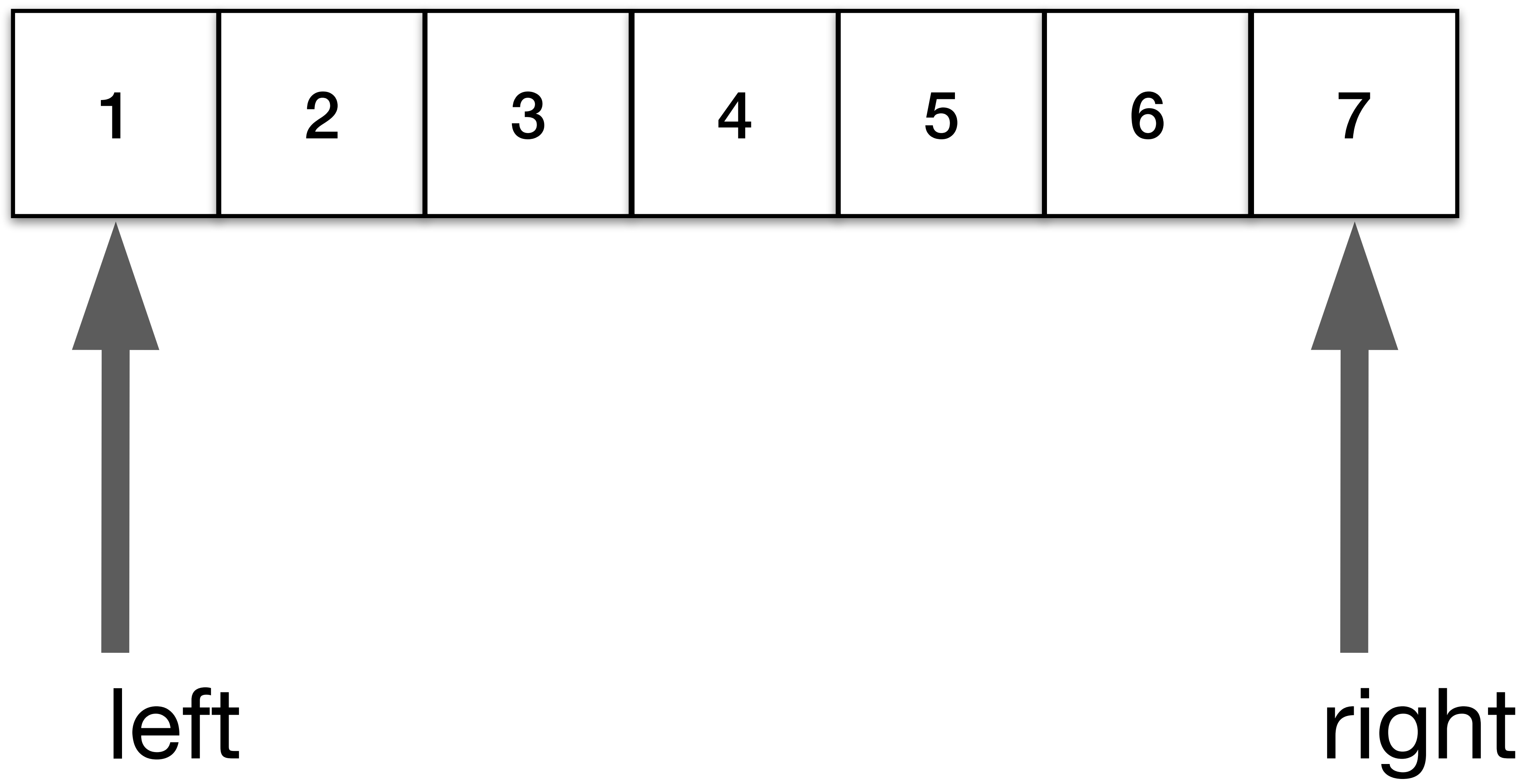
## Rotate Array - Test Case 1

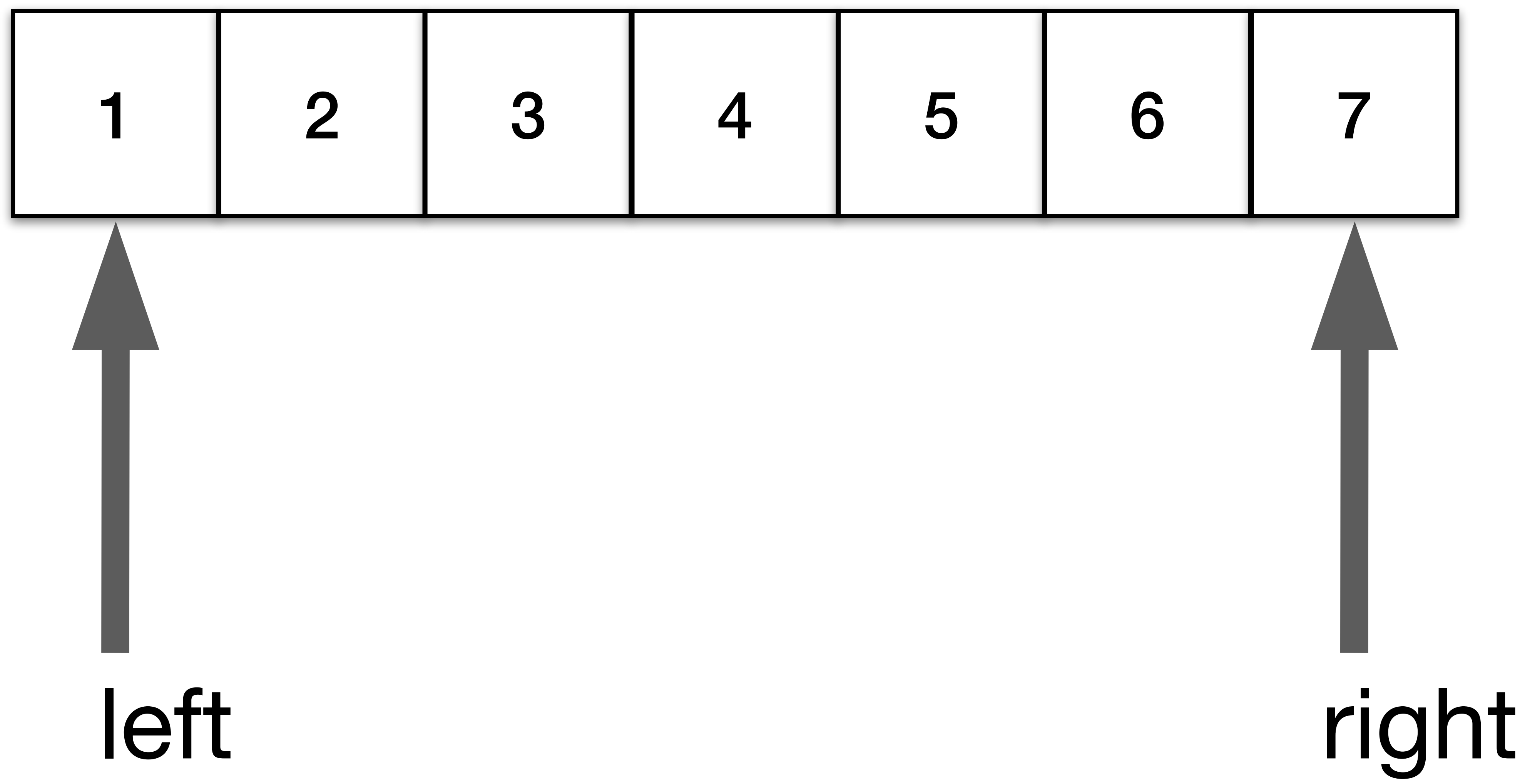


k = 3

Template : - Base structure opposite direction pointer template

