QUEUE ADT

Introduction

- ***A queue is a linear data structure in which the elements are added at one end called the rear and removed from the other end called the front.***
- ***A queue is called a FIFO (First-In, First-Out) data structure***
- We can explain the concept of queues using the following analogy:

  People moving on an escalator. The people who got

  on the escalator first will be the first one to step out of it.

Introduction

IMPLEMENTATION

A queue can be implemented by two methods

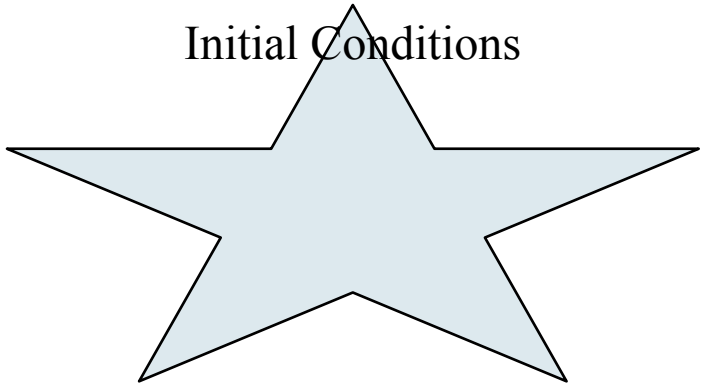
- ***Array Implementation of Queue***
- ***Linked Implementation of Queue***

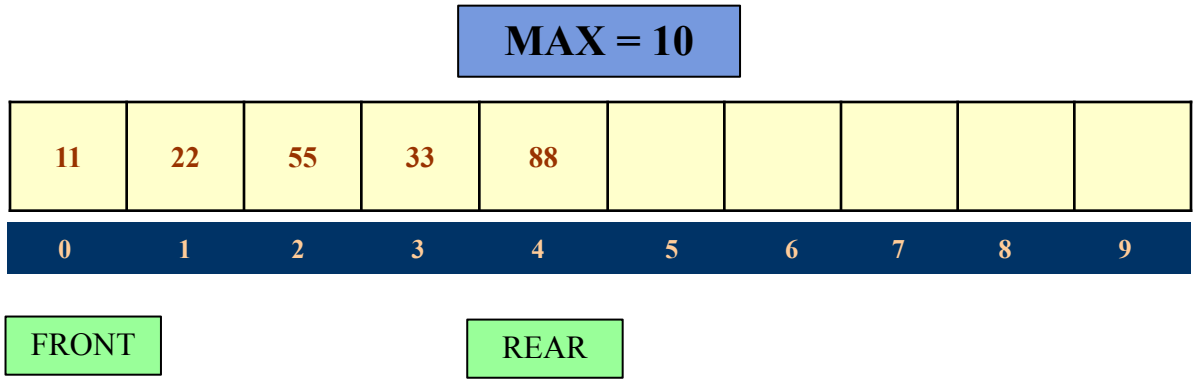
(also called Linked Queue)

Array Implementation of Queue

- Queues can be easily represented using linear arrays.
- Every queue has **FRONT** and **REAR** variables that point to the position from where **deletions and insertions** can be done, respectively

- **FRONT = REAR = -1** → **Initial Value**
- **FRONT = -1 OR FRONT > REAR** → **the queue is empty**
- **REAR = MAX-1** → **the queue is full**

