QUEUE

Prof Viswanathan V
VIT Chennai

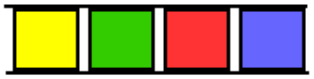
Queue

- **Queue** is a linear data structure in which the element is inserted from one end called the **REAR** and the removal of existing element takes place from the other end called as **FRONT**.

- This makes queue as **FIFO**(First in First Out) data structure, which means that element inserted first will be removed first.
- The process to add an element into queue is called **Enqueue** and the process of removal of an element from queue is called **Dequeue**.

- Applications : ATM centre, waiting list in registration

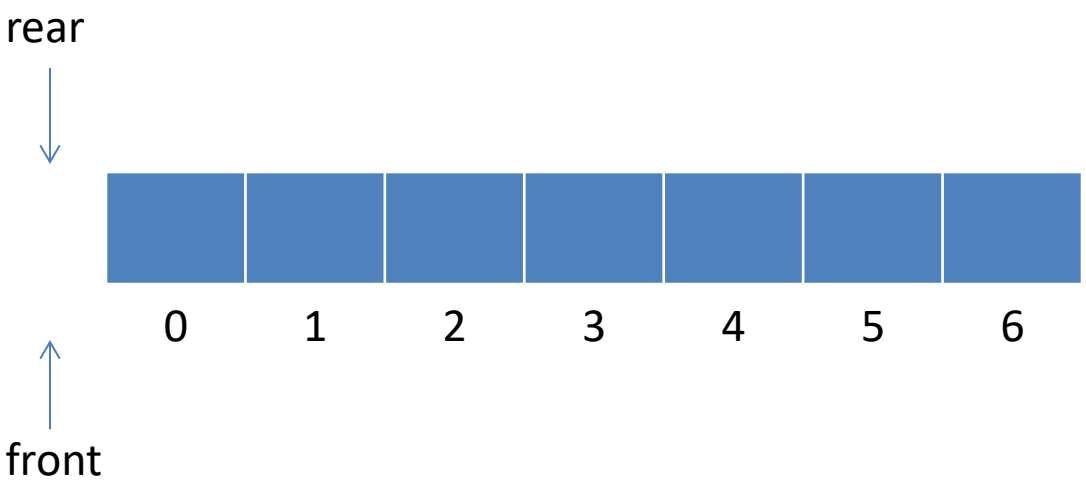
IN



OUT

Pseudo code to check whether the Queue is empty or not

```
ISEMPTY()
{
if  (front =-1 and rear = -1 )
    return true
else
   return false
}
```

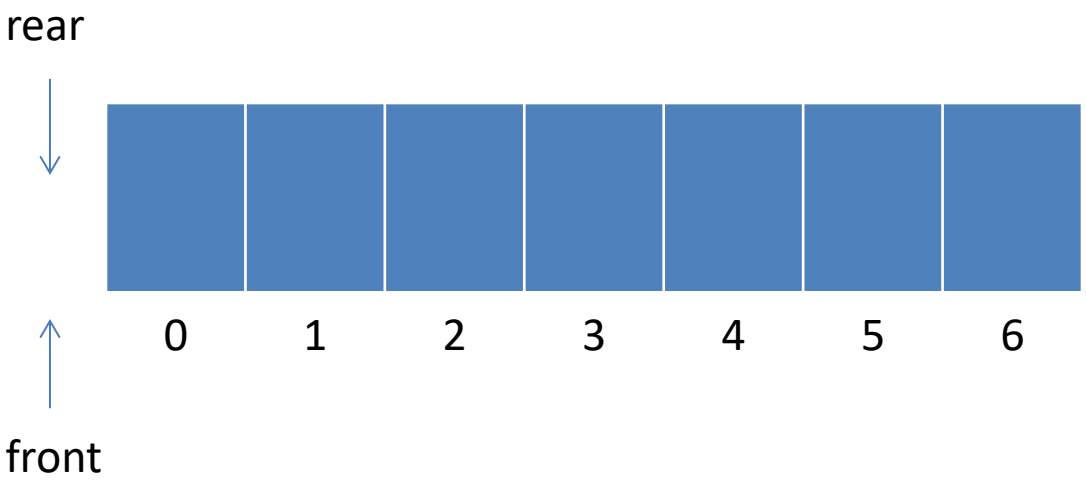


Pseudo code to check whether the Queue is full or not

```
ISFULL()
{
If  ( rear = size(A) -1  )
{
  print "Queue full"
  return true
}
else
  return false
}
```

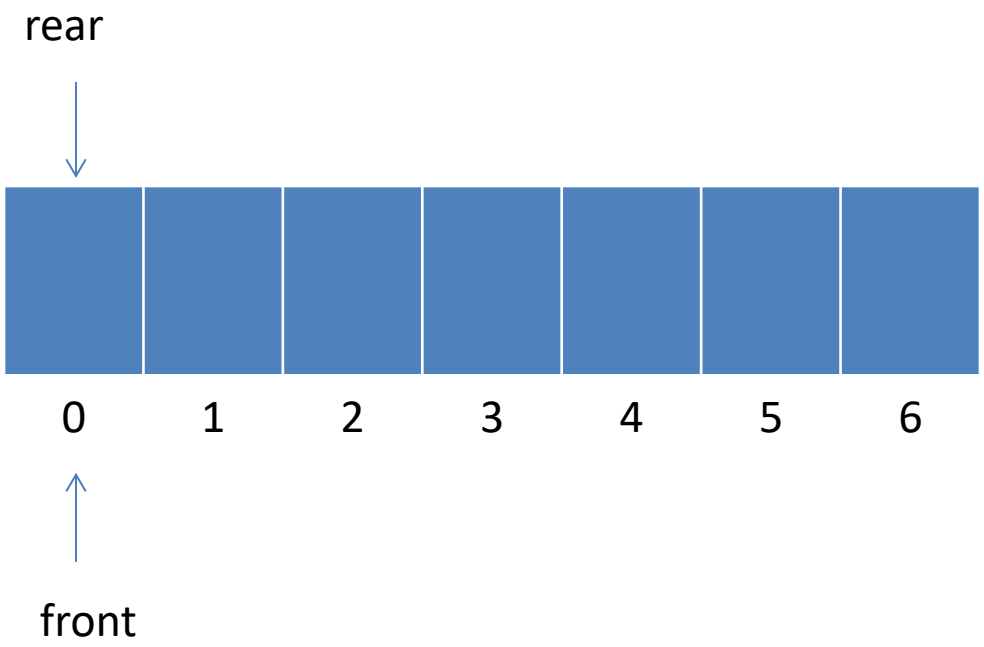
Pseudo code to insert an element into the queue

```
Enqueue(x)
{
   if  ISFULL()
      return
  else if  (front=-1 and rear = -1)
        {
         front = 0
         rear = 0
        }
       else
           rear = rear +1
 A(rear) = x
}
```