We need to swap the two numbers.

## Rotate Array - Test Case 1



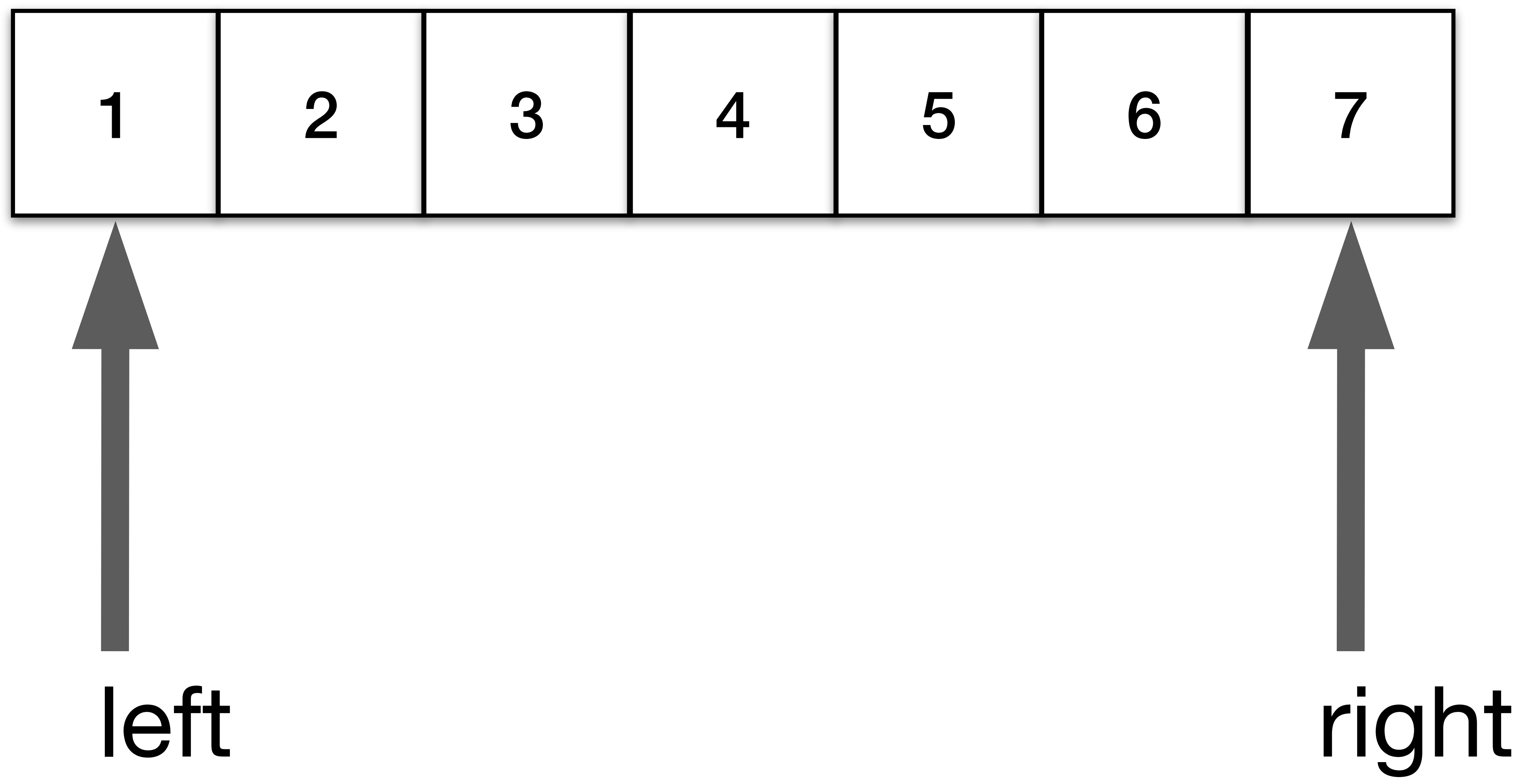
Algorithm : - Along with template we needs to implement below algorithm are to get our desire result

