DATA STRUCTURES

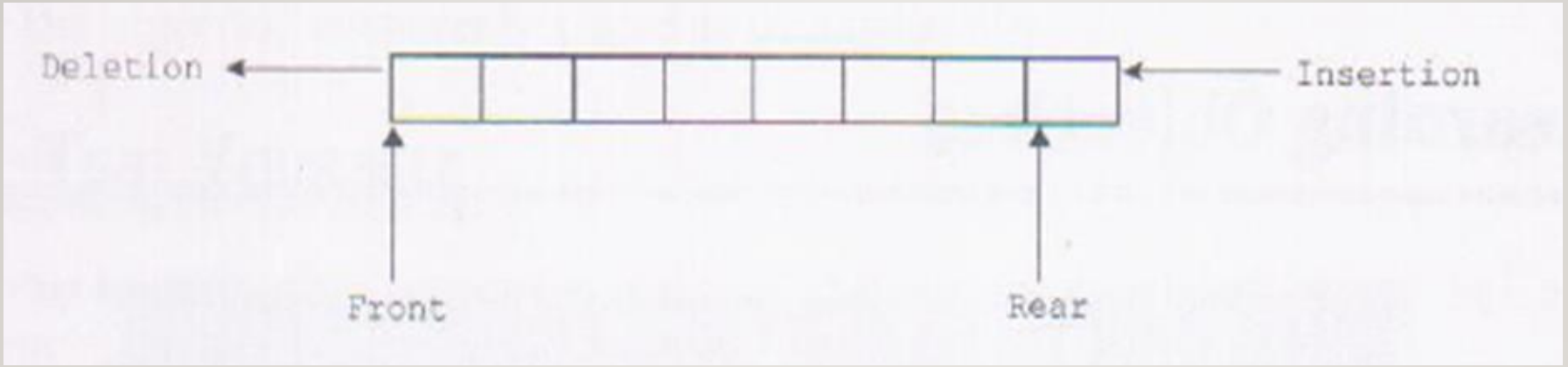
LECTURE-9

QUEUE

Dr. Sumitra Kisan

- A queue is a linear list in which elements can be added at one end and elements can be removed only at other end.
- That is, the first element added to the queue is the first element to be removed.
- Queue works on the principle of ‘ first-in-first-out’(FIFO).
- A common example of queue is people waiting in line at ticket window.

- A queue is a linear data structure in which an element is inserted at one end and element is deleted from other end.
- The end of the queue from which the element is deleted is known as **front.**
- The end at which new element is added is known as **rear**.



	9	7	18	14	36	45			
0	1	2	3	4	5	6	7	8	9

In this example, front = 1 and rear = 6.

TYPES OF QUEUE

1.CIRCULAR QUEUE

2.DEQUEUE

3.PRORITY QUEUE

OPERATIONS ON QUEUE

- The two basic operations that can be performed on a queue are:

- **Insert : to insert an element at rear of the queue**
- **Delete: to delete an element from front of the queue.**

- Before inserting a new element in the queue, it is necessary to test the condition of overflow.

