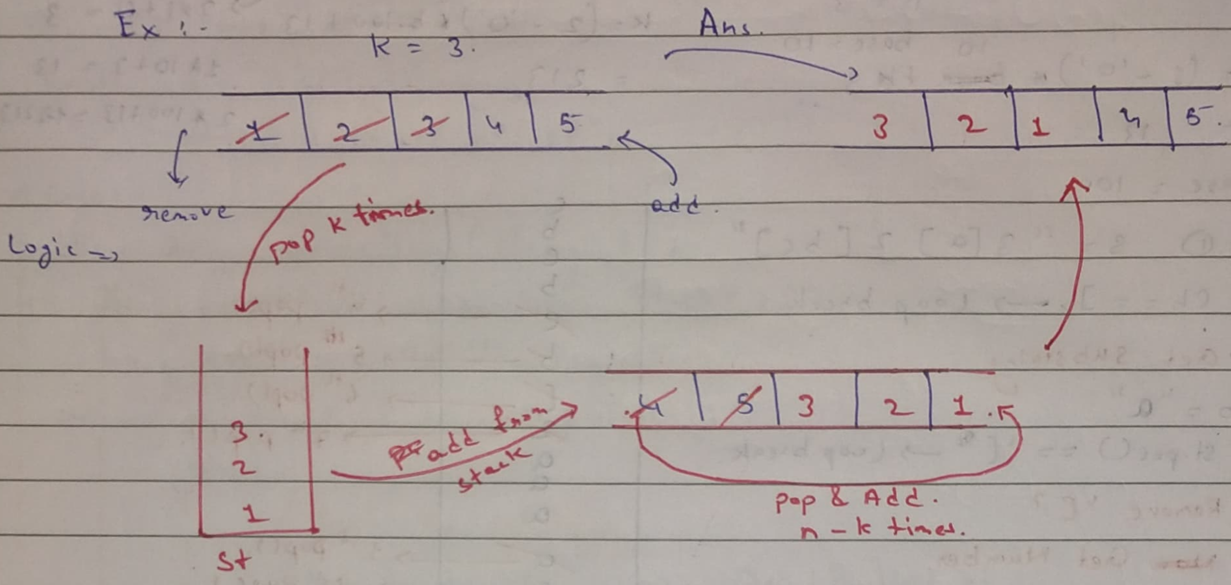
Queue.

① Reverse first k elements of a queue.

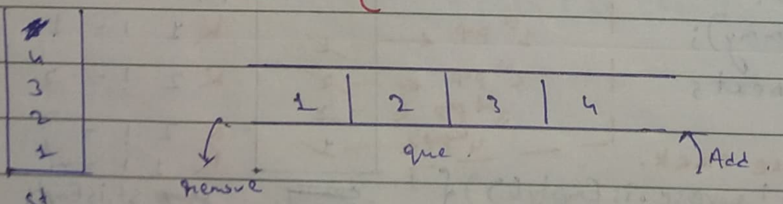
Ex :-



⇒ Code :-

```
public static void ReverseKEle( que ~~?~~, int k){
Stack<Integer> st = new stack<>();

    for (int i = 0 ; i<k ; i++){
        st.push (que.remove());
    }

    while (st.size() > 0){
        que.add (st.pop();
    }
    for (int i = 0 ; i< que.size() -k ; i++){
        que.add(que.remove());
    }
}.
```

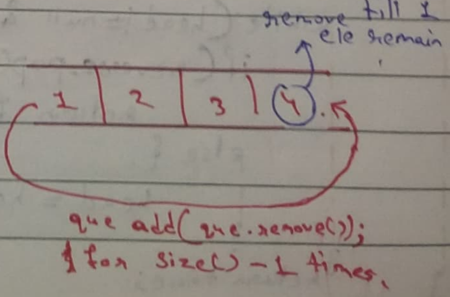
② Leet Code 225 { Implement Stack Using Queues }



```
~~void~~ push() {
    que.add(val);
}
```

```
peek() {
    for(i=0 to i< que.size-1){
        que.add(que.remove());
    }
    int val = que.peek();
    que.add(que.remove());
}
```

pop() is also same.



que add(que.remove());
for size() - 1 times.

