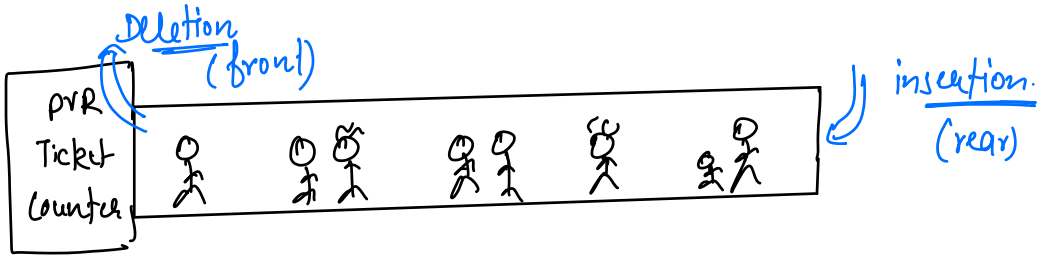



FIFO - [First In First Out]

- Movie Ticket / Car wash
- Call centre
- Printer.
- Job scheduler.
- Message Queue.

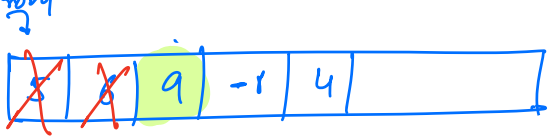
Operations by Queue.

O(1)

- Enqueue(x) → insert the data at rear end.
- Dequeue() → remove data from front.
- front() → data present at front end.
- size() → no. of elements in the queue.

Enq(5) Enq(6) Enq(9) Enq(-1) Deq() Deq() front() Enq(4)

5. 6. 9.

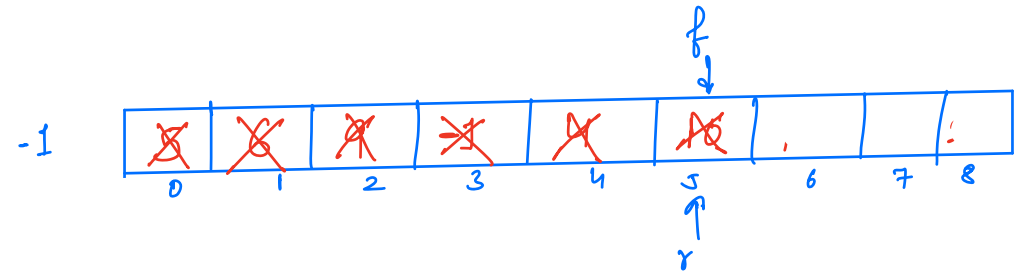


# Implementation Of Queues

Arrays:

f = -1 → idx of last element which was just removed

r = -1 → idx of last element which was just inserted



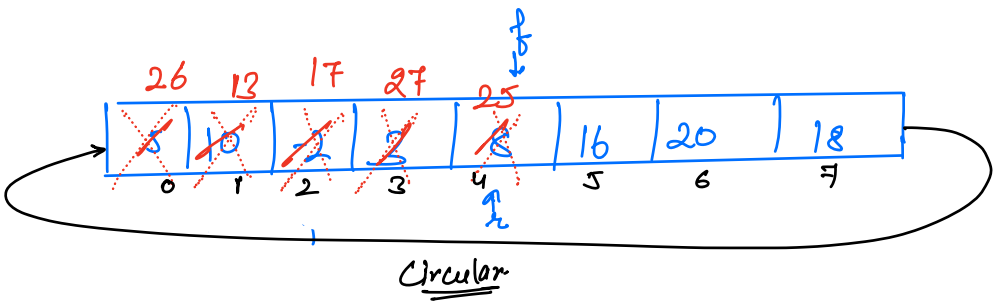

Enq(5) Enq(6) Enq(9) Enq(-1) Deq() Deq() Enq(4)

```
Enqueue(x){
    if(r==N-1){//Queue is full}
        r++
        arr[r]=x;
}
```