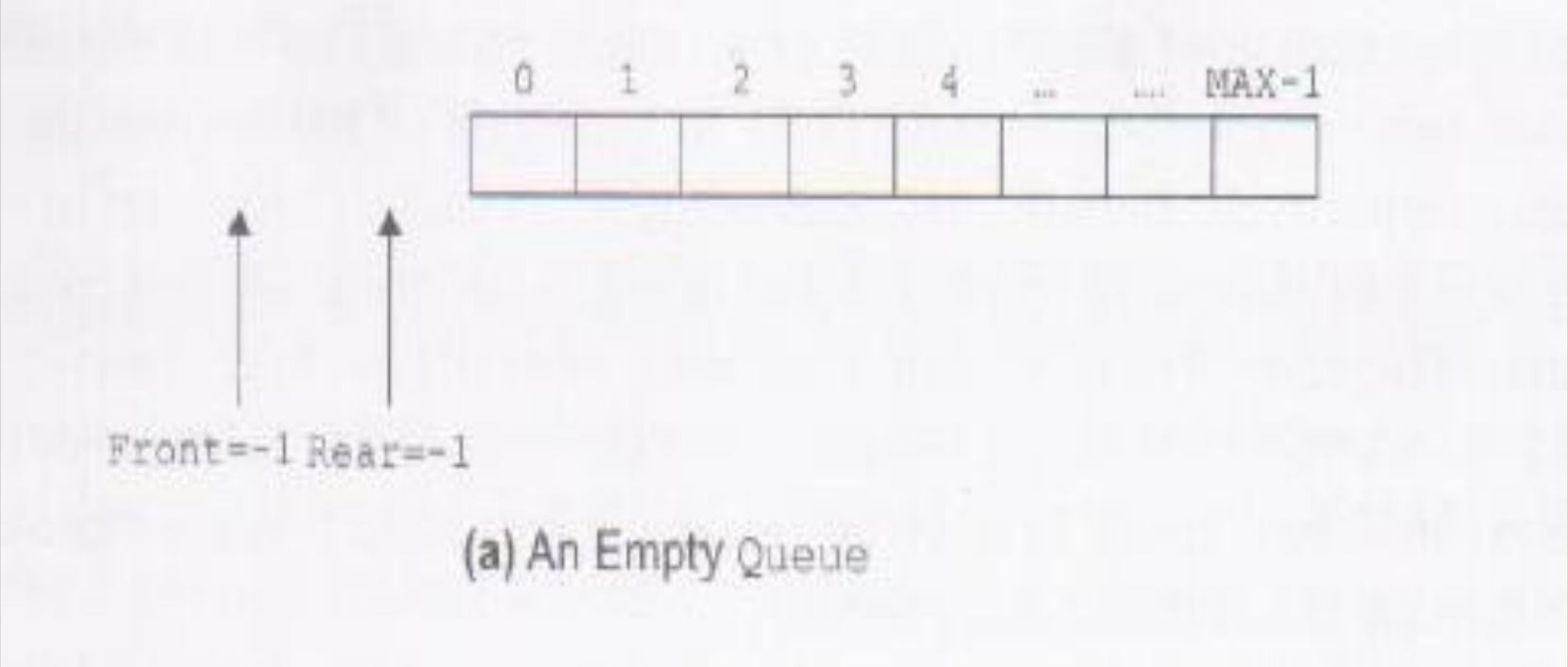
- *Overflow* occurs when the queue is full and there is no space for a new element.

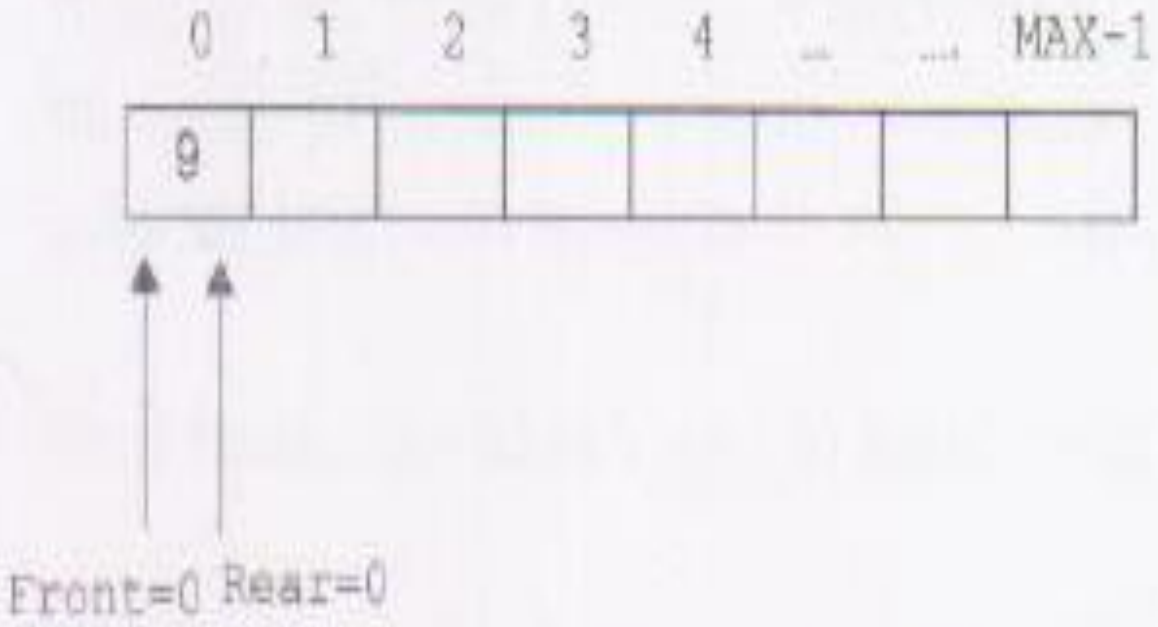
- Similarly, before removing the element from the queue, it is necessary to check the condition of underflow.