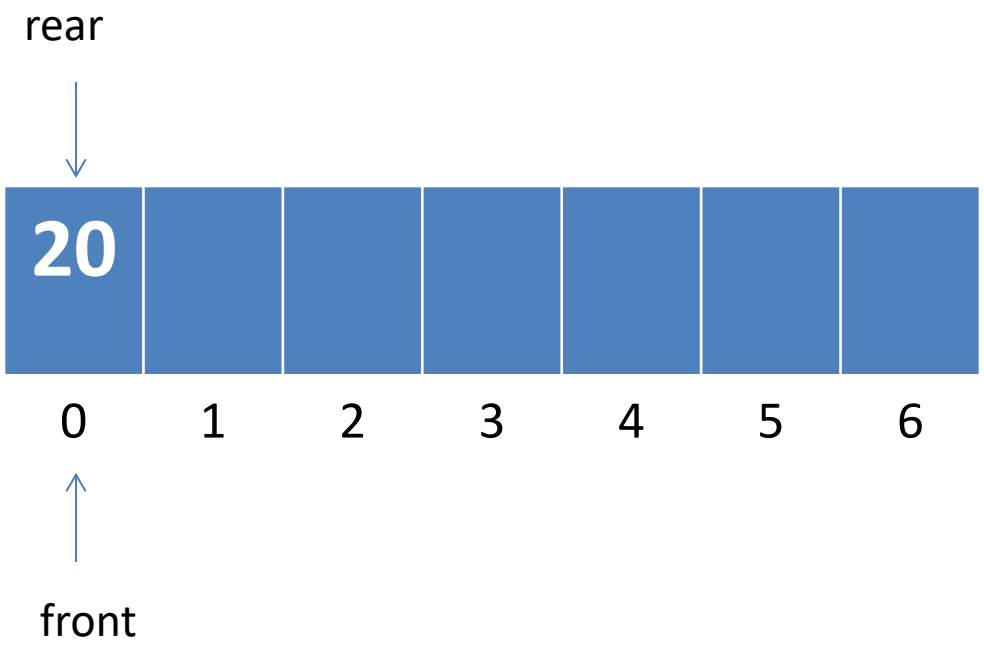
Enqueue(20)

```
Enqueue(x)
{
   if  ISFULL()
      return
   else if  (front=-1 and rear = -1)
         {
           front = 0
           rear = 0
          }
        else
           rear = rear +1
  A(rear) = x
}
```

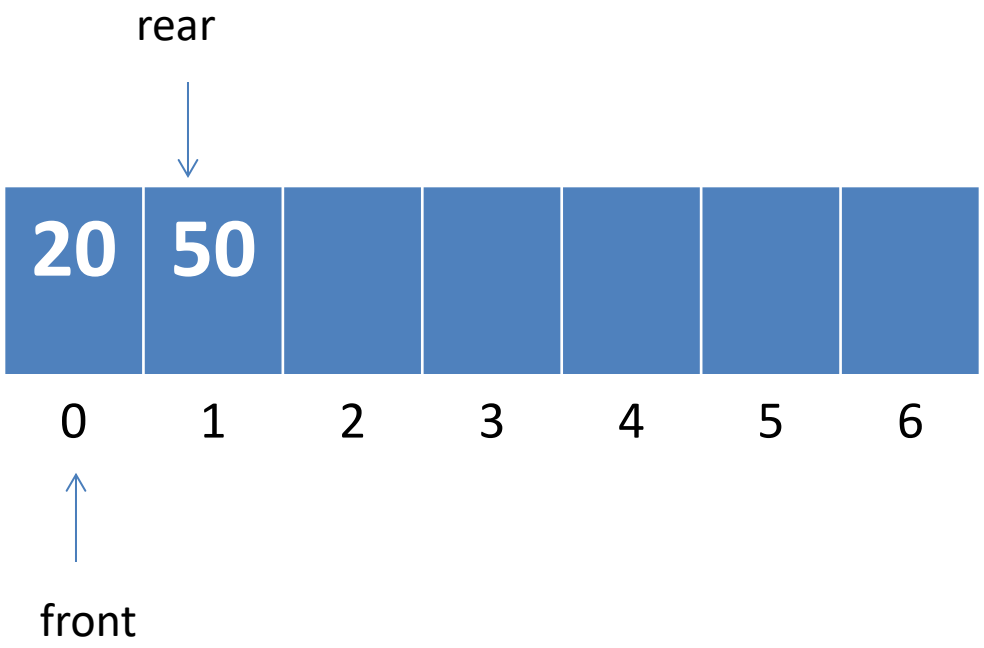


Enqueue(20)

```
Enqueue(x)
{
   if  ISFULL()
      return
   else if  (front=-1 and rear = -1)
         {
           front = 0
           rear = 0
          }
        else
           rear = rear +1
  A(rear) = x
}
```



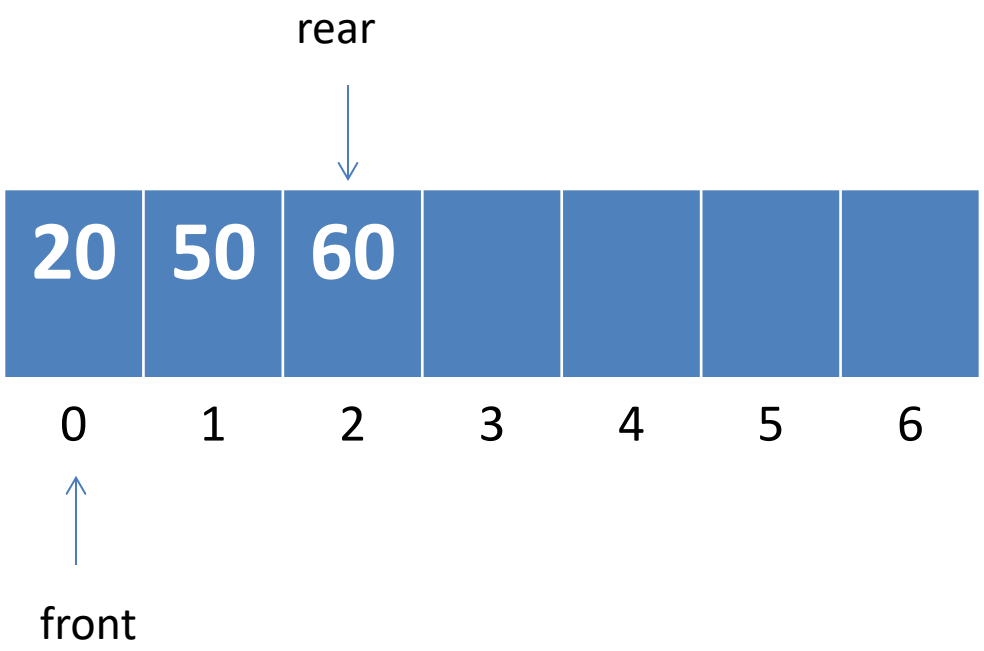
```
Enqueue(x)
{
   if  ISFULL()
      return
   else if  (front=-1 and rear = -1)
         {
           front = 0
           rear = 0
          }
        else
           rear = rear +1
  A(rear) = x
}
```



Enqueue(20)

Enqueue(50)

```
Enqueue(x)
{
   if  ISFULL()
      return
   else if  (front=-1 and rear = -1)
         {
           front = 0
           rear = 0
          }
         else
           rear = rear +1
  A(rear) = x
}
```