#Method - 1 (Push efficient).

```
=> Code :- Class My Stack {
   Queue<Integer> que = new LinkedList<>();
   public void push( int val) {
       que.add ( val);
   }
                                          public boolean empty(){
                                              return (que.size()==0);
                                          }
   public int Pop () {
       for (int i = 0 ; i<que.size()-1 ; i++){
           que.add(que.remove ());
       }
       int val = que.remove();
       return val;
   }
                                          public int top() {
                                          ~~publi~~ for(int i = 0 ; i< que size()-1 ; i++){
                                                  que.add(que.remove());
                                              }
                                              int val = que.peek();
                                              que. add(que.remove());
                                              return val;
                                          }
```

-> T.C :- push => O(1)
pop => O(n)
top => O(n).

S.C :- O(n).(que).

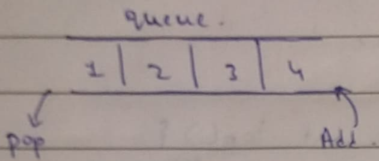
#Method - 2 (Pop & top efficient).

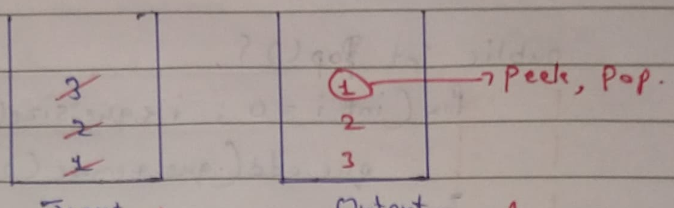
```
=> Code :- Class MyStack {
   Queue<Integer> q = new ~~My~~ LinkedList<>();
   ~~p~~
   public void push(int val){
       if(q.size()==0) q.add(val);
       else {
           q.add (val)
           for(int i = 0 ; i<q.size()-1 ; i++){
               q.add (q.remove());
           }
       }
   }
                                          public int top(){
                                              return q. peek();
                                          }
   public int pop(){
       return q.remove();
   }
                                          public boolean empty(){
                                              return (q.size()==0);
                                          }
}
```

③ LeetCode Q.No. 232 { Implement Queue Using Stacks. }

→ In this also you ~~m~~ can write push & or ~~po~~ pop efficient code.

⇒ Best Approach $\xrightarrow{for}$ all operation ⟶ T.C ⇒ O(n).





Input → for push

Output → for peek & pop.

```
push (x) {
    input.push(x)
}
```

```
peek ( ) {
    if (Output.isEmpty()){
        while(!input.isEmpty()){
            output.push(input.pop());
        }
    }
    return output.peek();
}
```

⇒ Code :-

```
public Class MyQueue {
    Stack<Integer> input = new Stack<>();
    Stack<Integer> Output = new Stack<>();

    public void push (int x){
        input.push(x);
    }

    public int pop(){
        peek();
        return output.pop();
    }

    public int peek(){
        if(output.isEmpty()){
            while(!input.isEmpty()){
                output.push(input.pop());
            }
        }
        return output.peek();
    }

    public boolean empty(){
        return input.isEmpty() && output.isEmpty();
    }
```

④ GfG ~~Leetcode~~ { First Negative in Each window of Size 'k'. } → Using Extra Space Queue

Ex ①

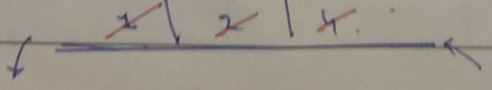
0 ① ② 3 ④ 5 6 7

arr = { 12, -1, -7, 8, -15, 30, 16, 28 }

k = 3, Ans → res = { -1, -1, -7, -15, -15, 0 } — (n-k+1)

if -ve push.