- Underflow occurs when the queue is empty.

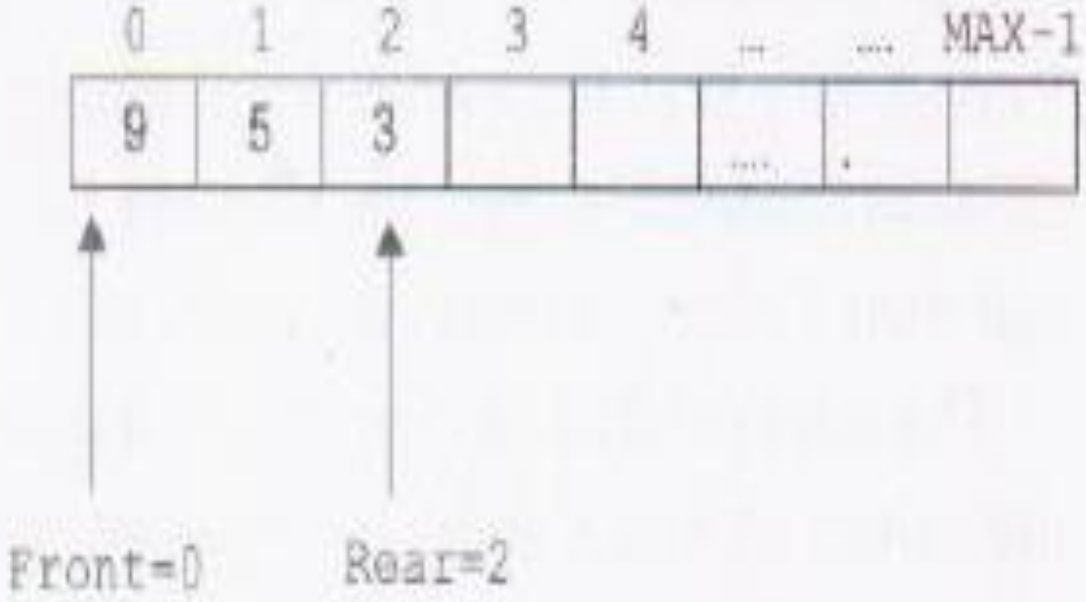
MEMORY REPRESENTATION OF QUEUE

- A queue can be represented in memory either as an array or as a singly linked list.
- Queue is full, when rear = MAX – 1, where MAX specifies the maximum number of elements that the queue can hold . MAX is size of array, if queue is represented as array. ***overflow***
- Queue is empty. If front = -1 and rear = -1 ***underflow***

