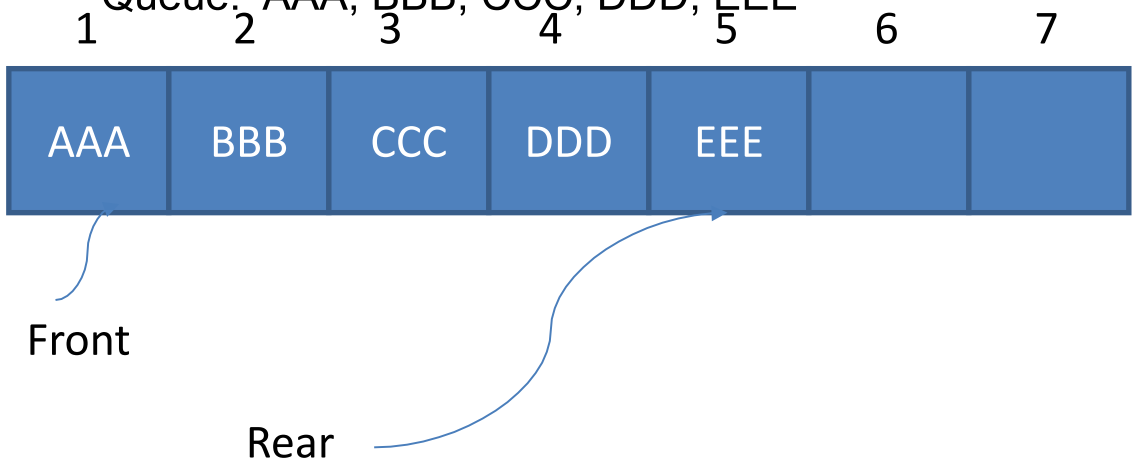
Queue

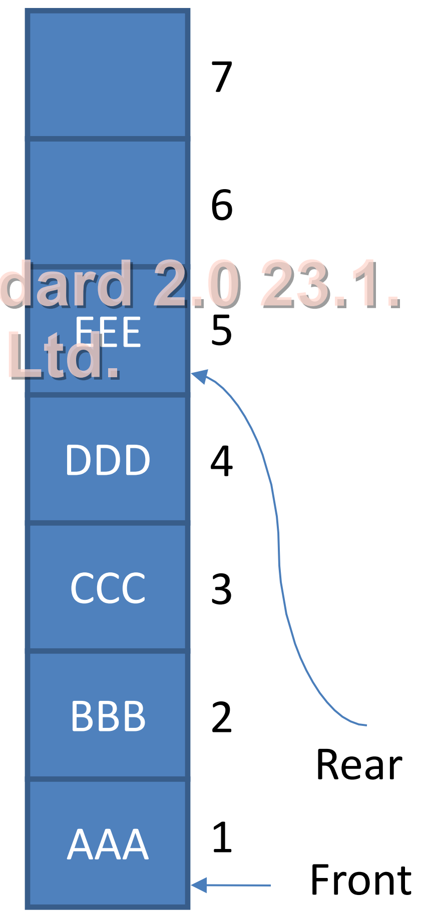
Queue (Linear Queue)

- It is a linear data structure consisting of list of items.
- In queue, data elements are added at one end, called ***the rear*** and removed from another end, called ***the front*** of the list.
- Two basic operations are associated with queue:
  1. "Insert" operation is used to insert an element into a queue.
  2. "Delete" operation is used to delete an element from a queue.

- FIFO list
- Example:

Queue: AAA, BBB, CCC, DDD, EEE





<u>Algorithms for Insert and Delete Operations in Linear Queue</u>

<u>For Insert Operation</u>

<u>Insert-Queue(Queue, Rear, Front, N, Item)</u>

Here, Queue is the place where to store data. Rear represents the location in which the data element is to be inserted and Front represents the location from which the data element is to be removed. Here N is the maximum size of the Queue and finally, Item is the new item to be added.

```
1. If Rear = N then Print: Overflow and Return.          /*…Queue already filled..*/

2. Set Rear := Rear +1

3. Set Queue[Rear] := Item

4. Return.
```

For Delete Operation

Delete-Queue(Queue, Front, Rear, Item)

Here, Queue is the place where data are stored. Rear represents the location in which the data element is to be inserted and Front represents the location from which the data element is to be removed. Front element is assigned to Item.

```
1. If Front = N+1   then Print: Underflow and Return.     /*…Queue Empty

2. Set Item := Queue[Front]

3. Set Front := Front + 1

4. Return.
```

Example: Consider the following queue (linear queue).