q =



Ex ②:-

0 ① ② 3 4 5 ⑥ 7

arr = { 12, -1, -7, 8, 15, 30, -16, 28 }

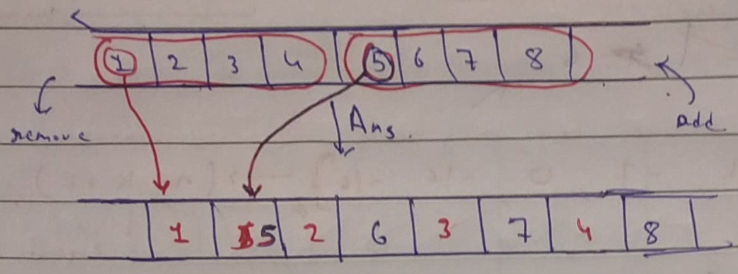
k = 3.

q = ~~1~~ | ~~2~~ | ~~#~~ 6

res = { -1, -1, -7, 0, -16, -16 } → (n-k+1)

```
=> Code :- psvm() {
    int[] A = { 12, -1, -7, 8, -15, 30, 16, 28 };
    ink k = 3;
    int n = A.Length;

    int[] res = new int[n-k+1];
    Queue<Integer> q = new LinkedList<>();

    for(int i = 0; i<n; i++){
        if(A[i] < 0){
            q.add(i);
        }
    }

    for(int i = 0; i<n-k+1; i++){
        if(q.size() > 0 && q.peek() < i)
            q.remove();
        if(q.size() > 0 && q.peek() <= i+k-1){
            res[i] = A[q.peek()];
        }
        else if (q.size() == 0)
            res[i] = 0;
        else
            res[i] = 0;
    }
    for(int ele : res){
        Sout(ele + " ");
    }
}
```

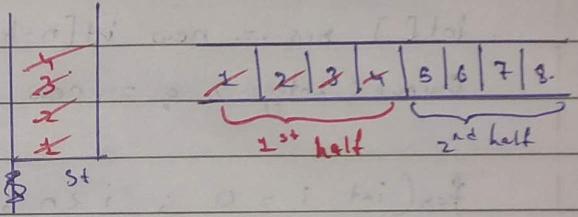
⑤ Gfg { Reorder Queue (interleave 1st half with 2nd half). } → size of Queue will be even
→ using stack



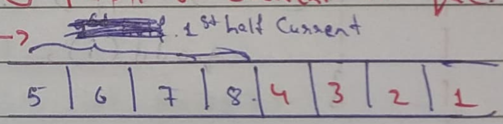
=> 5 steps.

```
=> Code :- p.s v { Reorder Queue ( que ) {
    Stack <Integer> st = new Stack< >();
    int n = que.size()/2;
    // Step - 1
    for(int i=0; i<n; i++){
        st.push(que.poll());
    }
    // Step - 2
    while(!st.isEmpty()){
        que.add(st.pop());
    }
    // Step - 3
    for(int i=0; i<n; i++){
        que.add(que.poll());
    }
    // Step - 4
    for(int i=0; i<n; i++){
        st.push(que.poll());
    }
    // Step - 5
    while(!st.isEmpty()){
        que.add(st.pop());
        que.add(que.remove());
    }
    Sout(que);
}
```

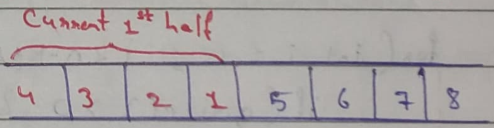
① push 1st half of que to stack



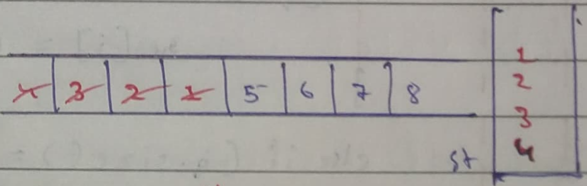
② Push st element to que.