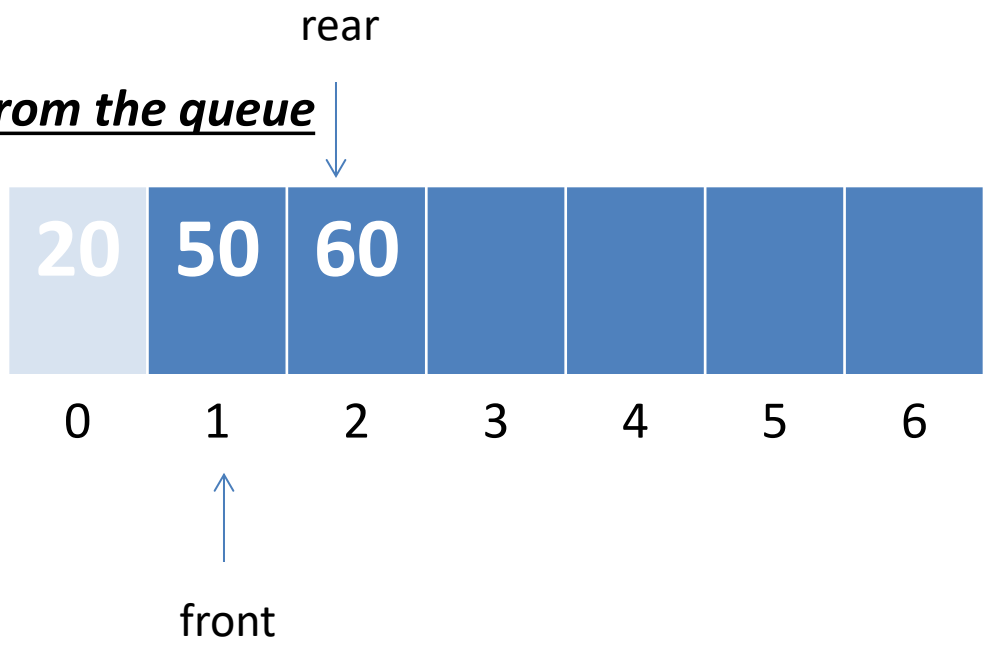
Enqueue(20)

Enqueue(50)

Enqueue(60)

Pseudo code to delete an element from the queue

```
Dequeue()
{
  if ISEMPTY()
     return
  else if  front = rear
           front = rear = -1
        else
           front = front +1
}
```



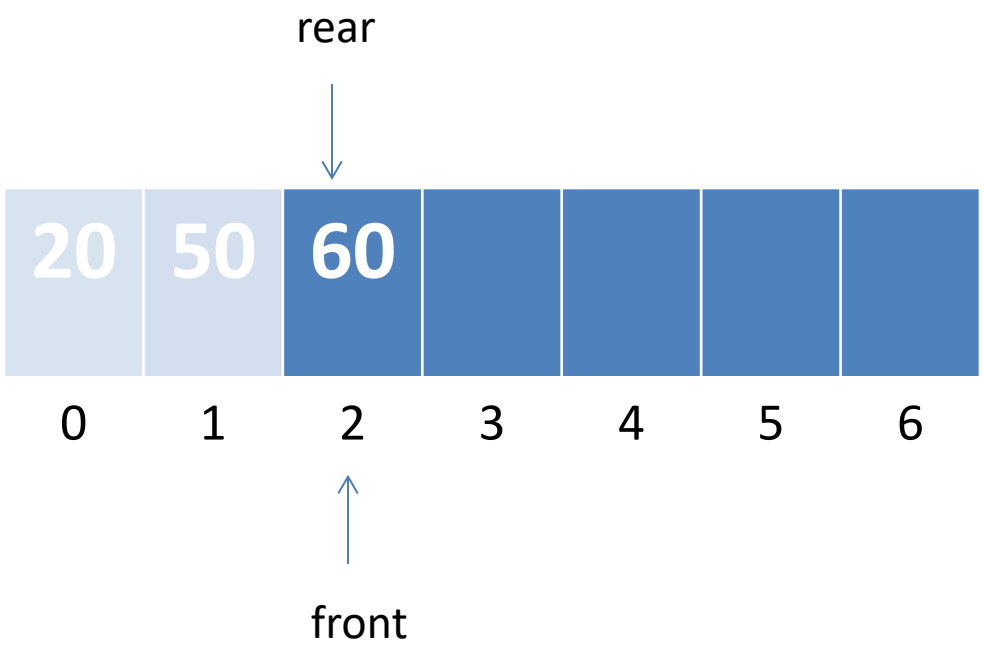
Enqueue(20)

Enqueue(50)

Enqueue(60)

Dequeue()

```
Dequeue()
{
  if ISEMPTY()
      return
  else if  front = rear
            front = rear = -1
         else
             front = front +1
}
```



Enqueue(20)

Enqueue(50)

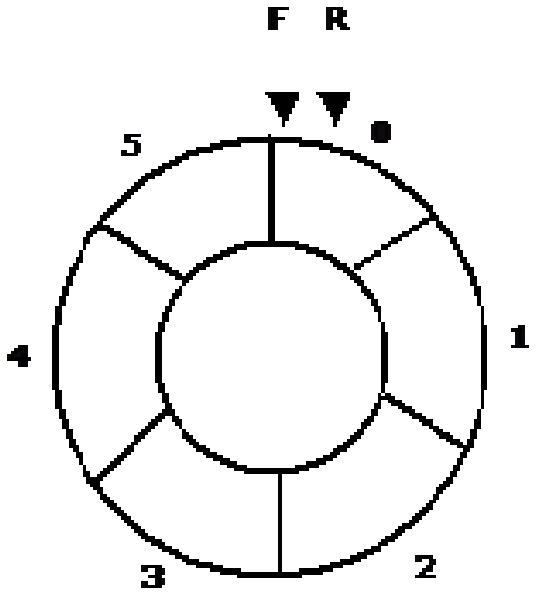
Enqueue(60)

Dequeue()

Dequeue()

Circular Queue Representation using Arrays

Let us consider a circular queue, which can hold maximum (MAX) of six elements. Initially the queue is empty.

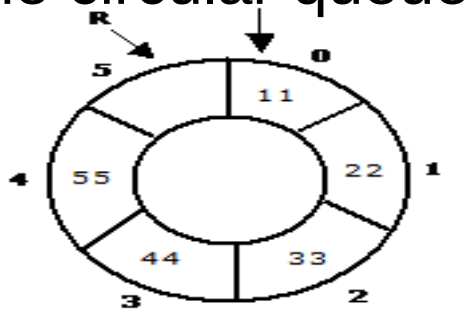


Circular Queue

Queue Empty
MAX = 6
FRONT = REAR = 0
COUNT = 0

Insertion and Deletion operations on a Circular Queue

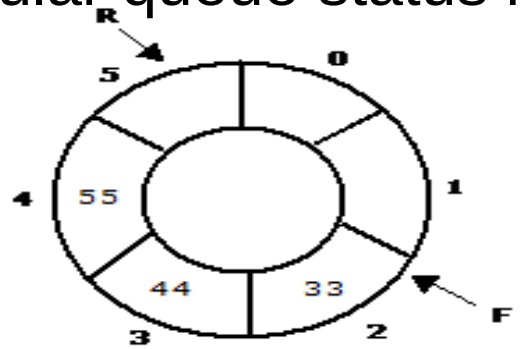
Insert new elements 11, 22, 33, 44 and 55 into the circular queue. The circular queue status is:



Circular Queue

FRONT = 0, REAR = 5
REAR = REAR % 6 = 5
COUNT = 5

Now, delete two elements 11, 22 from the circular queue. The circular queue status is as follows:

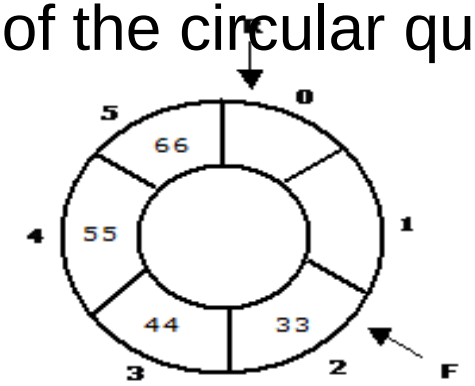


Circular Queue

FRONT = (FRONT + 1) % 6 = 2
REAR = 5
COUNT = COUNT - 1 = 3

Insertion and Deletion operations on a Circular Queue

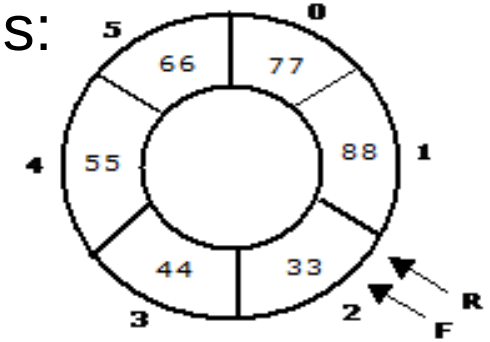
Again, insert another element 66 to the circular queue. The status of the circular queue is:



Circular Queue

FRONT = 2
REAR = (REAR + 1) % 6 = 0
COUNT = COUNT + 1 = 4

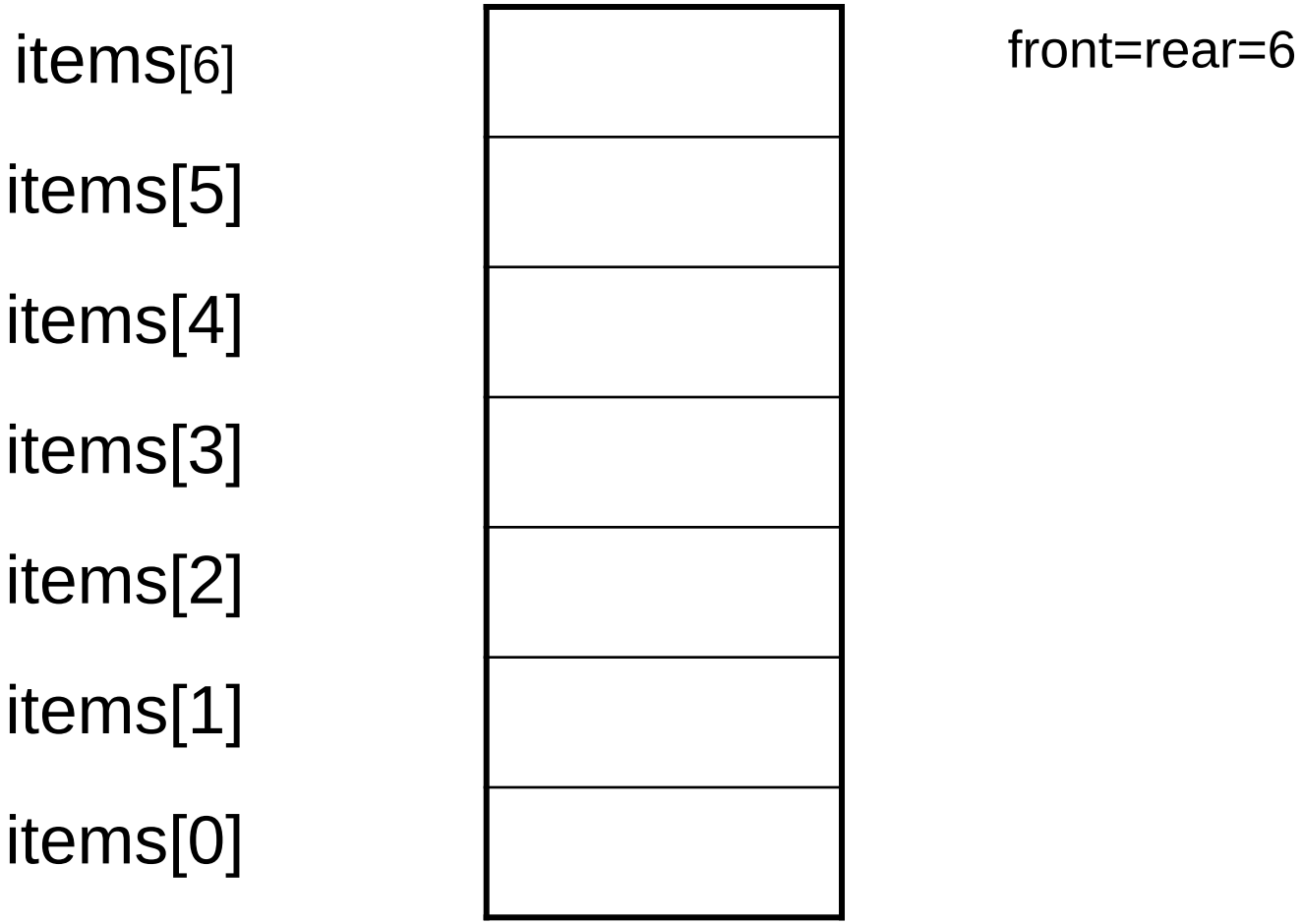
Again, insert 77 and 88 to the circular queue. The status of the Circular queue is:



Circular Queue

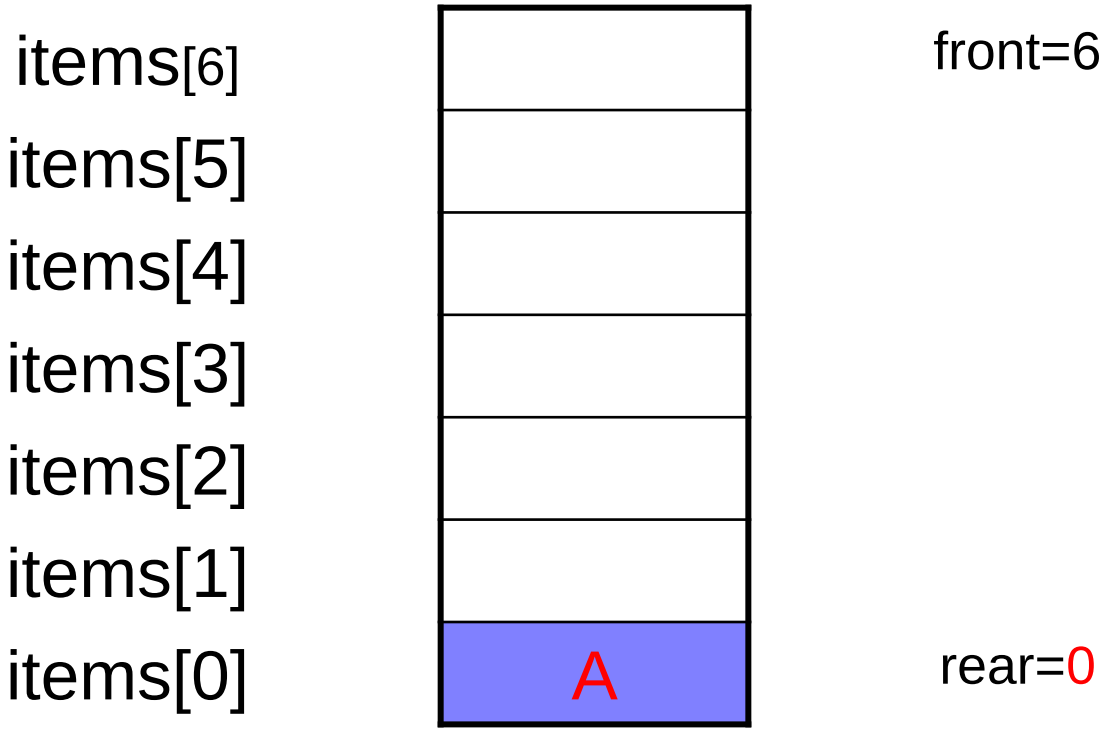
FRONT = 2, REAR = 2
REAR = REAR % 6 = 2
COUNT = 6