Rear = 4 and Front = 1 and N = 7

10	50	30	40			
1	2	3	4	5	6	7

(1) Insert 20. Now Rear = 5 and Front = 1

10	50	30	40	20		
1	2	3	4	5	6	7

(2) Delete Front Element. Now Rear = 5 and Front = 2

	50	30	40	20		
1	2	3	4	5	6	7

(3) Delete Front Element. Now Rear = 5 and Front = 3

		30	40	20		
1	2	3	4	5	6	7

(4) Insert 60. Now Rear = 6 and Front = 3

		30	40	20	60	
1	2	3	4	5	6	7

Drawback of Linear Queue

- Once the queue is full, even though few elements from the front are deleted and some occupied space is relieved, it is not possible to add anymore new elements, as the rear has already reached the Queue's rear most position.

Circular Queue

- This queue is not linear but circular.
- Its structure can be like the following figure:
- In circular queue, once the Queue is full the "First" element of the Queue becomes the "Rear" most element, if and only if the "Front" has moved forward. otherwise it will again be a "Queue overflow" state.

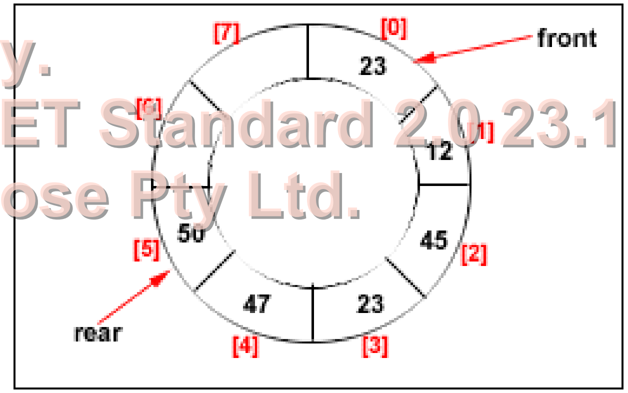


Figure: Circular Queue having Rear = 5 and Front = 0

Algorithms for Insert and Delete Operations in Circular Queue

For Insert Operation

Insert-Circular-Q(CQueue, Rear, Front, N, Item)

Here, CQueue is a circular queue where to store data. Rear represents the location in which the data element is to be inserted and Front represents the location from which the data element is to be removed. Here N is the maximum size of CQueue and finally, Item is the new item to be added. Initailly Rear = 0 and Front = 0.

```
1. If Front = 0 and Rear = 0 then Set Front := 1 and go to step 4.
2. If  Front =1 and Rear = N or Front = Rear + 1
           then Print: "Circular Queue Overflow" and Return.
3. If Rear = N  then Set Rear := 1 and go to step 5.
4. Set Rear := Rear + 1
5. Set CQueue [Rear] := Item.
6. Return
```

For Delete Operation

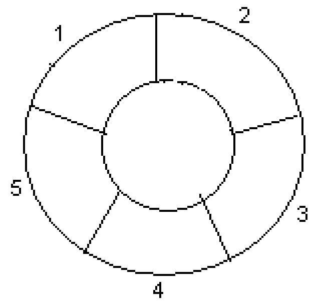
Delete-Circular-Q(CQueue, Front, Rear, Item)

Here, CQueue is the place where data are stored. Rear represents the location in which the data element is to be inserted and Front represents the location from which the data element is to be removed. Front element is assigned to Item. Initially, Front = 1.

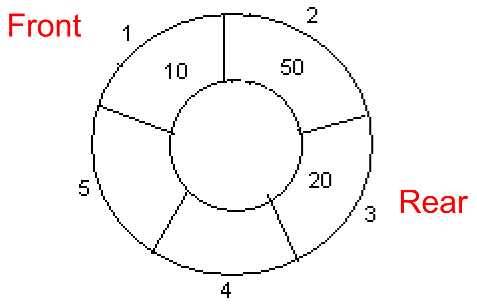
```
1. If Front = 0 then
        Print: “Circular Queue Underflow” and Return.    /*..Delete without Insertion
2. Set Item := CQueue [Front]
3. If Front = N then Set Front = 1 and Return.
4. If Front = Rear then Set Front = 0 and Rear = 0 and Return.
5. Set Front := Front + 1
6. Return.
```

Example: Consider the following circular queue with N = 5.

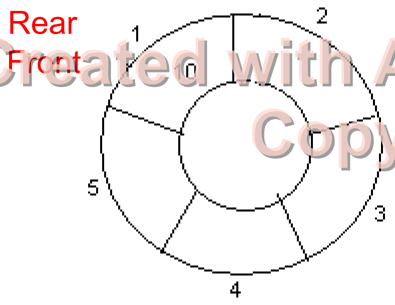
1. Initially, Rear = 0, Front = 0.