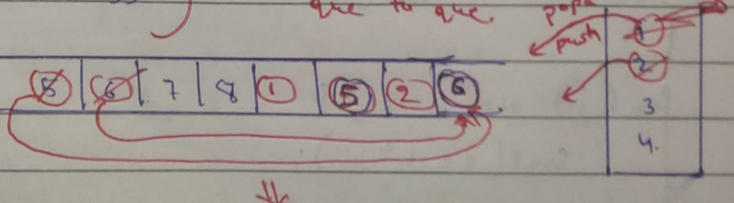
③ Push Current 1st half to que itself



④ Push Current 1st half to stack.



⑤ Now 1st push 1st ele from Stack to que.
& then push 1st ele from que to que.



Ans :- que => 1 | 5 | 2 | 6 | 3 | 7 | 4 | 8

⑥ LeetCode Q. No. (1700) { Number of Students Unable to Eat Lunch }

Ex ① :-

students Preference

Circular Sandwiches

Square Sandwiches

Students = {(1), 1, 0, 0}   Sandwiches = {(0), 1, 0, 1}.

→ preference Not match Student will Go back.

{(1), 0, 0, 1}   {(0), 1, 0, 1} Not Match

{~~(0)~~, (0), 1, 1}   {~~(0)~~, (1), 0, 1} Match Student will take Sandwich

{~~(1)~~, (1), 0}   {~~(1)~~, (0), 1} 1st Match, 2nd Not match

{~~0~~, ~~1~~}   {~~0~~, ~~1~~} 1st Match, 2nd Match.

Ans => 0//.

Ex ② :-

students = {1, 1, 1, 0, 0, 1}   Sandwiches = {~~1~~, ~~0~~, ~~0~~, 0, 1, 1}.

Logic :- Count :

Ones :- ~~4~~ 3

Zeroes :- ~~2~~ ~~1~~ 0.

if(Ones == 0) → return Zeroes.

if(zeroes == 0) → return Ones. ✓

if(sandwiches == 1)
Ones --;

if(sandwiches == 0)
Zeroes --;

Ans => (3)// → These Students will Not able to Eat Lunch//.

=> Code :-

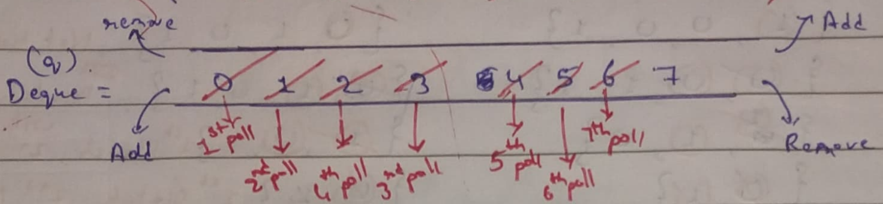
```
public int CountStudents(int[] students, int[] Sandwiches){
    int Ones = 0;  int Zeroes = 0;
    for(int stud : students){
        if(stud == 0) zeroes++;        } → Count Zeroes & Ones
        else Ones++;
    }
    for(int sandwich : sandwiches){
        if(sandwich == 0){
            if(zeroes == 0)
                return Ones;
            else
                Zeroes--;
        }
        else if(Sandwich == 1){
            if(Ones == 0)
                return Zeroes;
            else
                Ones--;
        }
    } return 0;
}
```

(7) LeetCode (239) { Sliding Window Maximum }

0 1 2 3 4 5 6 7

(k=3) nums = { 1, 3, -1, -3, 5, 3, 6, 7 }.