Initialize the queue.



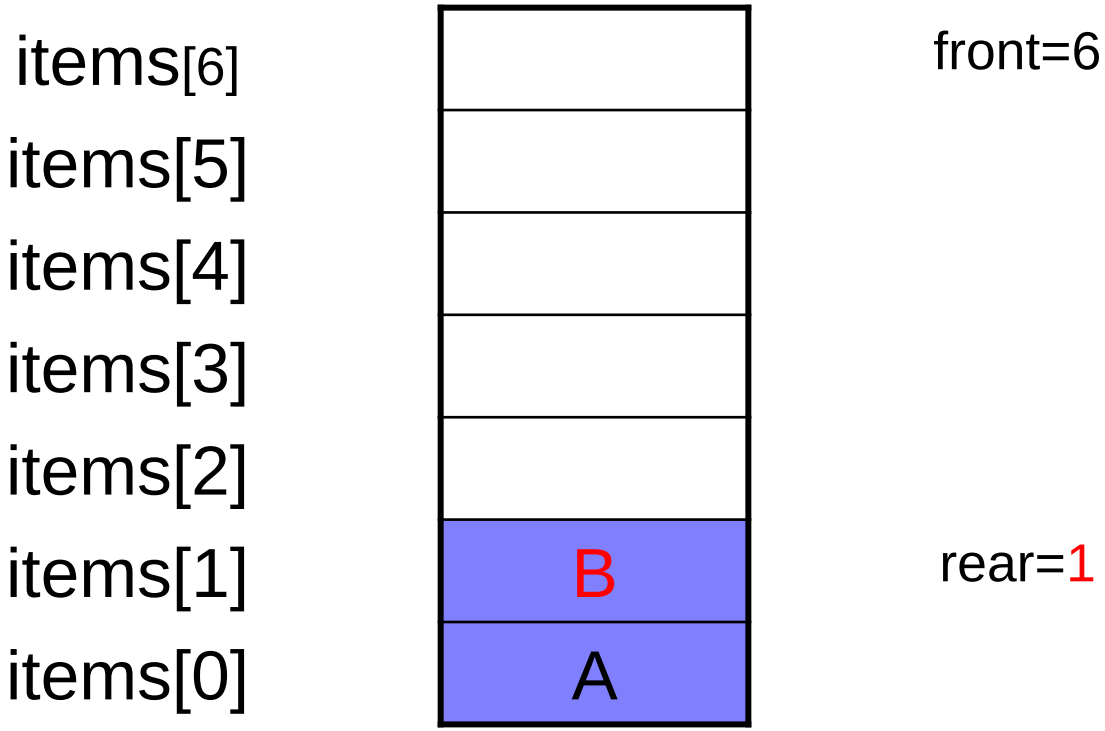
Insert items into circular queue

- ***Insert* A,B,C to the rear of the queue.**



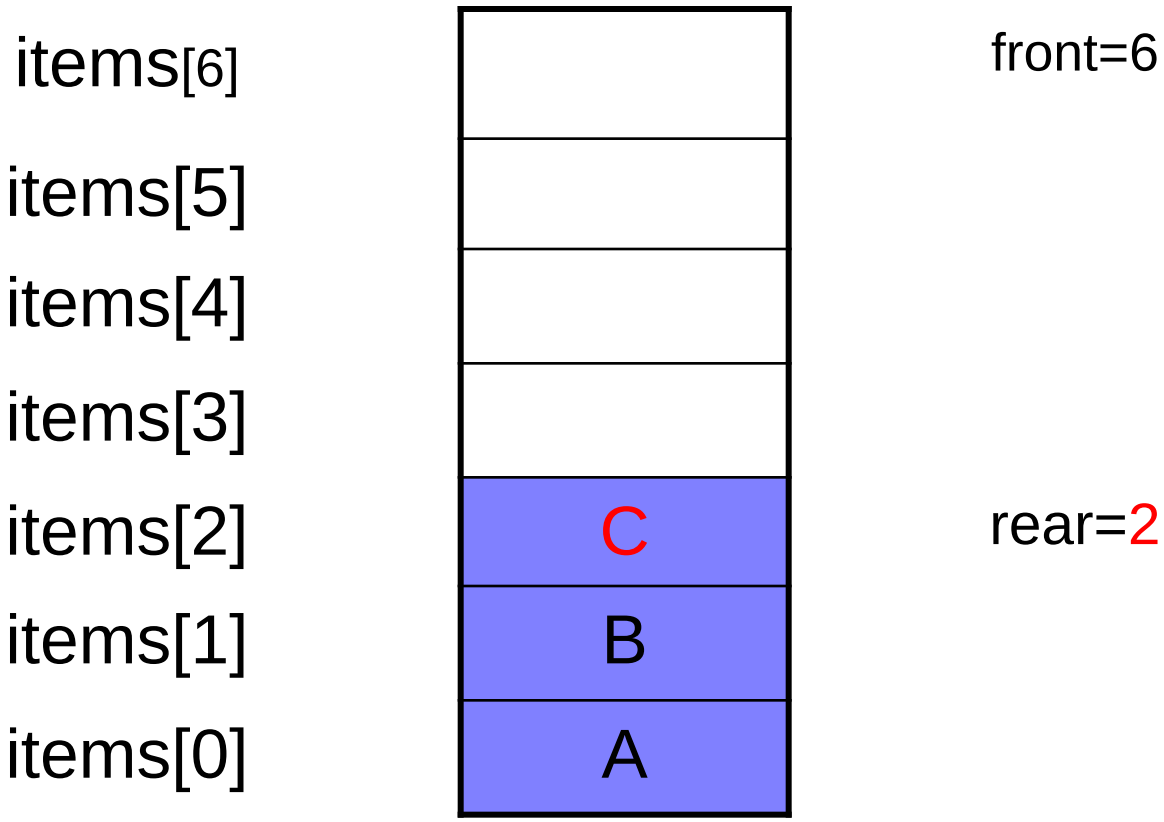
Insert items into circular queue

- ***Insert* A,B,C to the rear of the queue.**