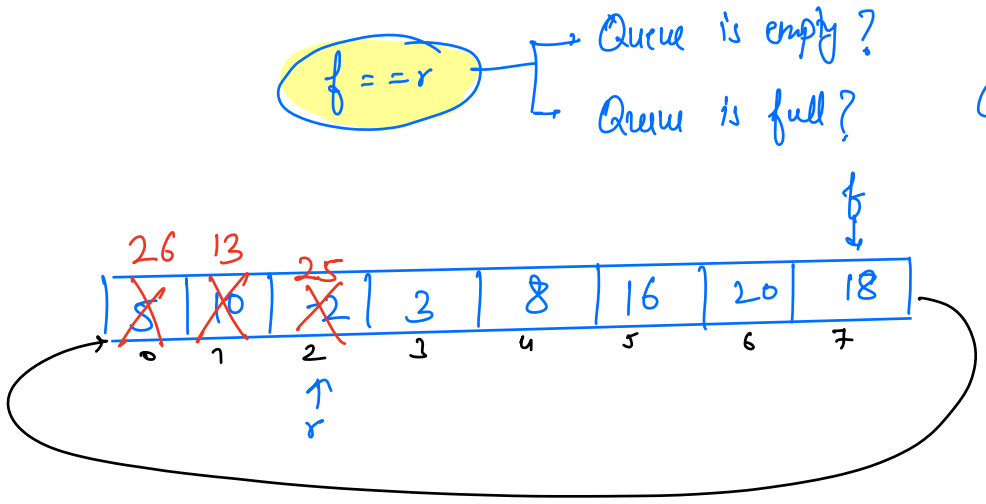
```
Dequeue(){
    if(f==r){//Queue is Empty}
    f=f+1;
}
```

```
front(){
    if(f==r){//Queue is Empty}
    return arr[f+1];
}
```

Eg:-




Circular




enq(26)
enq(13)
enq(25)
enq(-5);
Deq()
Deq()
Deq()
Deq()
Deq()
Deq()

```
Enqueue(x){
    if (size == capacity) {//Queue is full}
      r = (r+1) % N;
    arr[r] = x;
      size++
}
```

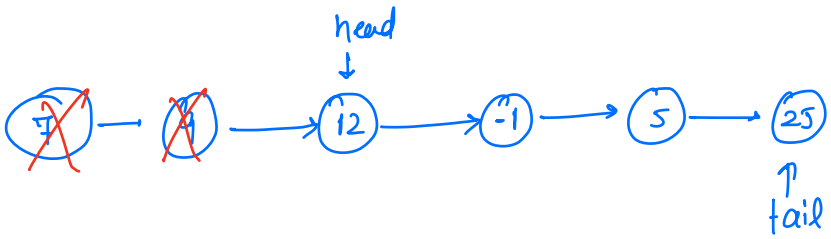
```
Dequeue(){
    if (size==0) { //Queue is Empty }
      f=(f+1) % N
      size--
}
```

```
front(){
    if (size==0) { //Queue is Empty }
    return arr[(f+1) % N];
}
```

→ Implementation Using Linked-list

→ Enqueue insertion → tail / head

→ Dequeue deletion → tail / head.
↓
O(N)




insertion

```
Node n = new Node(5);
tail.next = n;
tail = tail.next;
```

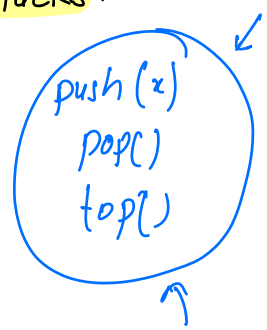
Enq(5)
Enq(25)

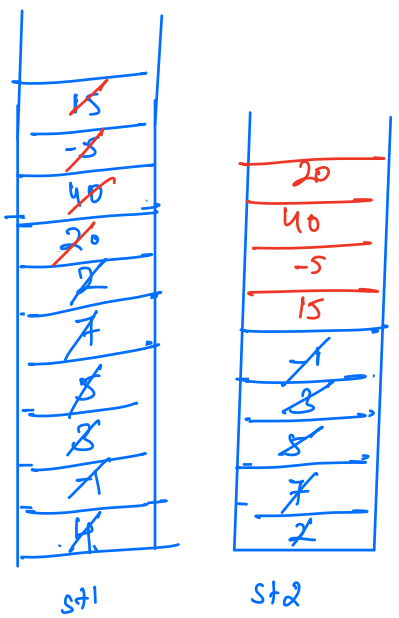
deletion

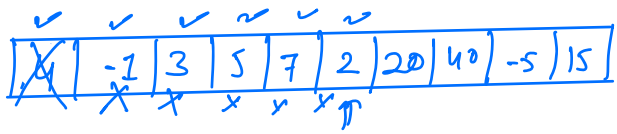
```
head = head.next
```

Q) Implement queue using stacks.

O(1) { enqueue()
dequeue()
front()




deq() → 4.
enq(20);
enq(40)
enq(-5);
enq(15)
deq() → -1
deq() → 3
deq() → 5
deq() → 7
deq() → 2
deq() → 20.

N dequ → 2N+1.

