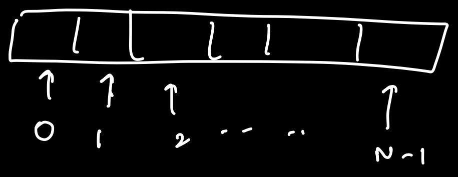
// int [ ] arr = new int [5] / int [N]



N = len of array.

arr[0]   arr[10]   . arr [i]

```
// void Print ( int arr[] / int[] arr )
{
    int n = arr.length
    for ( i=0 ; i<N ; i++)
    {
        print ( arr[i])
```

Q. Given an array of size N. Count no. of elements having atleast one element greater than itself.

arr[7] = {-3, -2, 6, 8, 4, 8, 5}   ans = 5.

arr[9] = {2, 3, 10, 7, 3, 2, 10, 8, 10}   ans = 6

for every element, go check in array if any element greater than itself.

$O(N^2)$

$1 < N < 10^5$

Obs 1: for max element, there is no element greater than itself.

Obs2 : Other than max, all elements will have atleast one element greater than itself.

Pseudocode

1. Iterate, get max ele in array.
2. Iterate, check how many != max

```
int countgreater( int [] arr)
{
    int n = arr.length
    int m =        Math.min / arr[0] /arr[1]
    for( i=0 ; i<N; i++)
    {
        if( arr[i] > m )   m= arr[i]
    }

    int ans=0
    for ( i=0; i<N ; i++)
    {
        if( arr[i] != m)   ans++
    }

    return ans
}
```

TC: $O(N)$

SC: $O(1)$

ToDo: Try to solve this in one traversal

Google

Q. Given a array, Check $\overset{\text{True/false}}{\text{if}}$ there exists a pair (i,j), such that

$$arr[i] + arr[j] = \underset{\text{Given}}{\underset{\downarrow}{K}} \quad \&\& \quad i \,!= j$$

Ex $arr = \{\overset{0}{3}, \overset{1}{-2}, \overset{2}{1}, \overset{3}{4}, \overset{4}{3}, \overset{5}{6}, \overset{6}{8}\}$

$K = 10$

$i = 3$

$j = 5$

True

$arr = \{\overset{0}{2}, \overset{1}{4}, \overset{2}{-3}, \overset{3}{7}\}$

$K = 8$

False

Sol Check every pair, see if it is meeting condition or not

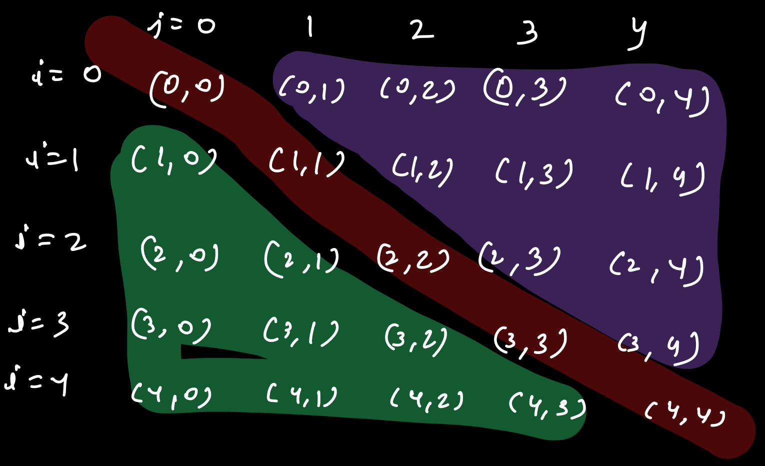
```
for( i=0 ; i< n; i++)
{
    for( j=0 ; j<n ; j++ )
    {
        // (i,j)
        If( i != j  && arr[i] + arr[j] == k )
                return True
    }
}

return false
```

TC : O(N^2)

SC : O(1)

0, 1, 2, 3, 4



	j=0	1	2	3	4
i=0	(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
i=1	(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
i=2	(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
i=3	(3,0)	(3,1)	(3,2)	(3,3)	(3,4)
i=4	(4,0)	(4,1)	(4,2)	(4,3)	(4,4)

0
1
2
3
4
N-1

# of iteration = N^2 TC: O(N^2)

$i=0$	×
$i=1$	$j<1$
$i=2$	$j<2$
$i=3$	$j<3$
⋮	
i	$j<i$

```
for( i=0 ; i< n ; i++ )
{
        for( j=0 ; j< i ; j++ )
        {
            // (i,j)
            if( i!=j && arr[i] + arr[j] == k )
                              return True
        }
}

return false
```

$$\#\text{ of iterations} = 0+1+2+3+\dots N-1$$

$$= \frac{N(N-1)}{2}$$

$$= \frac{N^2}{2} - \frac{N}{2}$$

TC: $O(N^2)$

SC: $O(1)$

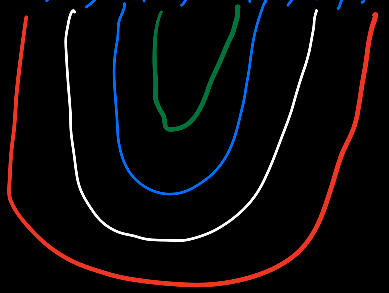
Q. Given an array. Reverse entire arr[].