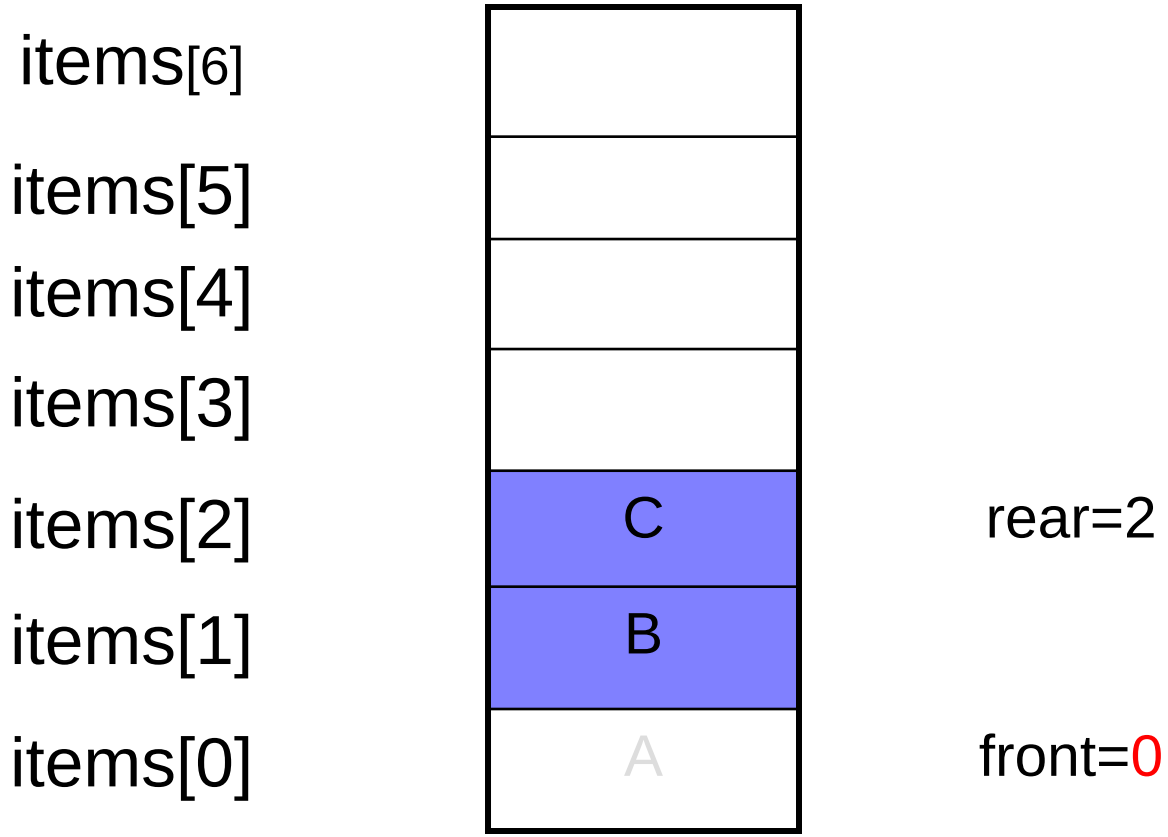
Insert items into circular queue

- ***Insert* A,B,C to the rear of the queue.**



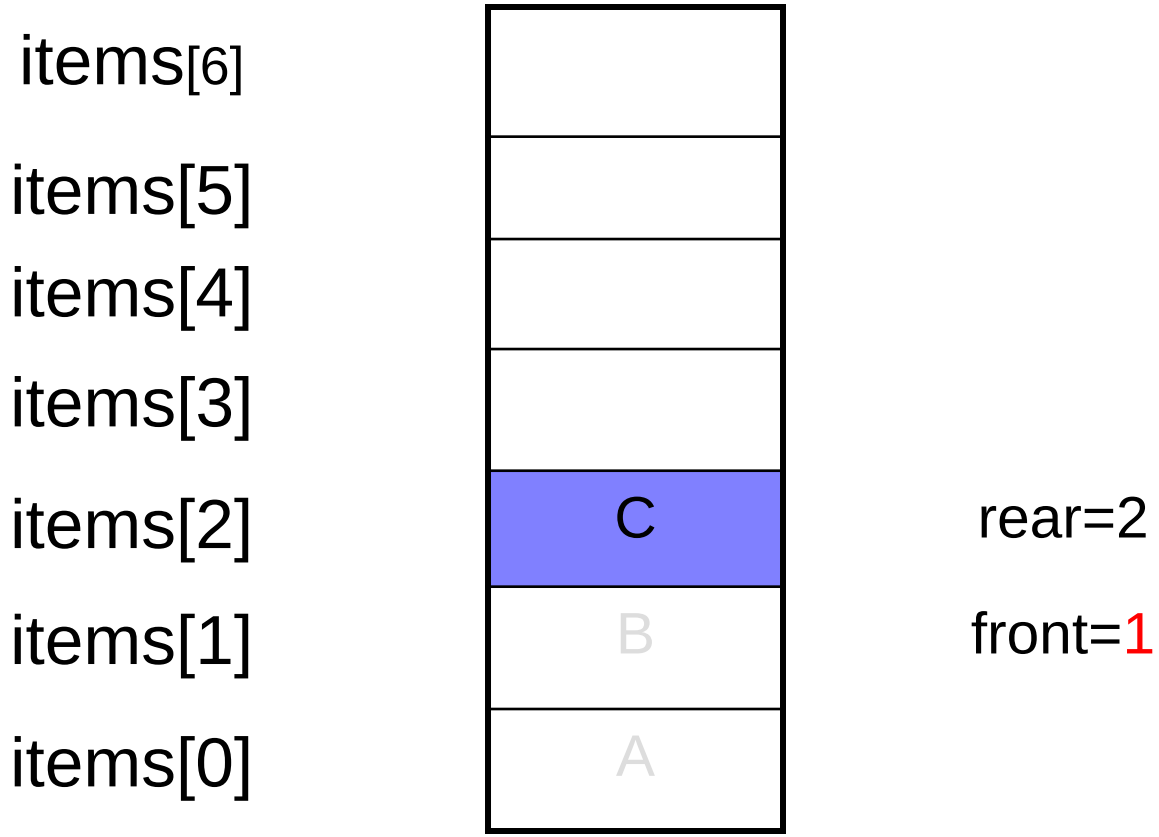
Remove items from circular queue

- ***Remove* two items from the queue.**