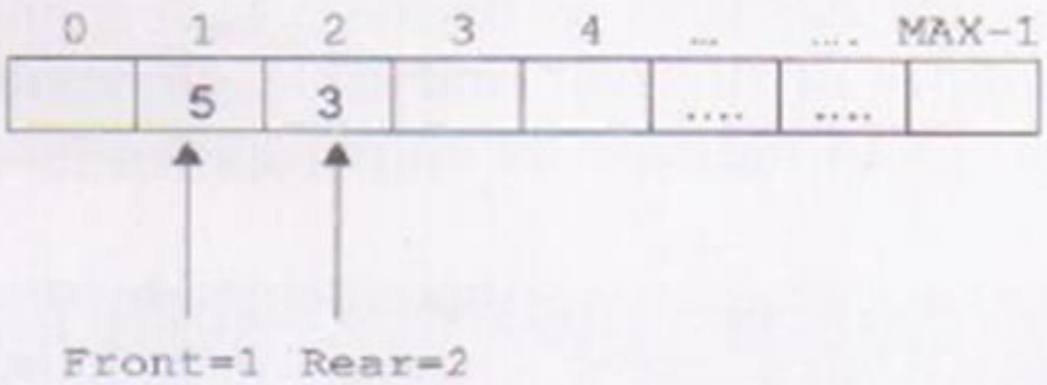




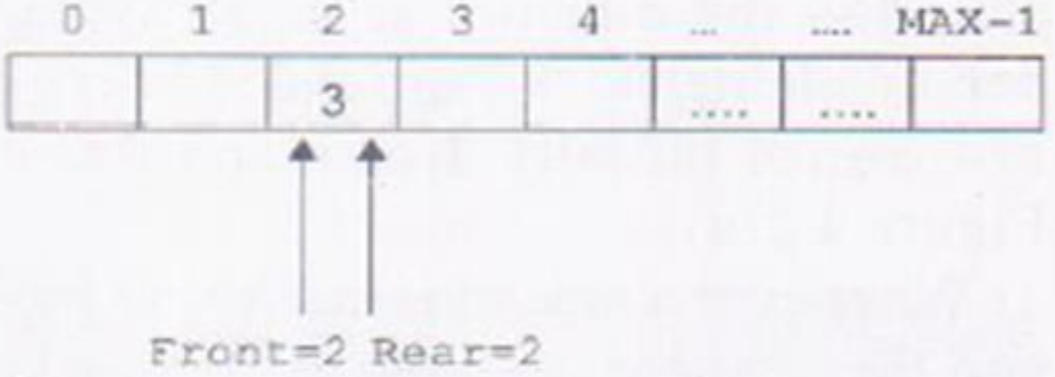
(b) Queue after Insertion of First Element



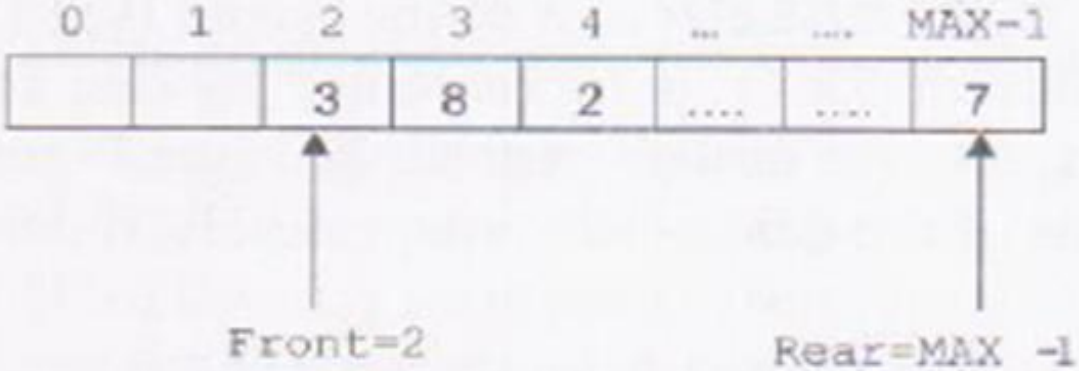
(c) Queue after Insertion of Few Elements



(d) Queue after Deleting First Element



(e) Queue after Deleting Second Element



(f) Queue having Vacant Space though Rear = MAX-1

To implements queue as an array in C the following structure named queue is used:

```
struct queue
 {
    int item[MAX];
    int Front;
    int Rear;
  };
```