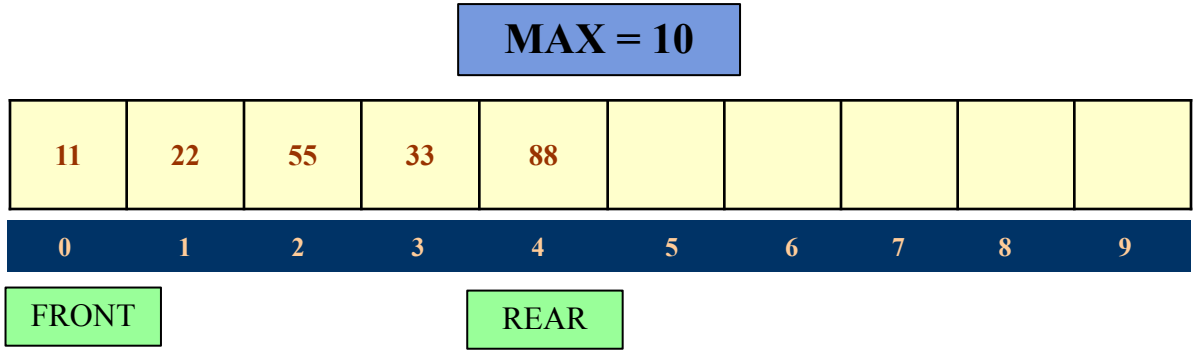




Algorithm for Insertion Operation

ALGORITHM TO INSERT AN ELEMENT IN A QUEUE

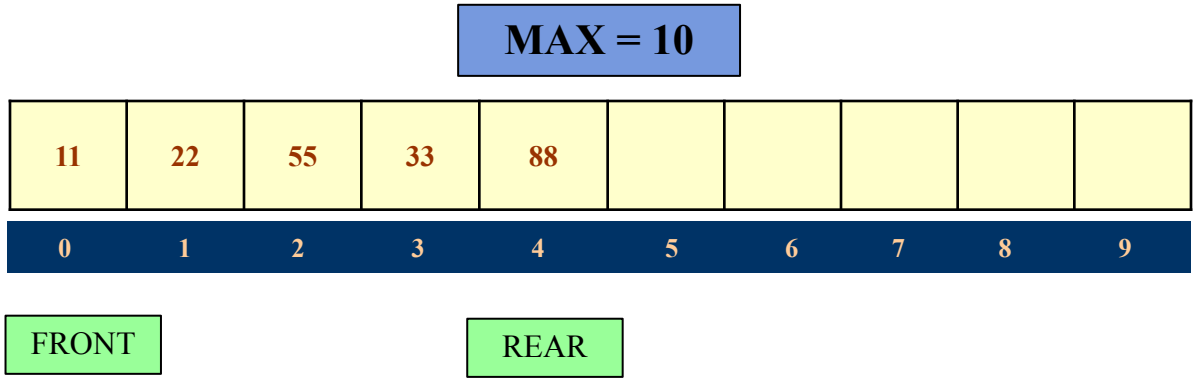
```
Step 1: IF REAR = MAX-1
          Write OVERFLOW
          Goto step 4
        [END OF IF]
Step 2: IF FRONT = -1 and REAR = -1
          SET FRONT = REAR = 0
        ELSE
          SET REAR = REAR + 1
        [END OF IF]
Step 3: SET QUEUE[REAR] = NUM
Step 4: EXIT
```



Algorithm for Deletion Operation

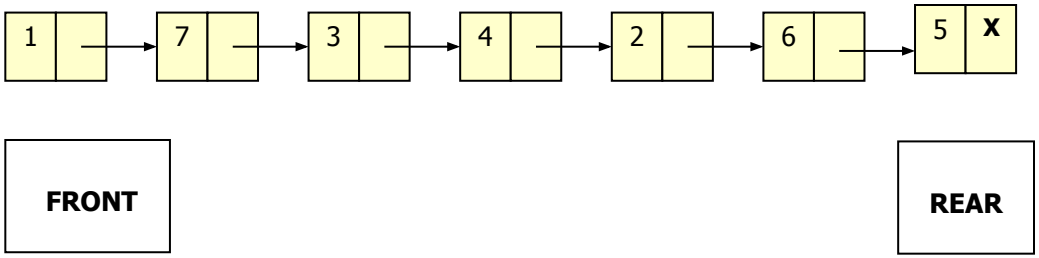
ALGORITHM TO DELETE AN ELEMENT FROM A QUEUE

```
Step 1: IF FRONT = -1 OR FRONT > REAR
         Write UNDERFLOW
        ELSE
         SET VAL = QUEUE[FRONT]
         SET FRONT = FRONT + 1
        [END OF IF]
Step 2: EXIT
```



