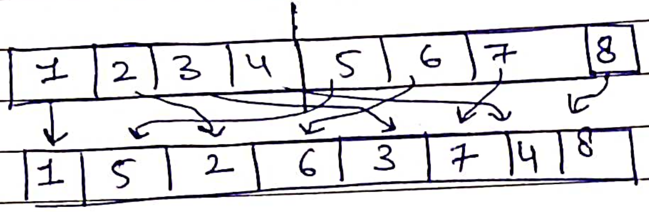
#QUEUE hai bhai.

Interleave first Half of Queue with second-half.

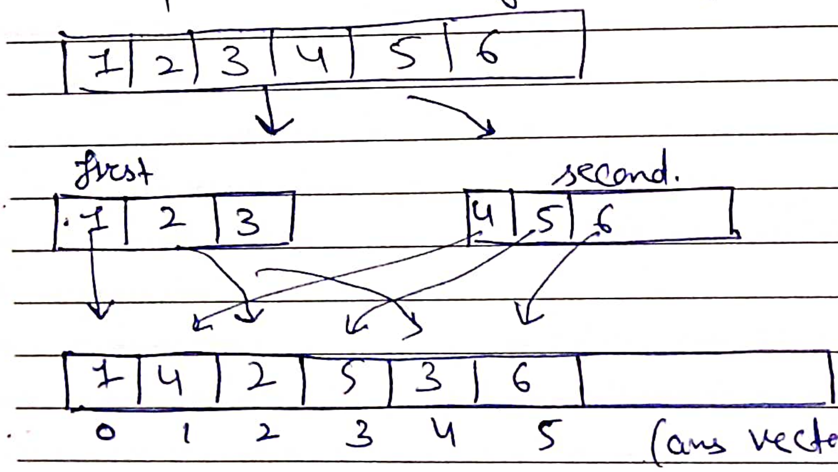
given

expected



basically divide the queue into two queues first and second.

and in ~~first~~ in ~~odd~~ even indices push elements from first queue. and push elements of second queue in odd indices.



(ans vector).

First- Negative integer in Every-window of size k.

Input Array:- −8, 2, 3, −6, 10. | k = 2

Desired Output → −8, 0, −6, −6.

Approach → Simple asf.

1.) Take a queue & put upto first 'k' negative numbers.

i) | −8 | | iteration 1. (iterated −8, 2).

actually don't put the value put index instead.

ii) | 0 | Queue |

→ Now start from kth index & run following commands till end of arr.

int 'y = 0 (if queqe is empty) or 'y = arr[q.front()] if it's not

→ In our test case queue isn't empty, so.
we'll push arr[q.front()] to our answer.

| −8 | ans vector. |

→ Now, if Queque isn't empty & (q.front() <= i − k)
pop q.front.

→ How i got this formula?

2-0-2-2

-8	2	3	-6	10
0	1	2	3	4

↳ see, Here we are making windows of size k (2)

Suppose q.front() = 0 (i.e -8) and i is pointing to index 2 (i.e 3).

we can see, the negative no. is out of the window
2-0 = 2, i.e <= k, so we pop ~~fo~~ it from queue.

~~and~~, So, our ans vector now is

-8	0	

Now if arr[i] is a negative integer too store it's ~~val~~ index in queue.

at index 3.

3	

→

-8	0	-6	-6

~~So~~ in, the end there'll be required Comparisons for last elemen.

int j = 0(if q isempty), q.front() if it's not.
ans.push_back(j);

return ans;