The Queues

A queue is logically a *first in first out (FIFO or first come first serve)* linear data structure. The concept of queue can be understood by our real life problems. For example a customer come and join in a queue to take the train ticket at the end (rear) and the ticket is issued from the front end of queue. That is, the customer who arrived first will receive the ticket first. It means the customers are serviced in the order in which they arrive at the service center.

It is a homogeneous collection of elements in which new elements are added at one end called *rear*, and the existing elements are deleted from other end called *front*.

The basic operations that can be performed on queue are

1. Insert (or add) an element to the queue (push)
2. Delete (or remove) an element from a queue (pop)

Push operation will insert (or add) an element to queue, at the rear end, by incrementing the array index. Pop operation will delete (or remove) from the front end by decrementing the array index and will assign the deleted value to a variable. Total number of elements present in the queue is **(rear-front)+1**, when implemented using arrays. Following figure will illustrate the basic operations on queue.

Fig. 4.1. Queue is empty.

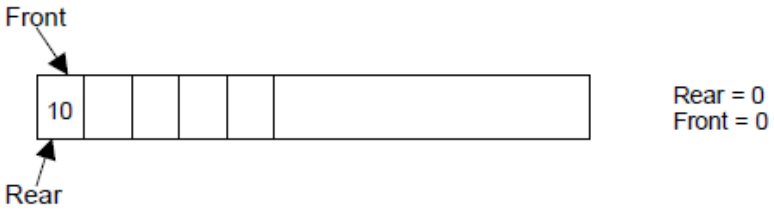

Fig. 4.2. push(10)

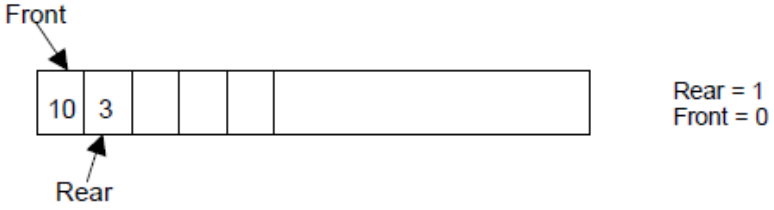

Fig. 4.3. push(3)

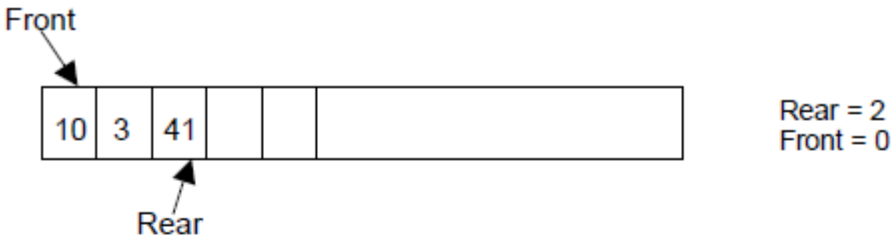

Fig. 4.4. push(41)

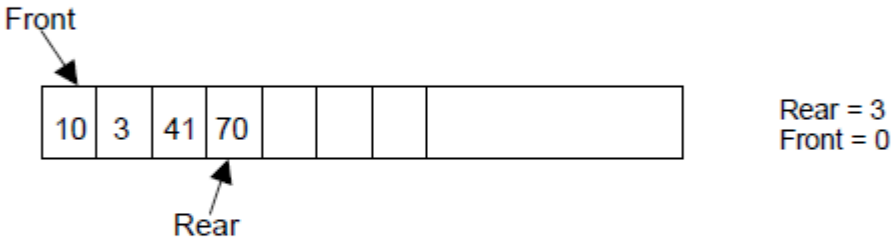

Fig. 4.5. push(70)

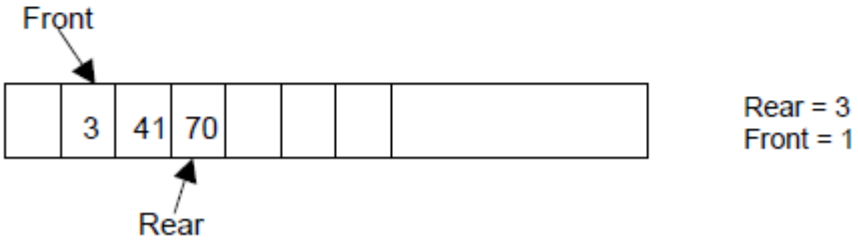

Fig. 4.6. x = pop() (*i.e.*; x = 10)