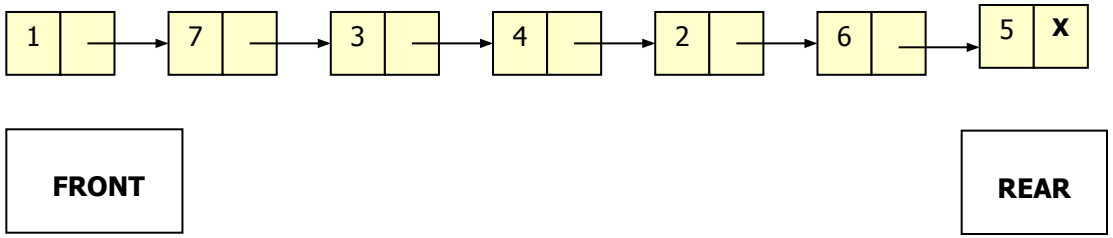
Linked Implementation of Queue (Linked Queue)

- In a linked queue, every element has two parts: one that stores data and the other that stores the address of the next element.
- The **START** pointer of the linked list is used as **FRONT**.
- Another pointer called **REAR** which will store the address of the last element in the queue.

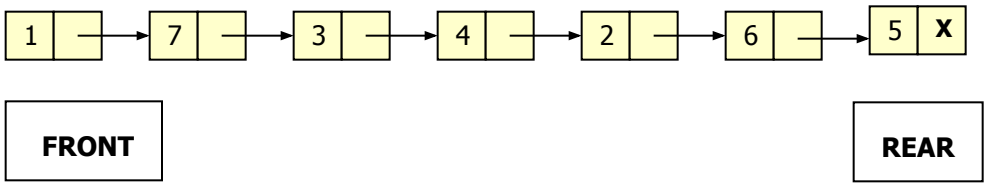


Linked Implementation of Queue (Linked Queue)

- **FRONT = REAR = NULL** – The queue is empty.
- All **insertions** will be done at the **rear end** and all the **deletions** will be done at the **front end**.
- **INSERTION** – Insert at end in Singly Linked List
- **DELETION** – Delete at beginning in Singly Linked List



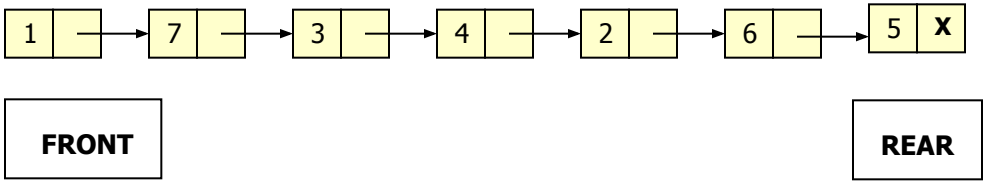
Inserting an Element in a Linked Queue



ALGORITHM TO INSERT AN ELEMENT IN A LINKED QUEUE

```
Step 1: IF AVAIL = NULL, then
         Write OVERFLOW
         Go to Step 6
     [END OF IF]
Step 2: SET New_Node = AVAIL
Step 3: SET New_Node -> DATA = VAL
Step 4: SET New_Node -> NEXT = NULL
Step 5: IF FRONT = NULL, then
         SET FRONT = REAR = New_Node
        ELSE
         SET REAR -> NEXT = New_Node
         SET REAR = New_Node
        [END OF IF]
Step 6: END
```

Deleting an Element from a Linked Queue



ALGORITHM TO DELETE AN ELEMENT FROM A LINKED QUEUE

```
Step 1: IF FRONT = NULL, then
         Write "Underflow"
         Go to Step 5
        [END OF IF]
Step 2: SET PTR = FRONT
Step 3: FRONT = FRONT->NEXT
Step 4: FREE PTR
Step 5: END
```