res = { 3, 3, 5, 5, 6, 7 }



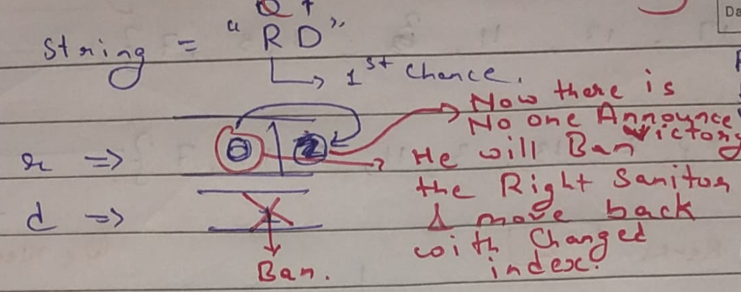
⟹ Code :-

```
public int[] maxSlidingWindw( int[] nums, int k){.
if(nums == null || k <= 0)
        return new int[0] ;
int n = nums.length ;
int[] res = new int[n-k+1];
int resi = 0;

Deque<Integer> q = new Arrray Deque<>();

for(int i = 0 ; i<n ; i++){
    while(!q.isEmpty() && q.peek() < i-k+1) {
        q.poll();
    }


    while (!q.isEmpty() && nums[q.peeklast()] < nums(i])
        q.pollLast();
    q.offer(i);
    if(i >= k-1){
        res [resi++] = nums [q.peek()];
    }
}

return res ;
}.
```

(8) LeetCode Q. No. (649) { Dota 2 Senate. }

Ex (1) => String = "RD"

↳ 1st Chance.


Q ↑
r =>
0
2
d =>
Ban.
Now there is No one Announce Victory
He will Ban the Right Sanitor & move back with Changed index.

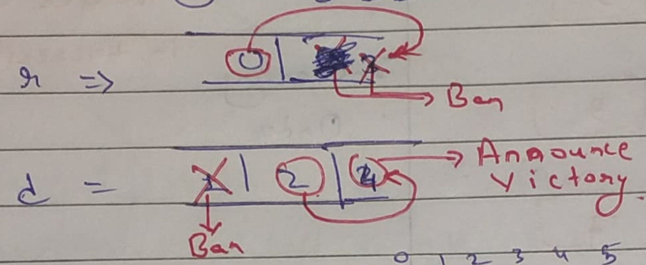
R = Radiant
D = Dire

Has 1 power can perform 1 operation ~~out of 2~~.

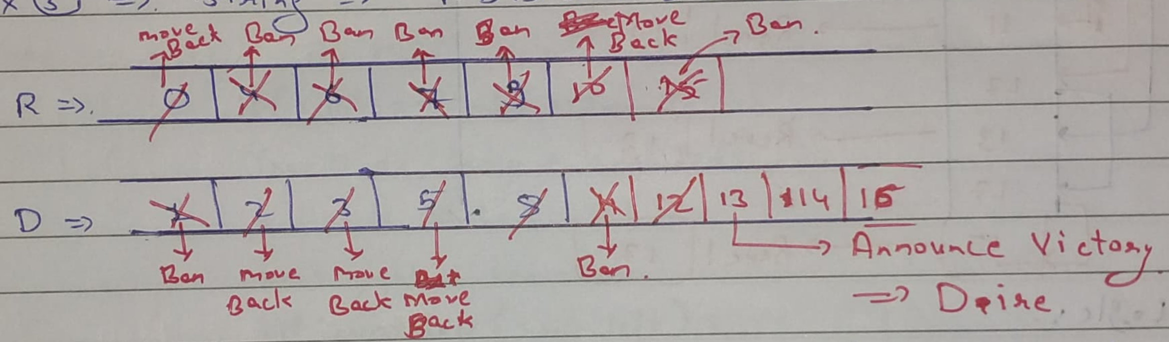
(1) Ban One Senator's Right

(2) Announce the Victory.

Ex (2) => string = "RDD"
0 1 2.