Insert Operations On Queue

```
Step 1:   IF REAR=MAX-1, then;
                Write OVERFLOW
                Goto Step 4
           [END OF IF]
Step 2:  IF FRONT == -1 and REAR = -1, then
              SET FRONT = REAR = 0
          ELSE
             SET REAR = REAR + 1
         [END OF IF]
Step 3: SET QUEUE[REAR] = VAL
Step 4: Exit
```

Deletion Operation

Step 1: IF FRONT = -1 OR FRONT > REAR, then
Write UNDERFLOW
Goto Step 3
[END OF IF]

Step 2: SET VAL = QUEUE[FRONT]
SET FRONT = FRONT + 1

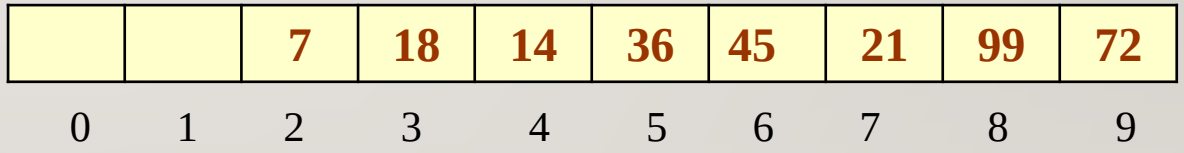
Step 3: Exit

Stack v/s Queue

Stack	Queue
A Linear List Which allows insertion or deletion of an element at one end only is called as Stack	A Linear List Which allows insertion and one end and deletion at another end is called as Queue
Since insertion and deletion of an element are performed at one end of the stack, the elements can only be removed in the opposite order of insertion.	Since insertion and deletion of an element are performed at opposite end of the queue, the elements can only be removed in the same order of insertion
Stack is called as Last In First Out (LIFO) List.	Queue is called as First In First Out (FIFO) List.
The most accessible element is called as TOP of the stack and insertion and deletion is performed at this end only.	Insertion of element is performed at FRONT end and deletion is performed from REAR end
Example of stack is arranging plates in one above one.	Example is ordinary queue is supermarket billing queue.
Insertion operation is referred as PUSH and deletion operation is referred as POP	Insertion operation is referred as ENQUEUE and deletion operation is referred as DQUEUE
Function calling in any languages uses Stack	Task Scheduling by Operating System uses queue

