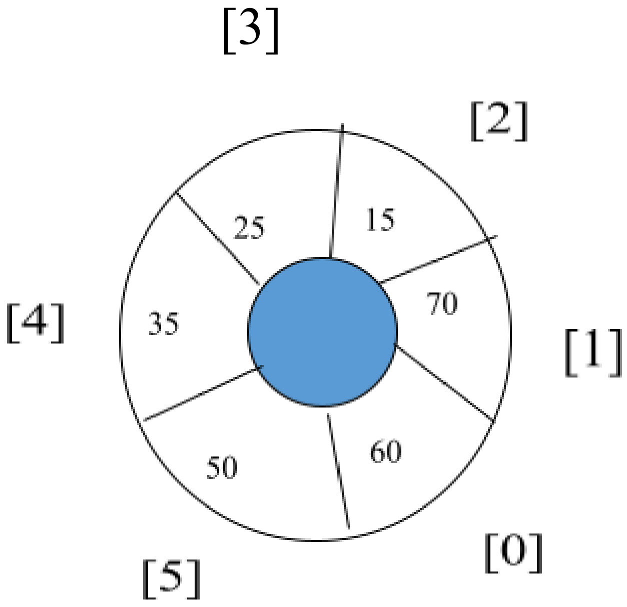
Delete 2 elements: 15 25 F =4 R =1



F =4 R =1

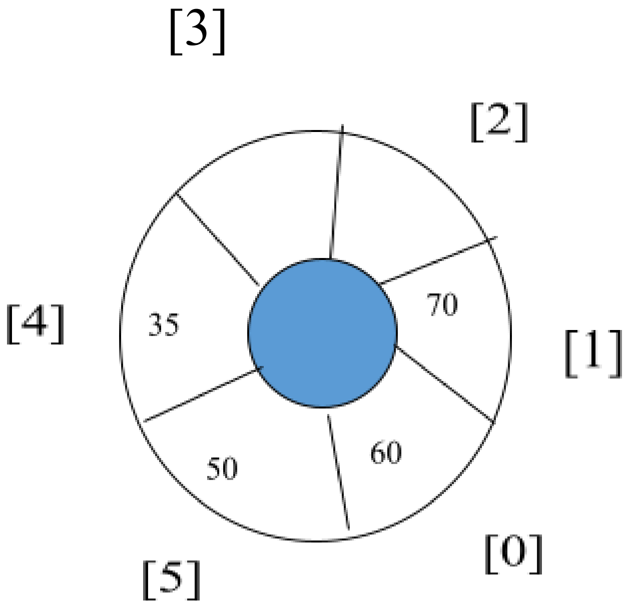
Insert 100 and 200 F = 4 R = 3

Insert 300



F=2   R=0

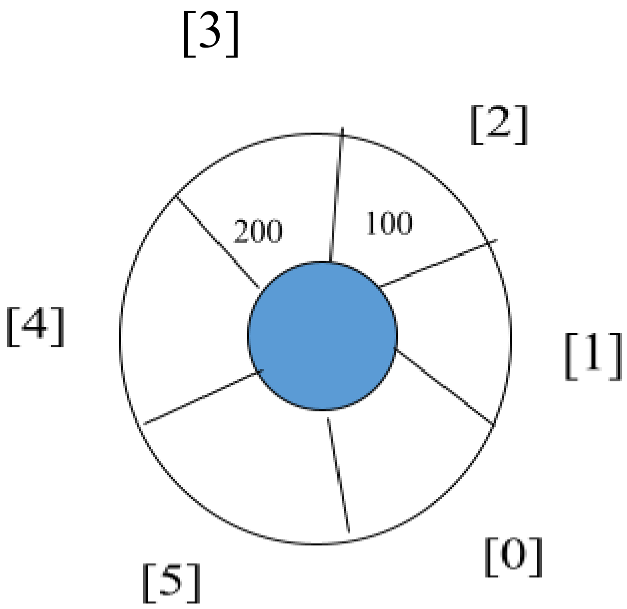
F= 2 R = 1



F=2   R=0

F= 4   R = 1

Deleted: 15   25



F=2 R=0

F= 2 R = 3

Delete 4 elements: 35 50 60 70

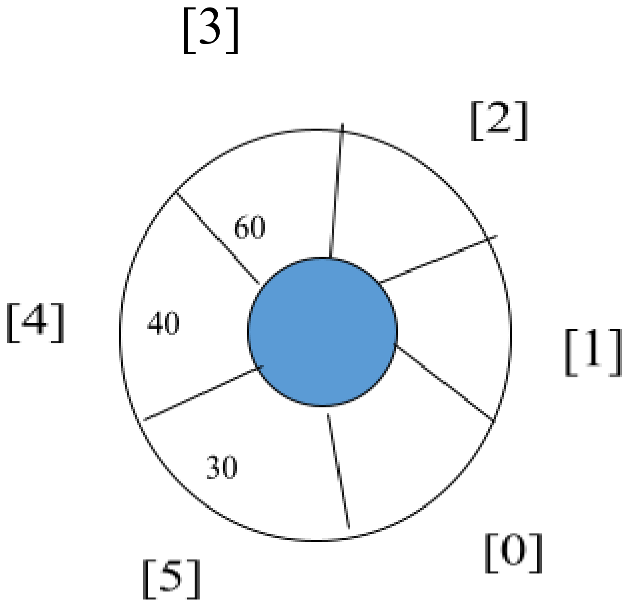
With circular diagrammatic representation show the status of a circular queue whose size is 6 for the sequence of operations given below. After each operation write the front most and last item along with front and rear values.