Step 1: - Reverse the entire array to bring the last k elements to the front in reversed order.

Step 2: -Reverse the first k elements to restore the original order.

Step 3: -Reverse the remaining part of the array (from index k to end) to restore its order as well.

## Rotate Array - Test Case 1 - Step 1

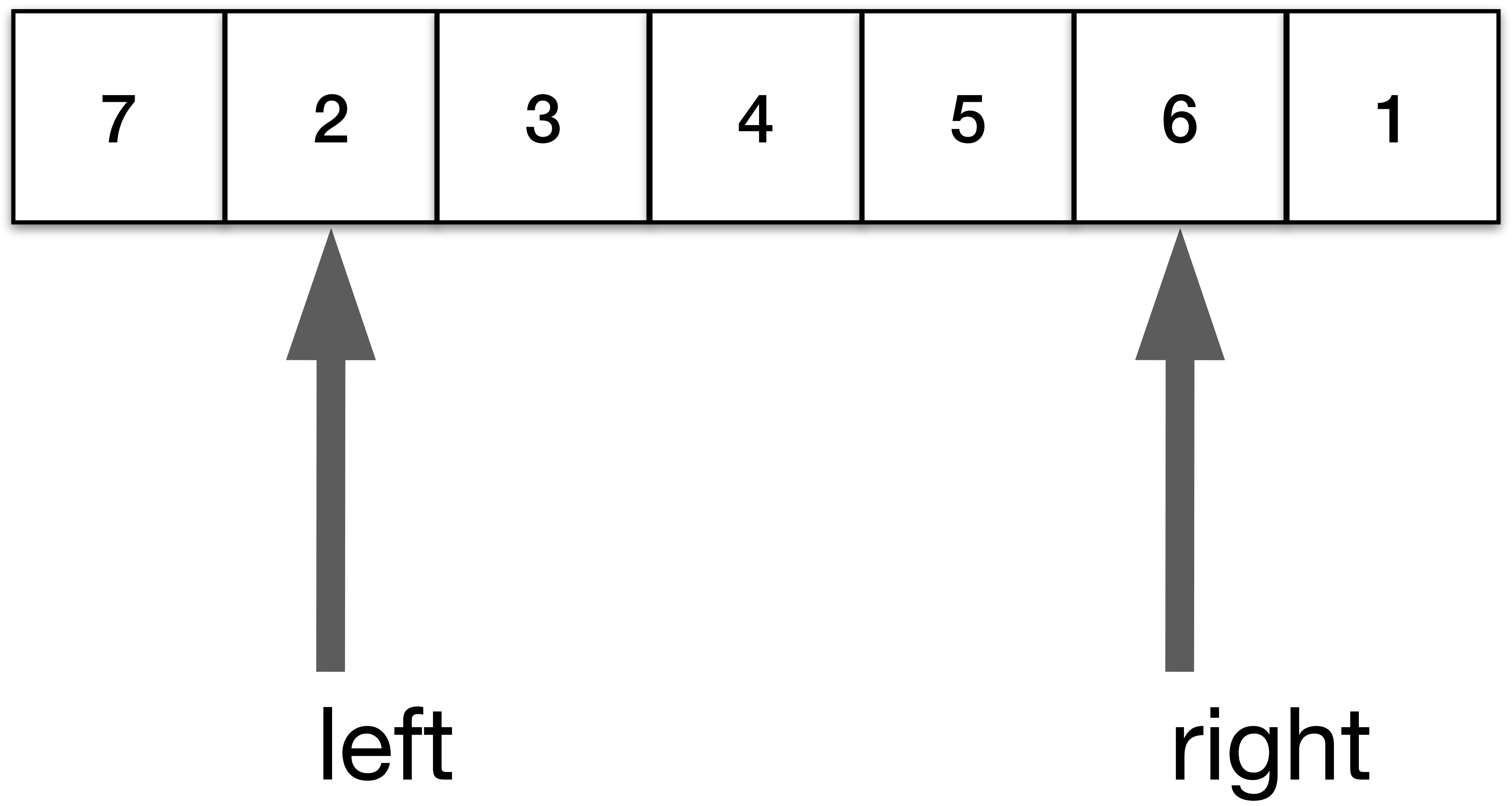


k = 3

Step 1: - Reverse the entire array to bring the last k elements to the front in reversed order.

Swapping 1 and 7 number and moving both the pointers.