Note : arr[] itself should change

Expected SC: O(1)

Ex arr[8] = [-1, 4, 7, 6, -2, 7, 8, 10]

0	1	2	3	4	5	6	7
-1	4	7	6	-2	7	8	10

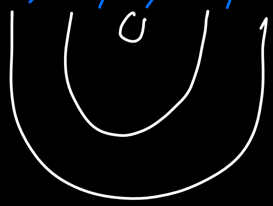


i	j
0	7
1	6
2	5
3	4
4	3

arr[8] : [10, 8, 7, -2, 6, 7, 4, -1].

arr[5] = [3, 2, 1, 4, 5]

0	1	2	3	4
3	2	1	4	5



i	j
0	4
1	3
2	2
3 >	1

```
int a = ?
int b = ?
    temp = b
    b = a
    a = temp
```

```
i = 0
j = n-1

while ( i < j )
{
    swap( arr[i], arr[j] )        temp
    i++
    j--
}
```

# of iterations = $\frac{N}{2}$

TC: O(N)
SC: O(1)

Todo: Solve it using for loop

Q. Given an array, s, e. reverse array [s, e]

	0	1	2	3	4	5	6	7	8
arr[] =	{ -3,	4,	2,	3,	7,	9,	6,	2,	10 }

s = 3
e = 7

```
reversepart ( arr[], s, e)
{
        i = s
        j = e
        while ( i < j )          → temp
        {
            swap( arr[i], arr[j] )
            i++
            j--
        }
}
```

# of iteration = $\frac{e-s+1}{2}$

TC: O(e - s)

Airtel, Amazon

s = 0
e = N-1

Q. Given an array, Rotate array from last from first, K no ~~of~~ times. (K < N)

SC: O(1)

arr[] =	-3	2	1	4	6	9	8
K = 1	8	-3	2	1	4	6	9

K = 2   9   8   -3   2   1   4   6

K = 3   6   9   8   -3   2   1   4

arr[10] : $a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7\ a_8\ a_9\ a_{10}$

K = 4 $a_{10}\ a_1\ a_2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$

$a_9\ a_{10}\ a_1\ 2\ 3\ 4\ 5\ 6\ 7\ 8$

$a_8\ a_9\ a_{10}\ a_1\ 2\ 3\ 4\ 5\ 6\ 7$

$$\underbrace{a_7\ a_8\ a_9\ a_{10}}_{\text{last K elements}}\ \underbrace{a_1\ a_2\ a_3\ a_4\ a_5\ a_6}_{\text{first N-K elements}}.$$

$a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7\ a_8\ a_9\ a_{10}$

↓ Reversing.

$$\begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ a_{10} & a_9 & a_8 & a_7 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 \end{array}$$

$$\underbrace{a_{10}\ a_9\ a_8\ a_7}_{\text{Reverse first K elements}}\ \underbrace{a_6\ a_5\ a_4\ a_3\ a_2\ a_1}_{\text{Reverse last N-K elements.}}$$

$a_7\ a_8\ a_9\ a_{10}\ a_1\ a_2\ a_2\ a_3\ a_4\ a_5\ a_6.$

1. Reverse entire array
2. Reverse first K elements.
3. Reverse last N-K elements.

```
void   RotateKtimes( int arr[], int K)
{
        int n = arr.length
        reversepart( arr, 0, N-1)     // reverse entire array
        reversepart( arr, 0, K-1)     // revers first K
        reversepart( arr, K, N-1)     // reverse last N-K
```

# of iteration $= \frac{N}{2} + \frac{K}{2} + \frac{N-K}{2}$

$= \frac{N}{2} + \cancel{\frac{K}{2}} + \frac{N}{2} - \cancel{\frac{K}{2}}$

$= N$

TC : O(N)

SC : O(1)

Doubts.

1 time.

```
for(i=1 ; i≤N; i++)
{
        int  temp = i
}
```

```
for(i=0 ; i<=K ; i++)
{
        arr [0] = arr[N]
        arr[N+1] = arr[N]
}
```

N = si

lose arr[0]

Error

2^{2n}

$(2^2)^n$ (4^n)

TC : 4^n