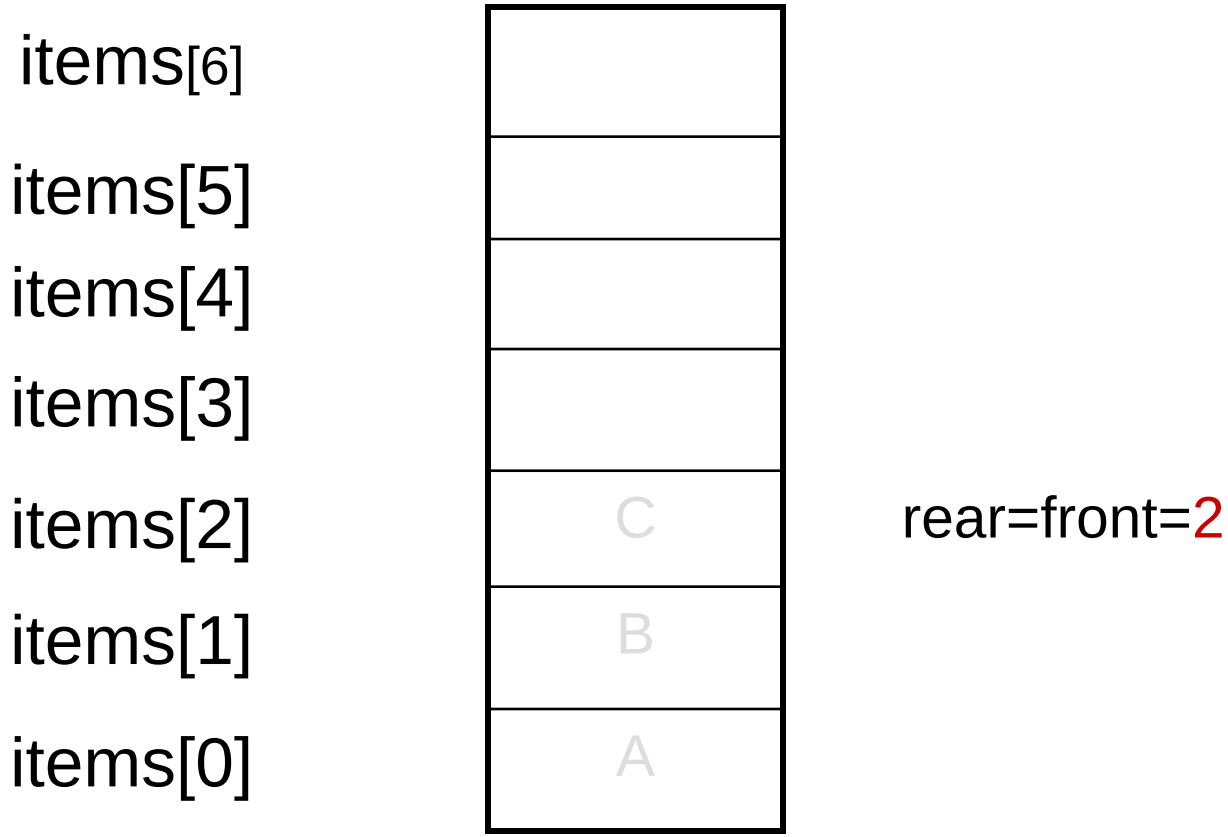
Remove items from circular queue

- ***Remove* two items from the queue.**

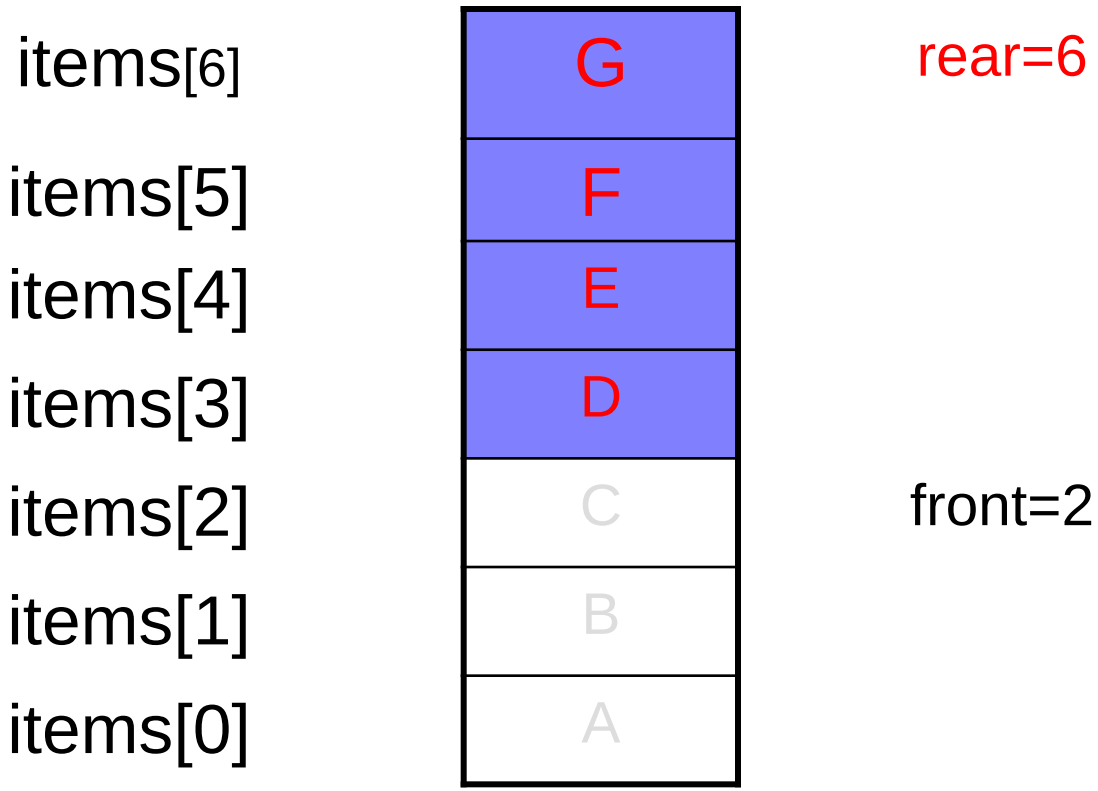


Remove items from circular queue

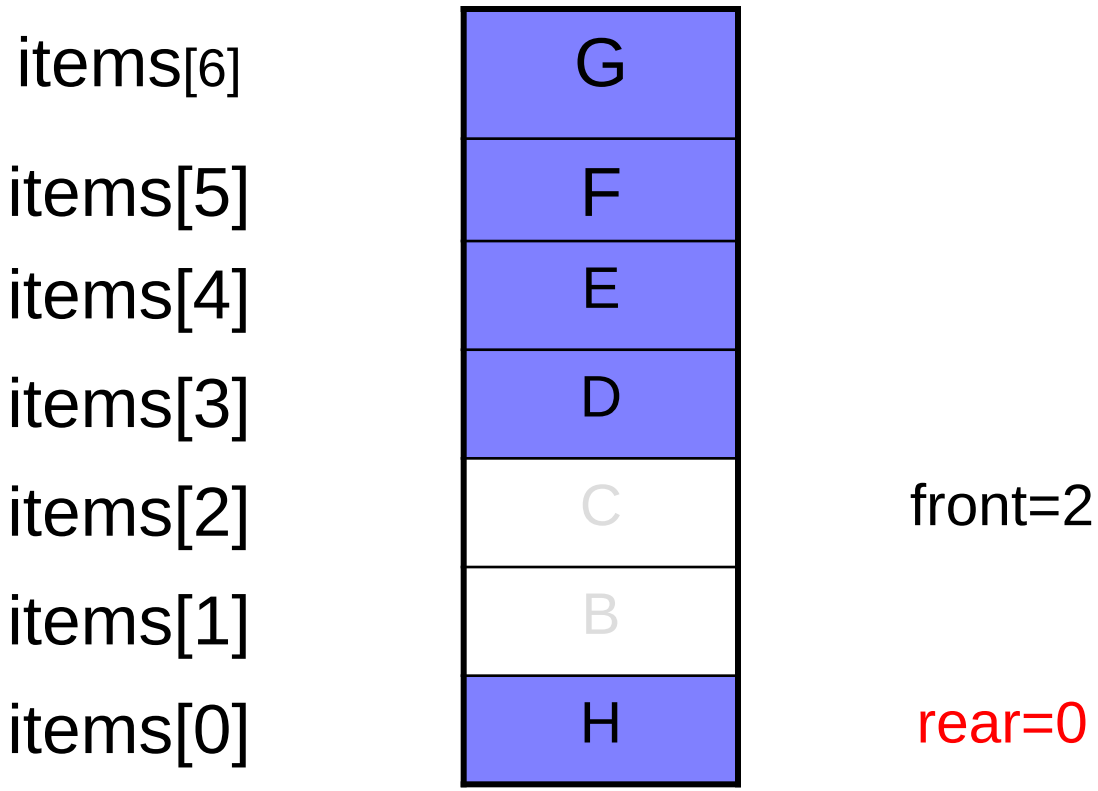
- ***Remove* one more item from the queue.**