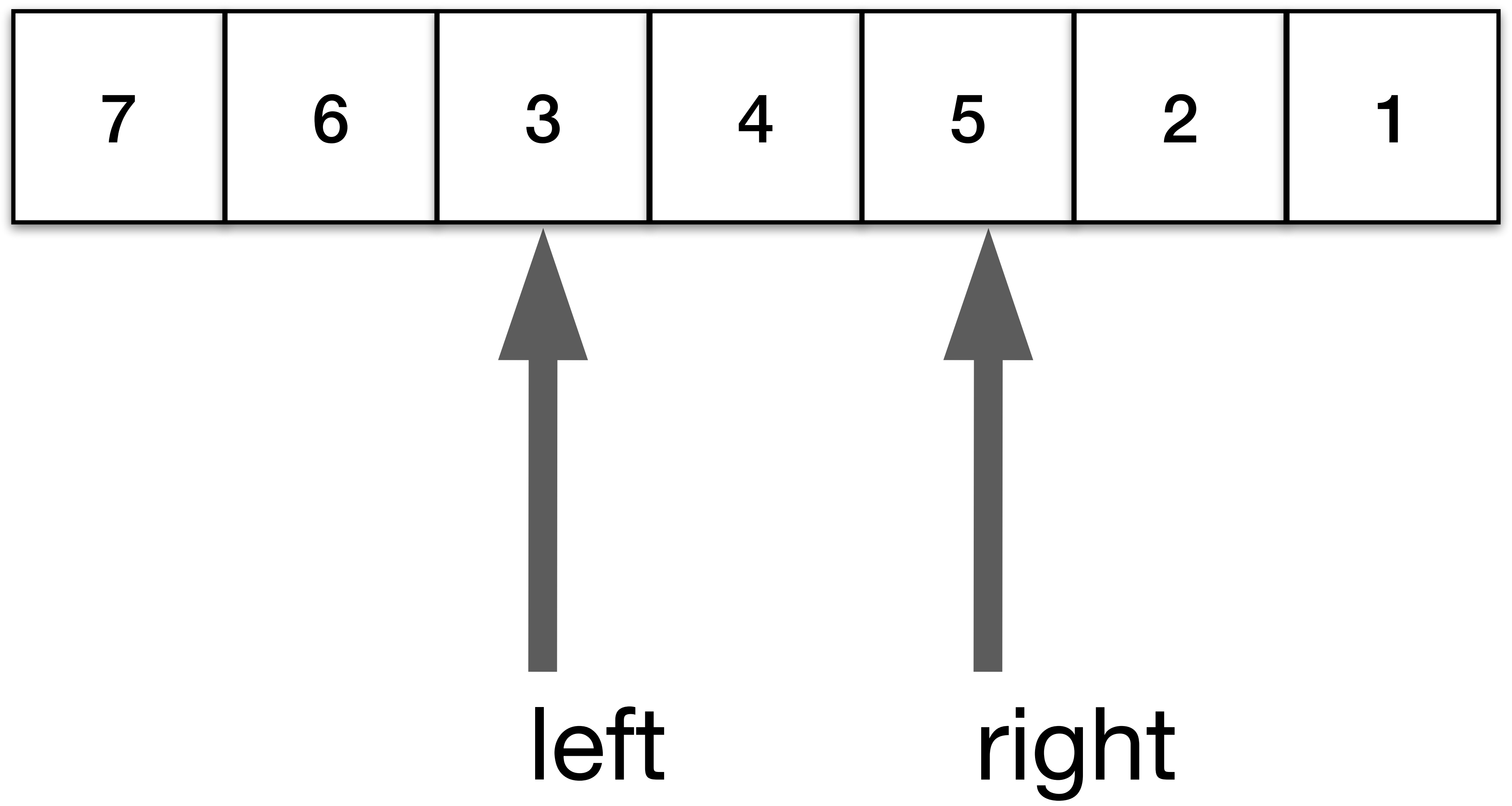
## Rotate Array - Test Case 1 - Step 1



k = 3

Swapping 2 and 6 number and moving both the pointers.

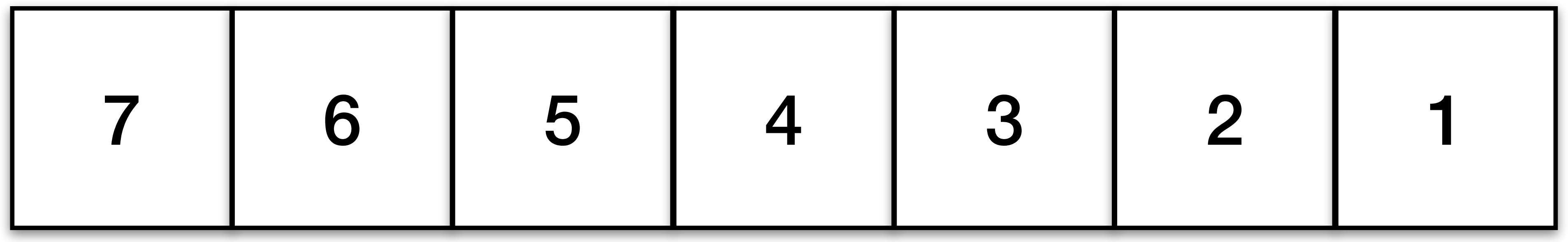
## Rotate Array - Test Case 1 - Step 1



k = 3

Swapping 3 and 5 number.

