Note

Reveal Cards In Increasing Order.

Q) You are given a deck of cards → deck = [17, 13, 11, 2, 3, 5, 7]
Now there is a certain order in which the deck of cards are getting pick

Step1: Pick the card on the top, reveal it and take it out

Step2: After step 1 if there are more cards left, then pick the top card and put it at the bottom of the deck

Step3: If deck is not empty. Redo from Step 1.

You have to order the cards in such a fashion, so that the cards picked up in the above order, always reveals the cards in increasing order.

[2, 13, 3, 11, 5, 17, 7] → 2 is revealed, 13 is skipped.
3 is revealed, 11 is skipped
and so on . . .

[2, 3, 5, 7, 11, 13, 17] → order of revealing

(A) How I did it?

I took a normal array of indexes already sorted.

[0, 1, 2, 3, 4, 5, 6, . . .] → Now

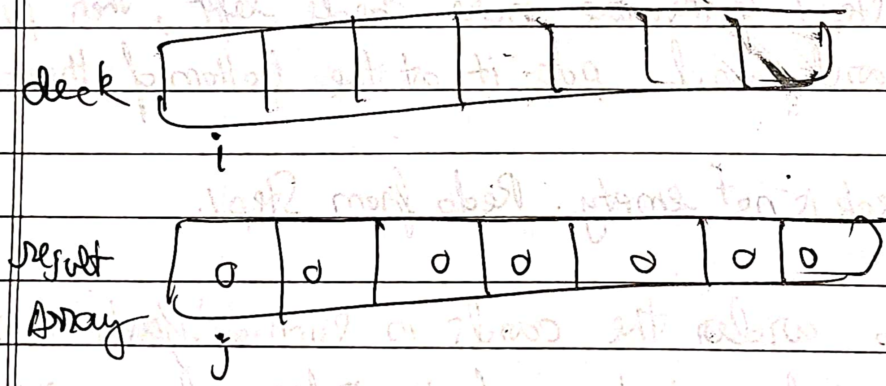
Now I replicated the behaviour using a Queue and marked which index is getting ~~pop~~ popped when.

You get ~~get~~ the sorted result

```
while ( !q.isNotEmpty) {
    // Reveal top element
    // pop

    // move next element to
       back of the queue
}.
```

Solution, which I found on Leetcode?



1) Move (i=i+1). and j = (j+2) % N

2) ~~if~~ populate result[j] = ans[i]

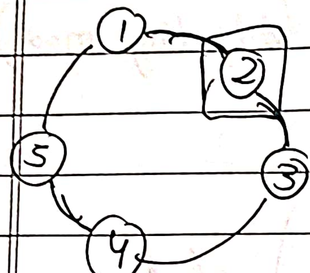
3) if (result[j] != 0) that means its populated, then do j++.

(Note)

(Find the winner of the Circular Game)

Q) Question is Simllar to Russian Roulette.

(N = 5, K = 2)



At each step.
the Kth Alive ~~guy~~ is to be eliminated.

* Start from ① go to Kth node.
Eliminate K.

* Then Start from next Allive Person and go the Kth Alive person in the circle and eliminate

* Do this till only 1 person is alive.
Find the winner.

A) Intuition: Recursion.

Lets say. for. a circle of N persons. I have to find the winner.

[1, 2, 3, 4, 5, 6, 7, 8, ... N]

Lets say I have a magic function f( ).

To which if I pass the in~~dex~~ ~~of~~ (the number of people = N & K)

then it tells me the index of the winner.

So for N people.

I know (K%N)th guy will be eliminated.

So my winner will be

$$(K\%N) + f(N-1, K)$$

but since its a circle, we have to cover it in a loop.

$$\left((K\%N) + f(N-1, K)\right) \% N.$$

code.

```
f(n,K){

  int looser = K%n;

  if(looser==0){
      looser= N  => to adjust 1 based indexing
  }

  int winner = (looser + f(n-1, K)) % N.

   if (winner==0){

        winner= N; => adjust
   }

  return winner;

}
```