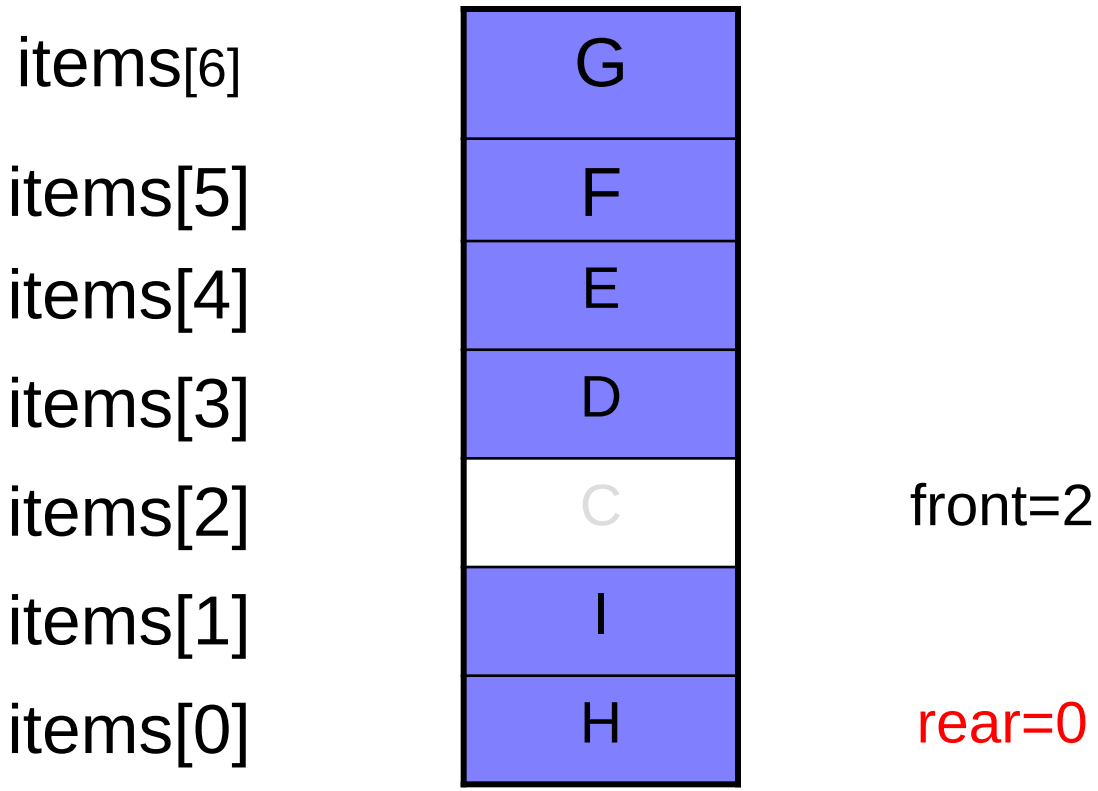
- ***Insert* D,E,F,G to the queue.**



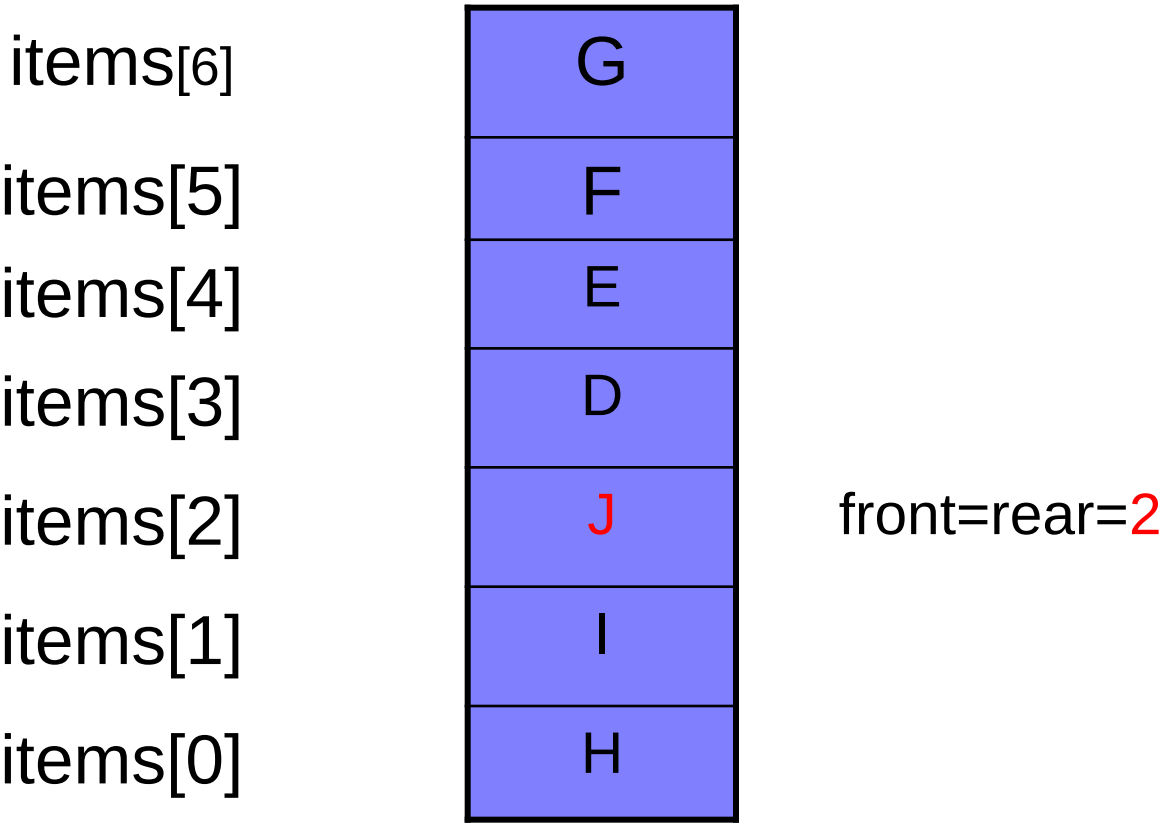
- ***Insert*** **H and I to the queue.**



- ***Insert*** **H and I to the queue.**



- ***Insert* J to the queue.**




Thank
you