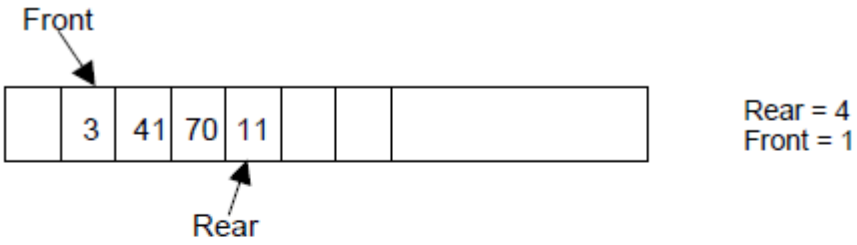

Fig. 4.7. push(11)

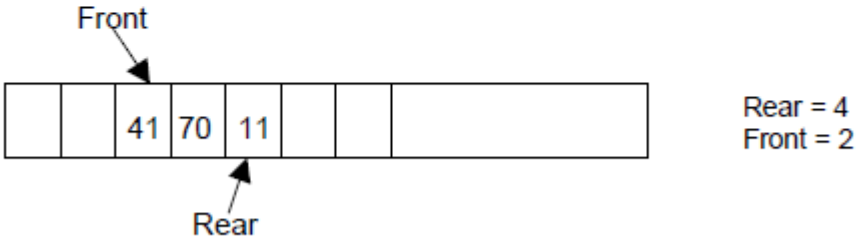

Fig. 4.8. x = pop() (*i.e.*; x = 3)

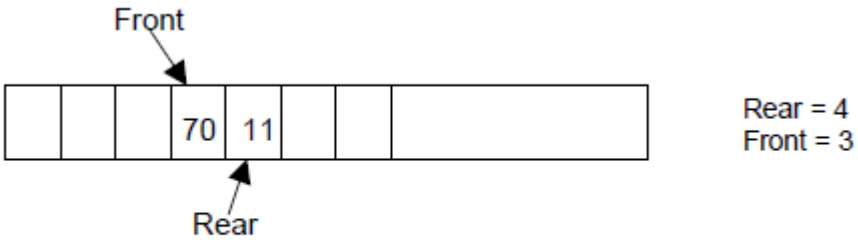

Fig. 4.9. x = pop() (*i.e.*, x = 41)

Queue can be implemented in two ways:

1. Using arrays (static)
2. Using pointers (Linked List) (dynamic)

Implementation of queue using pointers will be discussed later. Let us discuss underflow and overflow conditions when a queue is implemented using arrays.

If we try to pop (or delete or remove) an element from queue when it is empty, underflow occurs. It is not possible to delete (or take out) any element when there is no element in the queue.

Suppose maximum size of the queue (when it is implemented using arrays) is 50. If we try to push (or insert or add) an element to queue, overflow occurs. When queue is full it is naturally not possible to insert any more elements.

ALGORITHM FOR QUEUE OPERATIONS

Let Q be the array of some specified size say SIZE

1- INSERTING AN ELEMENT INTO THE QUEUE

1. Initialize front= –1, rear = –1
2. Input the value to be inserted and assign to variable "data"
3. If (rear = = SIZE-1)
   (*a*) Display "Queue overflow"
   (*b*) Exit
4. Else
   (*a*) Rear = rear +1
5. Q[rear] = data
6. Exit

2- DELETING AN ELEMENT FROM QUEUE

1. If (rear< front) or (front and rear is equal to -1)
    (a) Front = –1, rear = –1
    (b) Display “The queue is empty”
    (c) Exit
2. Else
    (a) Data = Q[front]
3. Front = front +1
4. Exit

3- DISPLAY THE ELEMENTS OF QUEUE

1. If (rear< front) or (front and rear is equal to -1)
    (a) Display “The queue is empty”
    (b) Exit
2. Else
    (a) i=front to rear
        (1) display Queue[i]
    (b) Exit
3. Exit