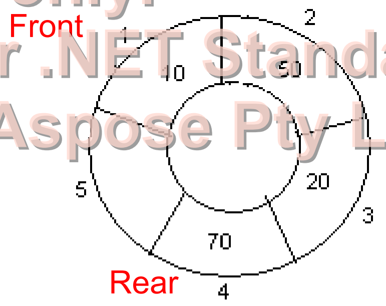
4. Insert 20, Rear = 3, Front = 0.



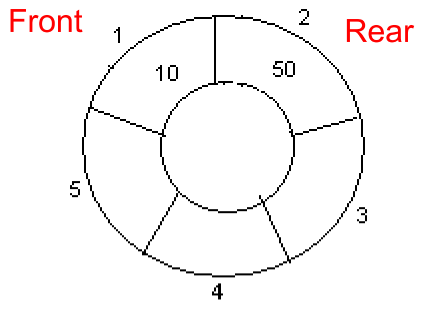
2. Insert 10, Rear = 1, Front = 1.



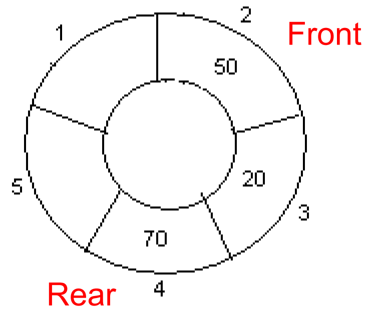
5. Insert 70, Rear = 4, Front = 1.



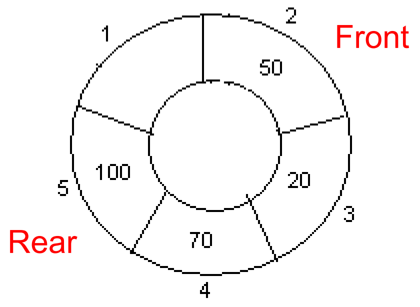
3. Insert 50, Rear = 2, Front = 1.



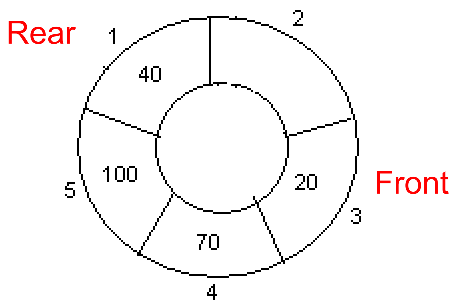
6. Delete front, Rear = 4, Front = 2.



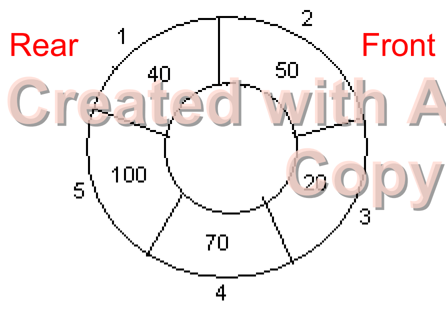
7. Insert 100, Rear = 5, Front = 2.



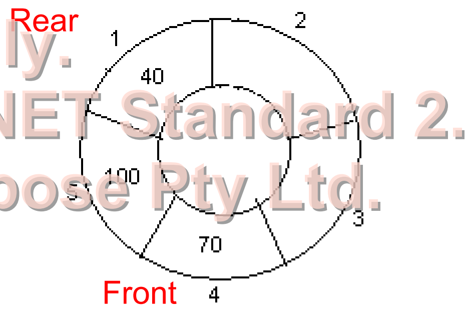
10. Delete front, Rear = 1, Front = 3.



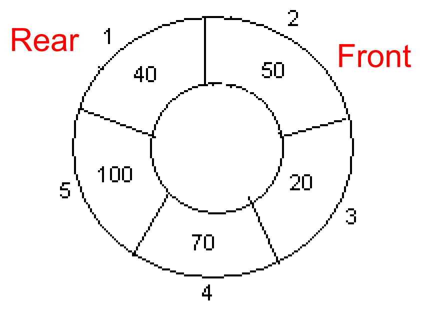
8. Insert 40, Rear = 1, Front = 2.



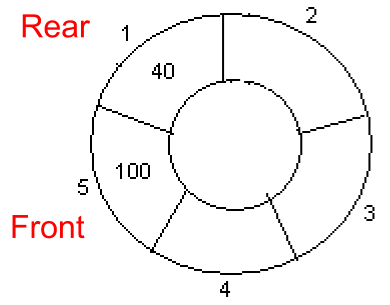
11. Delete front, Rear = 1, Front = 4.



9. Insert 140, Rear = 1, Front = 2.
As Front = Rear + 1, so Queue overflow.



12. Delete front, Rear = 1, Front = 5.



END!!!