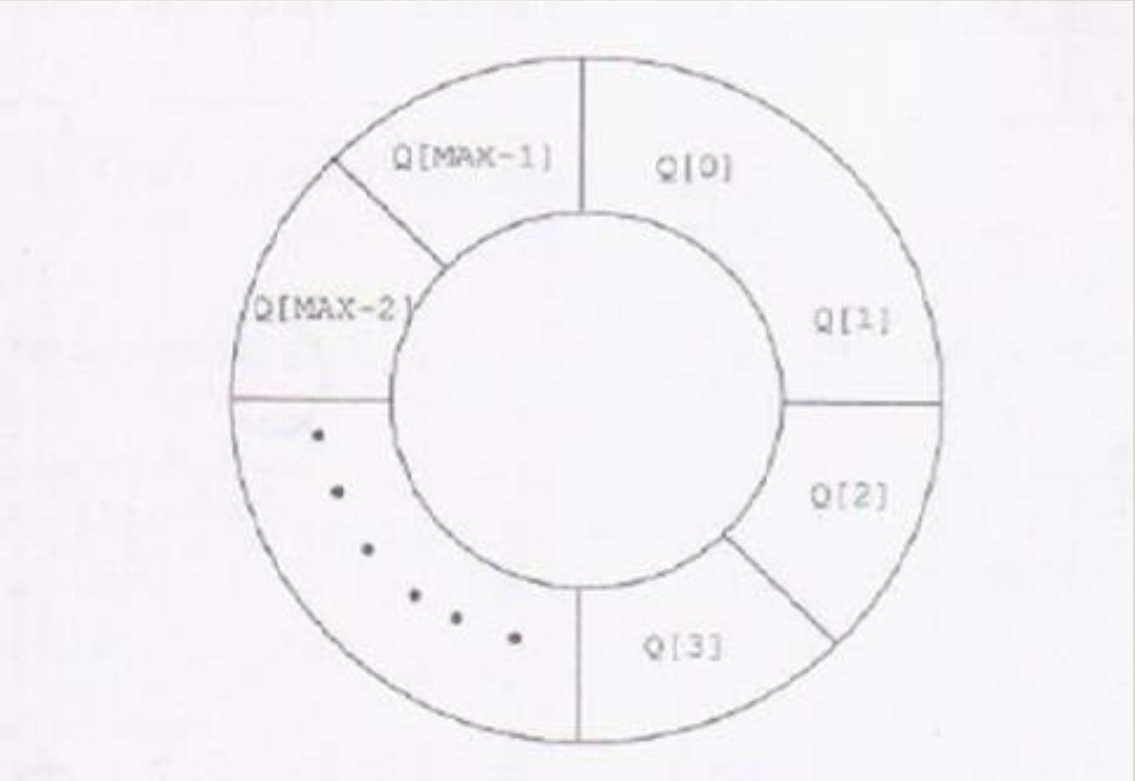
CIRCULAR QUEUE

- As discussed earlier, in case of queue represented as an array, once the value of the rear reaches the maximum size of the queue, no more elements can be inserted.
- However, there may be the possibility that space on the left of the front index is vacant. Hence, in spite of space on the left of front being empty, the queue is considered to be full.
- This wastage of space in the array implementation of a queue can be avoided by shifting the elements to the beginning of array if the space is available.
- In order to do this, the values of Rear and Front indices have to be changed accordingly. However, this is a complex process and is difficult to be implemented. An alternative solution to this problem is to implement a queue as a circular queue.



In this queue, front = 2 and rear = 9.
Now, if you want to insert a new element, it cannot be done because the space is available only at the left of the queue.

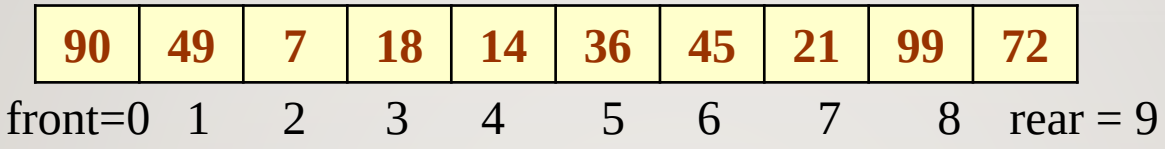
- The array implementation of circular queue is similar to the array implementation of queue. The only difference is that as soon as the rear index of the queue reaches the maximum size of the array, Rear is reset to the beginning of the queue provided it is free. The circular queue is full only when all the locations in the array are occupied.



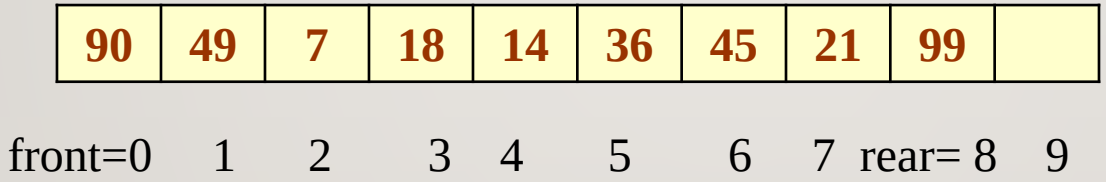
Inserting an Element in a Circular Queue

For insertion we check for three conditions which are as follows:

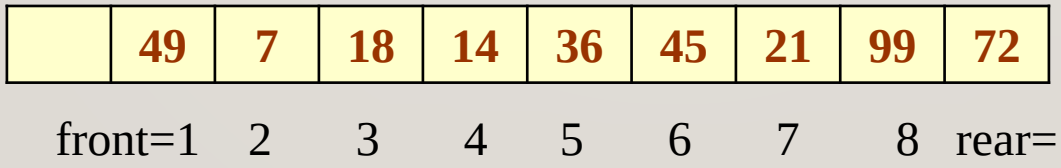
- If front=0 and rear= MAX – 1, then the circular queue is full.