Note: Initially queue contains three elements 60, 40, 30(inserted in the same sequence) with F=3. Write the initial Queue. Following operations are performed in sequence one after the other.

- Insert 100, 200, and 300

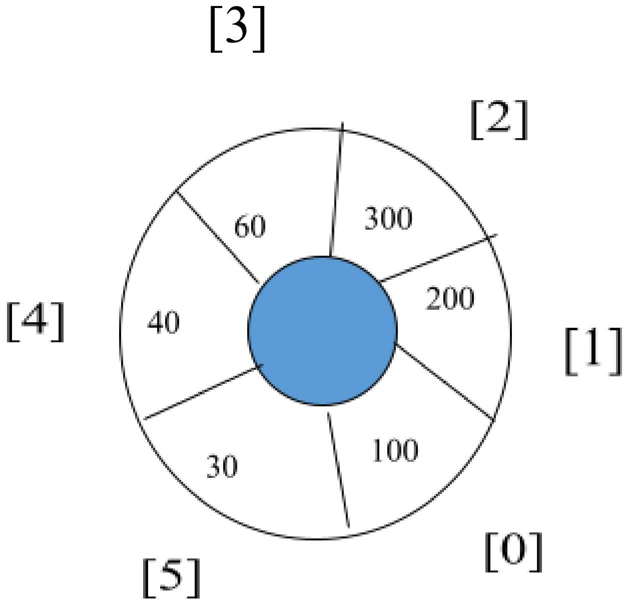
- Delete 3 items
- Insert 10, 20, 50

Insert 80



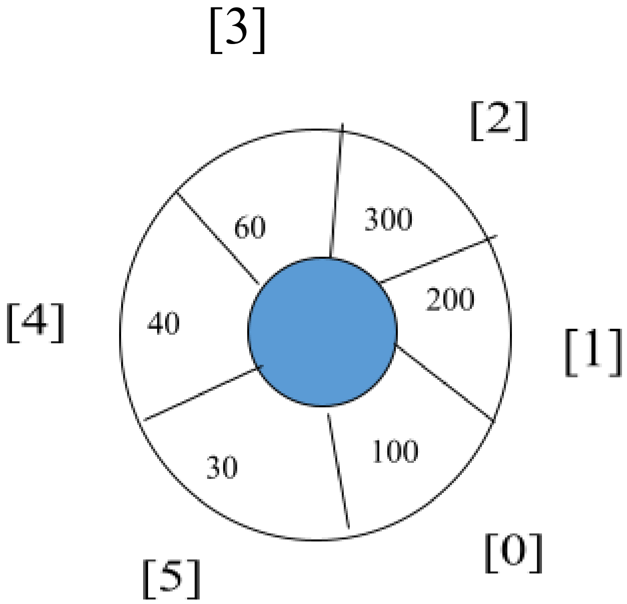
F=3

R=5



F=3

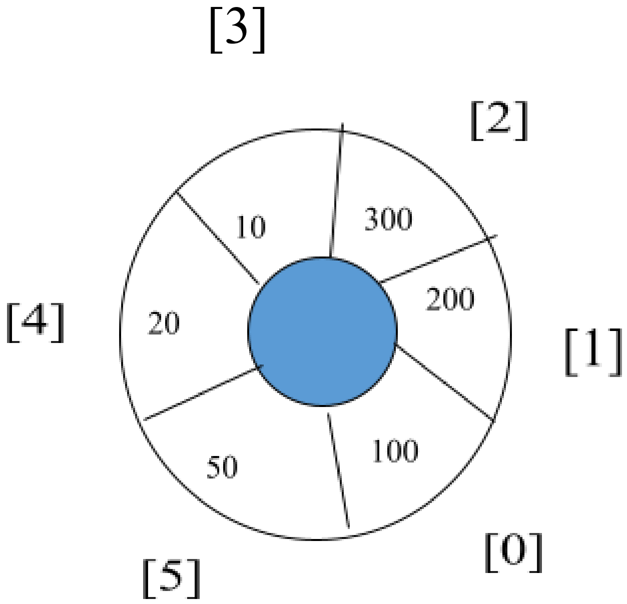
R=2



F=0

R=2

Deleted: 60 40 30



F=0

R=5

Priority Queue:

Ascending order Or Descending Order

Ascending order :

1) **Method 1:**

    InsertRear->constant O(1)

    Delete⍰Findmin/Findmax(Desc)-> Linear (N)

0	1	2	3	4	5
6	5	4	1	2	3

F=0

R =5

0	1	2	3	4	5
6	5	4	2	3	

F=0 ; R = 4

2) Method 2:

Insert in ascending order: Shifting is required-> Linear (N)

Delete� Delete Front �constant

0	1	2	3	4	5
1	2	3	4	5	6

F=0 ; R = 5   (6, 5, 4, 2, ,3, 1)

```
// Insert by order (Asc)
j= R  // 0  -1  1
 While(j>=0 &&  item<q.items[j])
       {
          Q.items[j+1] = Q.items[j]
           J- -
       }    Q.items[++j] =item    ++R
// to find minimum
 Min = Q.items[0]; i = 0
 For(i=1; i<=Q.rear; i++)
         If(q.items[i]<min)
           Min = q.items[i];   pos = i
```