1 deq → $\frac{2N+1}{N}$ ⇒ 2. - O(1)

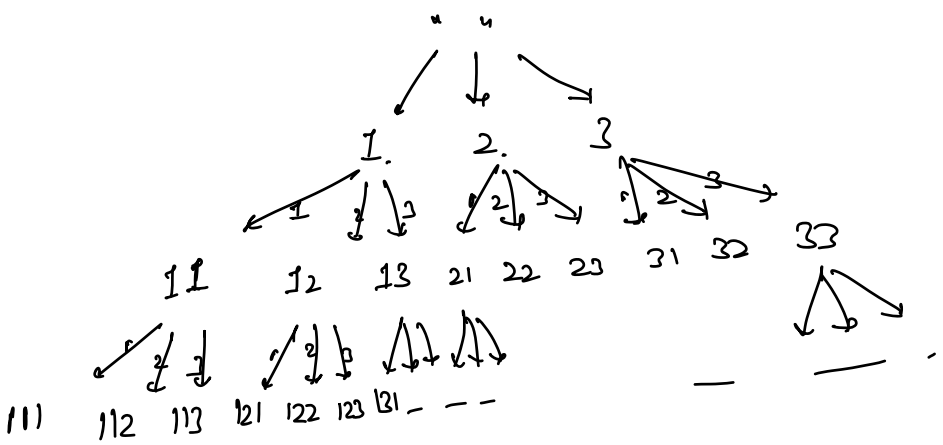
O(1) { Enqueue(x) → push x in st1
O(1) in any case. { Dequeue() → if | st2.size() == 0)
yes → transfer everything from st1 to st2 and then st2.pop()
no → pop from st2.

→ Nth number

1, 2, 3.

N=1 1
N=2 2
N=3 3
N=4 1 1
N=5 1 2
N=6 1 3
2 1
2 2
2 3
3 1
3 2
3 3
1 1 1
1 1 2
1 1 3
1 2 1
1 2 2
1 2 3

[ N=13, ans = 111
N=17, ans = 122
N=6, ans = 13 ]




N=17

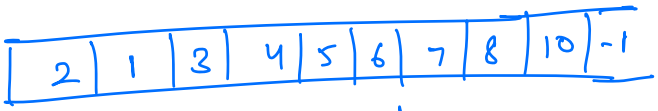
| 1 | 2 | 3 | 11 | 12 | 13 | 21 | 22 | 23 | 31 | 32 | 33 | 111 | 112 | 113 | 121 | 122 |

count = 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

```
Queue<int> q;
    q.enqueue(1);
    q.enqueue(2);
    q.enqueue(3);
    while(count <= N){
        x = q.front()
        q.dequeue()
        q.enqueue(x*10+1);
        q.enqueue(x*10+2);
        q.enqueue(x*10+3);
    }
```

T.C → O(N)
S.C → O(N)

- Reverse a Queue.

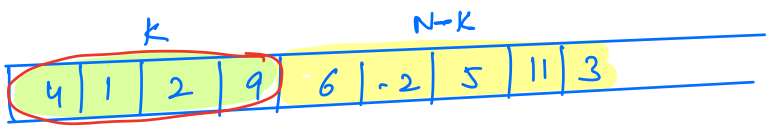


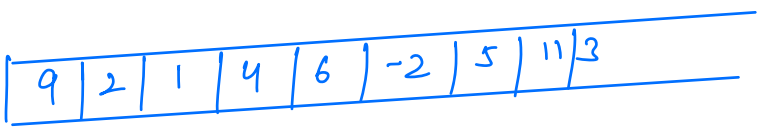

-1 10 8 7 6 5 4 3 1 2,

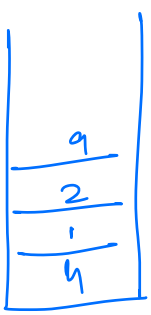
reverse()


(K)

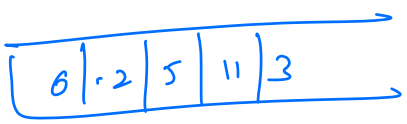
[Reverse first 4 elements of the queue.



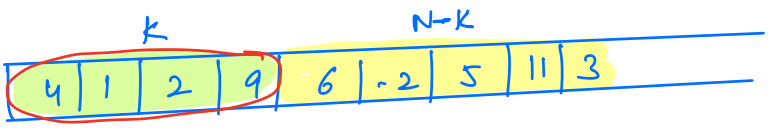

qu- 

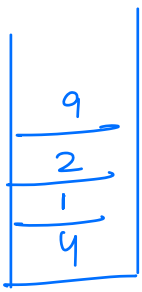

1 stack

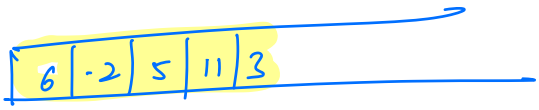



1 Queue.




① push first K elements in the stack




② pop all elements from stack & enqueue them in the same queue.


③ Dequeue (5) n-K elements from the queue & enqueue them in the same queue.