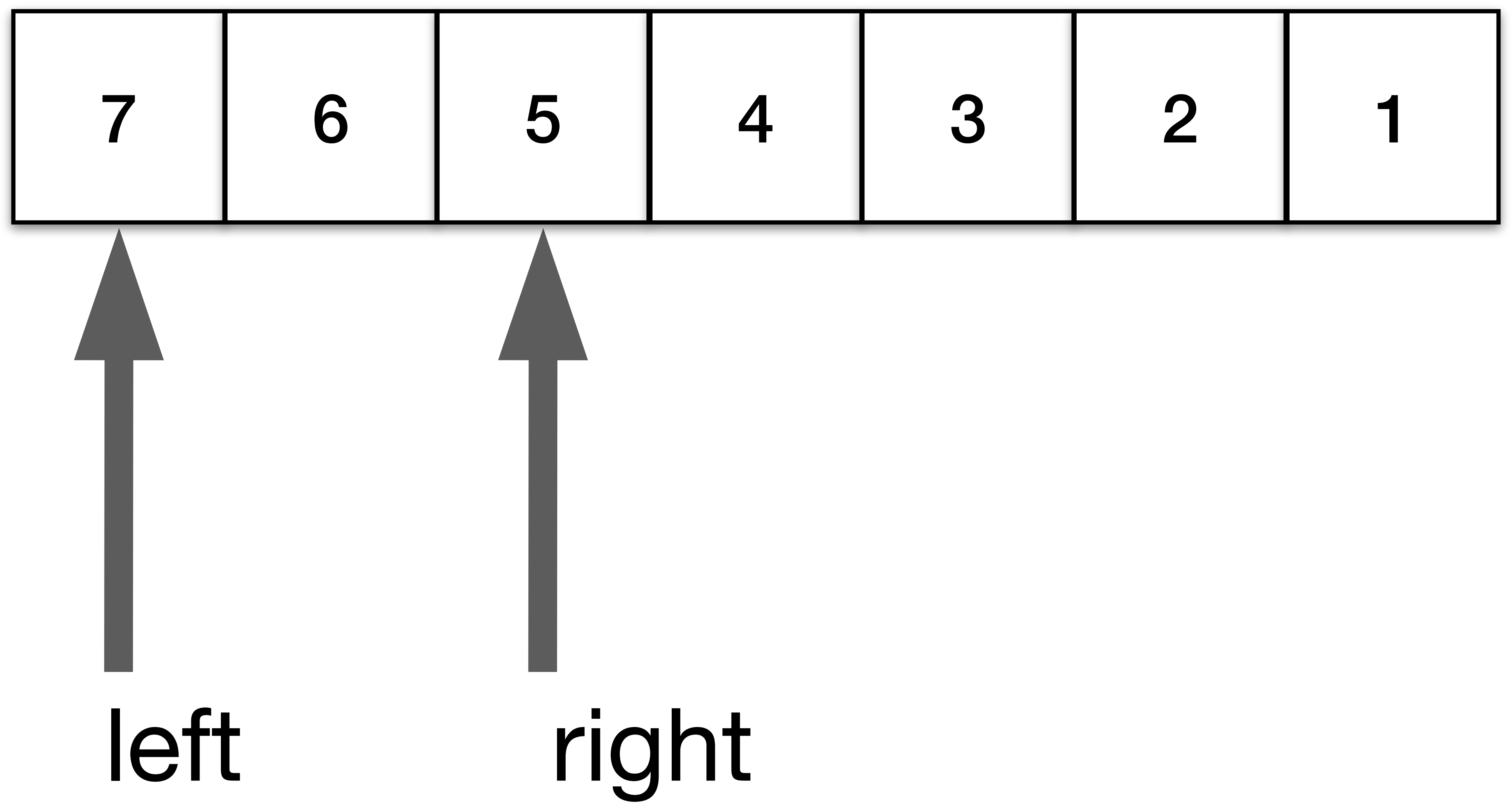
## Rotate Array - Test Case 1 - Step 1



k = 3

First step of algorithm completed

# Rotate Array - Test Case 1 - Step 2



k = 3

Step 2: - Reverse the first k elements to restore their original order.

Swapping 7 and 5 number.