Ex (3) => string => "RDDDRDRRDR" [n = 10]
0 1 2 3 4 5 6 7 8 9



=> Code :-

```
public string predictPaityVictory(string senate){
    Queue<Integer> r = new LinkedList<>();
    Queue<Integer> d = new linkedList<>();
    int n = senate.length();
    for(int i = 0; i<n; i++){
            if(senate.charAt(i)=='R')
                    r.add(i);
            else
                    d.add(i);
    }
    while(!r.isEmpty() && !d.isEmpty()){
            if(r.peek() < d.peek()){
                    r.add(n++);
            }
            else
                    d.add(n++);
            r.remove();
            d.remove();
    }
    return r.isEmpty() ? "Dire" : "Radient";
}
```

(9) LeetCode Q. No. (950) { Reveal Cards in Increasing Order. }

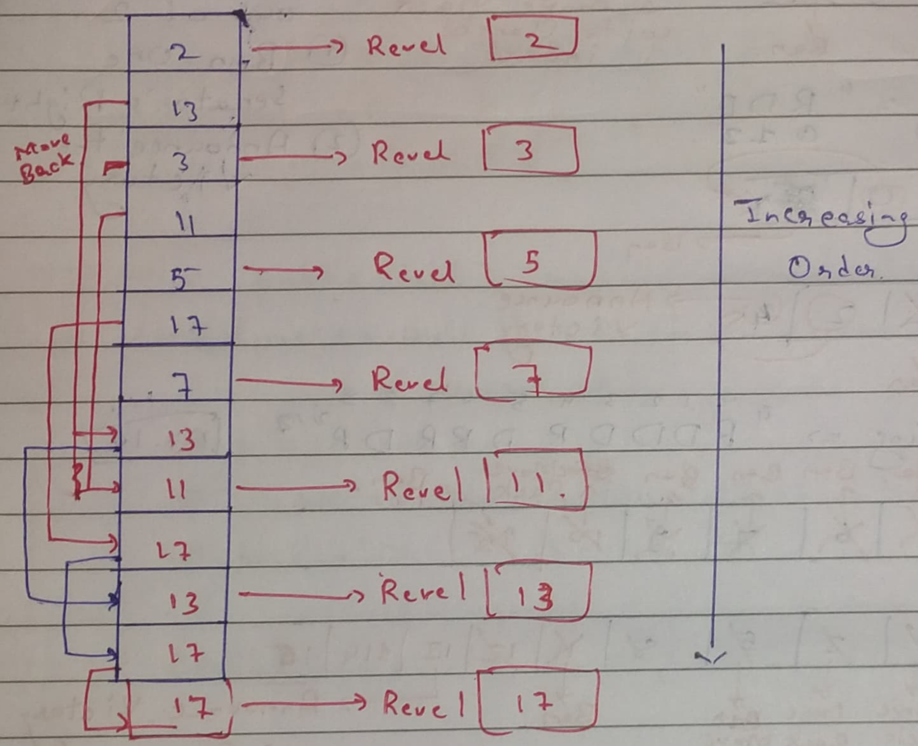
Ex (1)

deck = { 17, 13, 11, 2, 3, 5, 7 }.

Ans :-

↳ res = { 2, 13, 3, 11, 5, 17, 7 }.

We Need to find the Sequence in which Reveled card ~~is~~ are in Increasing Order



Logic :-

deck = { 17, 13, 11, 2, 3, 5, 7 }.

sort ↳ = { 2, 3, 5, 7, 11, 13, 17 }.

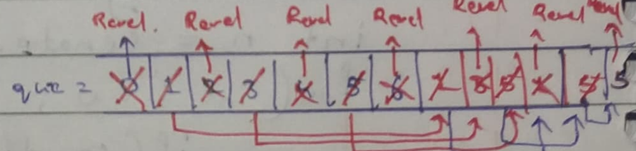
(i)

Cread res[] & que.

0	1	2	3	4	5	6
2	13	3	11	5	17	7

res =

que =



revel = res[q.poll] = deck(i).

move back.

```
=> Code :- public int[] deckRevealedIncreasing (int[] deck){
|          int n = deck.length;
|          Arrays.sort(deck);
|          Queue<Integer> q = new LinkedList< >();
|          for(int i=0; i<n; i++){
|                    q.add(i);
|          ~~int[] res = new int[n];~~
|          for(int i=0; i<n; i++){
|          |        res[q.poll()] = deck[i];
|          |        q.add(q.poll());
|          }
}
           return res }
```