- If rear != MAX – 1, then the rear will be incremented and value will be inserted



If front!=0 and rear=MAX -1, then it means that the queue is not full. So, set rear = 0 and insert the new element.



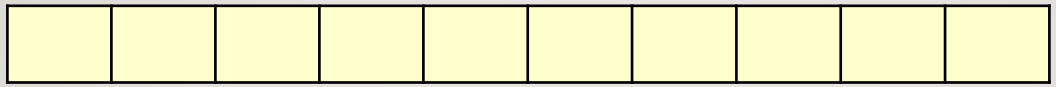
Insert an Element in a Circular Queue

```
Step 1:   IF (FRONT=0 && REAR=MAX–1) || (REAR=FRONT-1), then
            Write “OVERFLOW”
             Goto Step 4
           [END OF IF]
Step 2: IF FRONT = -1 and REAR = -1, then;
            SET FRONT = REAR = 0
         ELSE IF REAR = MAX – 1 and FRONT != 0
            SET REAR = 0
         ELSE
            SET REAR = REAR + 1
         [END OF IF]
Step 3: SET QUEUE[REAR] = VAL
Step 4: Exit
```

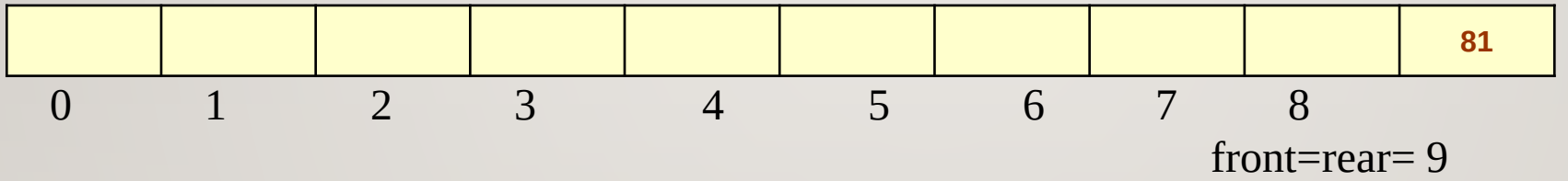
Deleting an Element from a Circular Queue

To delete an element again we will check for three conditions:

- If front = -1, then it means there are no elements in the queue. So an underflow condition will be reported.

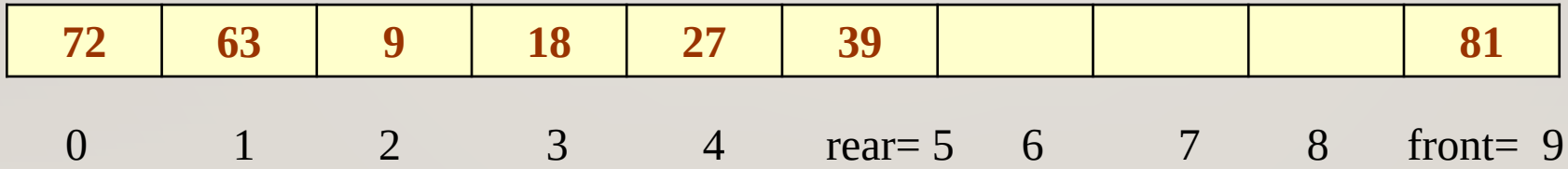


- If the queue is not empty and after returning the value on front, if front = rear, then it means now the queue has become empty and so front and rear are set to -1.



Delete this element and set rear = front = -1

- If the queue is not empty and after returning the value on front, if front = MAX -1, then front is set to 0.



Delete an Element from a Circular Queue

```
Step 1: IF FRONT = -1, then
            Write “Underflow”
            Goto Step 4
        [END OF IF]
Step 2: SET VAL = QUEUE[FRONT]
Step 3: IF FRONT = REAR
            SET FRONT = REAR = -1
        ELSE
            IF FRONT = MAX -1
                SET FRONT = 0
            ELSE
                SET FRONT = FRONT + 1
            [END OF IF]
        [END OF IF]
Step 4: EXIT
```