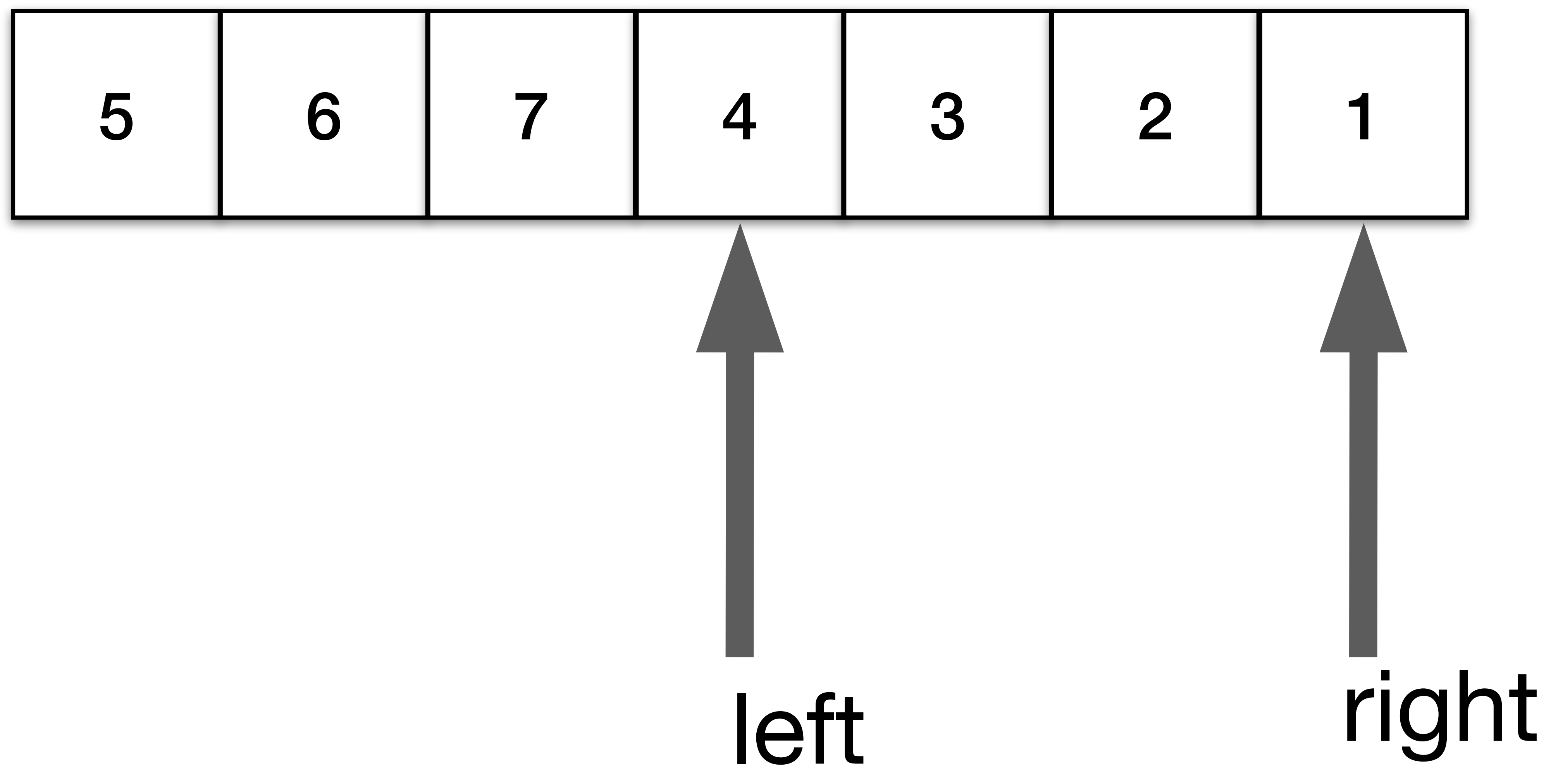
## Rotate Array - Test Case 1 - Step 1

| | | | | | | |
|---|---|---|---|---|---|---|
| 5 | 6 | 7 | 4 | 3 | 2 | 1 |

k = 3

Second step of algorithm completed

# Rotate Array - Test Case 1 - Step 3

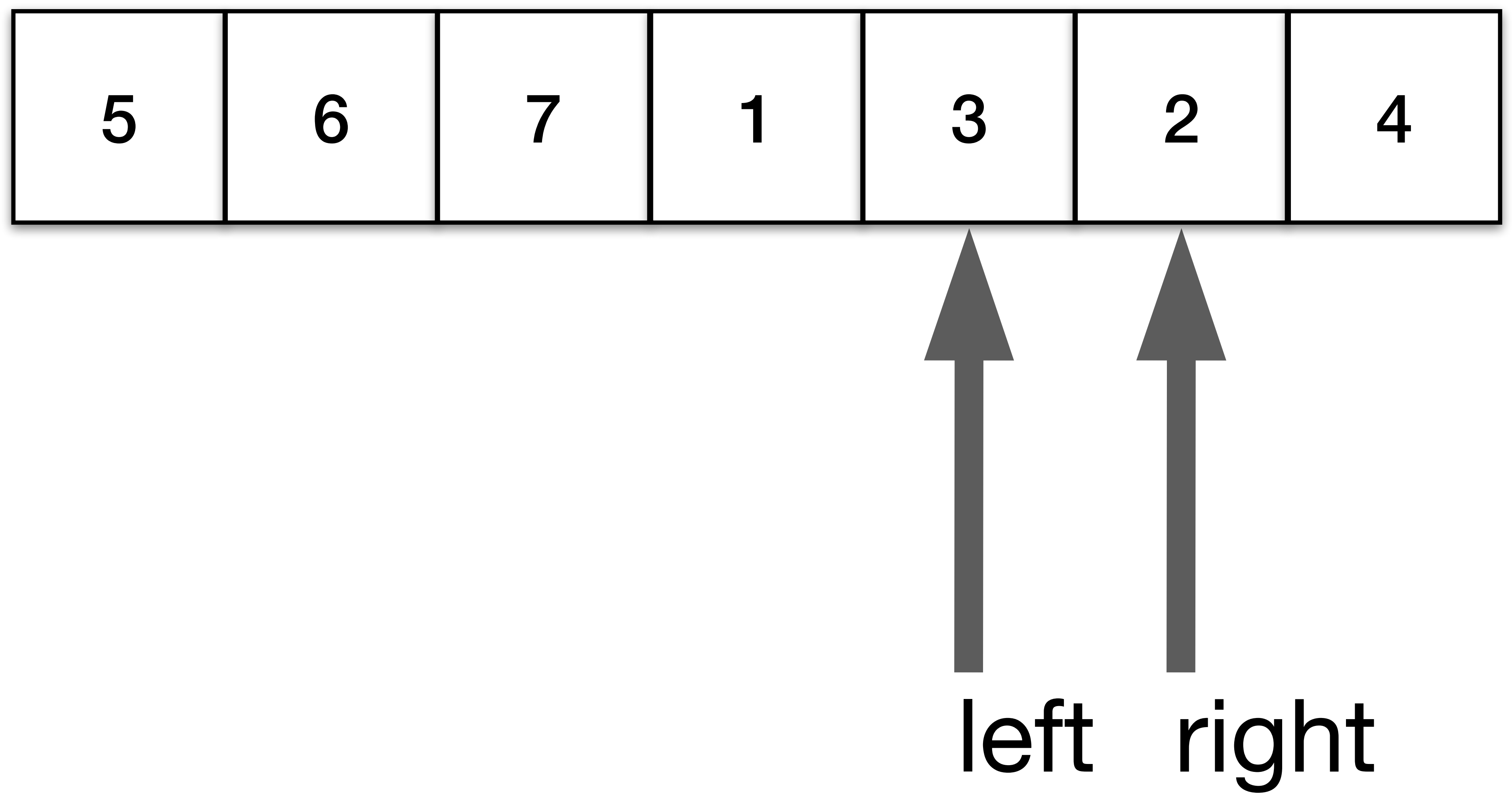


k = 3

Step 3: -Reverse the remaining part of the array (from index k to end) to restore its order as well.

Swapping 4 and 1 number and moving both the pointers.

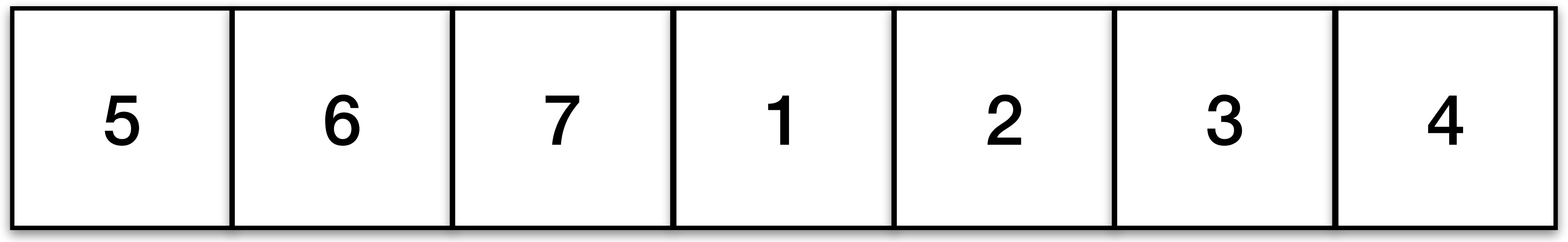
## Rotate Array - Test Case 1 - Step 3



k = 3

Swapping 3 and 2 number.

## Rotate Array - Test Case 1 - Step 3

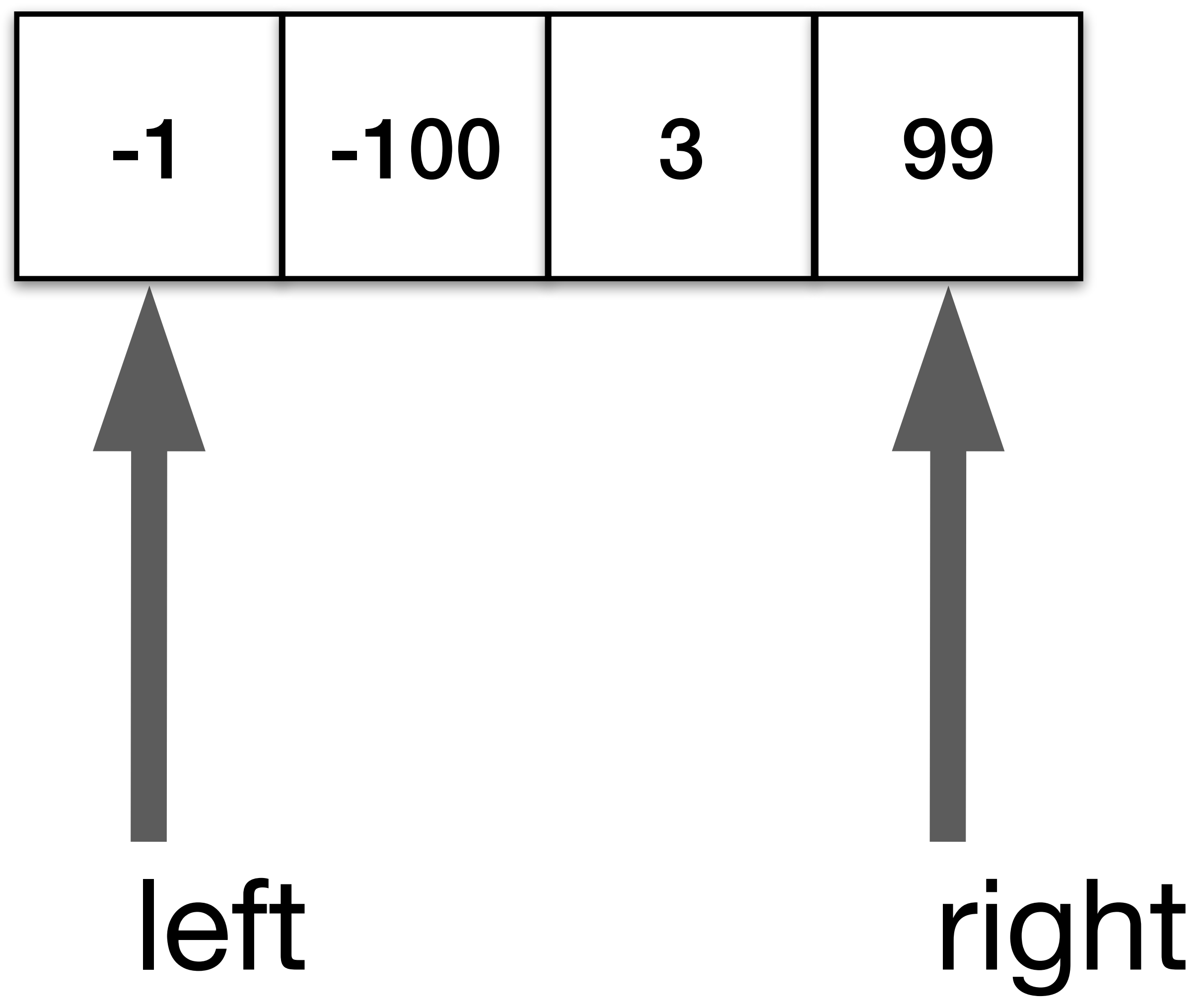


k = 3

Third step of algorithm completed

This is the output

# Rotate Array - Test Case 2 - Step 1

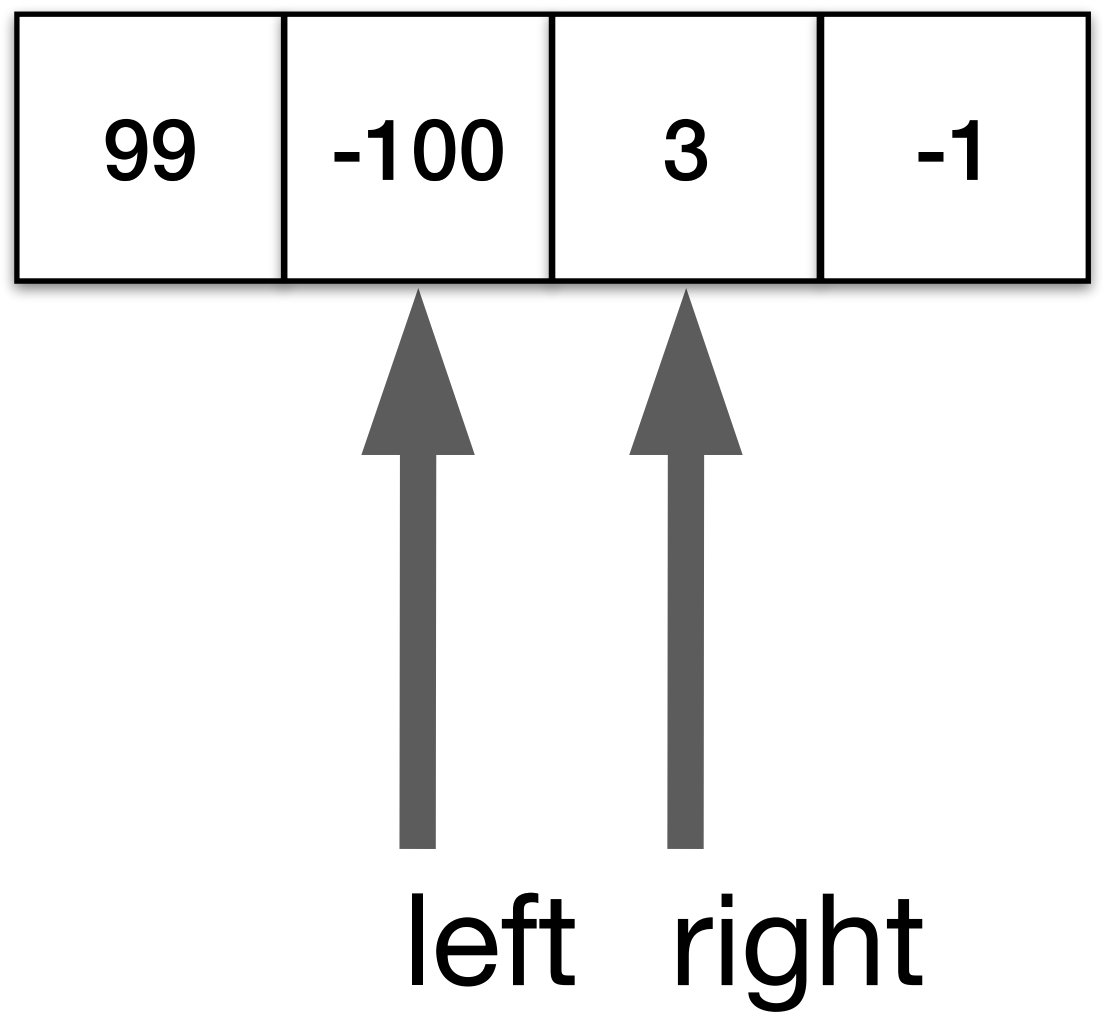


k = 2

Step 1: - Reverse the entire array to bring the last k elements to the front in reversed order.

Swapping -1 and 99 number and moving both the pointers.

# Rotate Array - Test Case 2 - Step 1



k = 2

Swapping -100 and 3 number.

## Rotate Array - Test Case 2 - Step 1

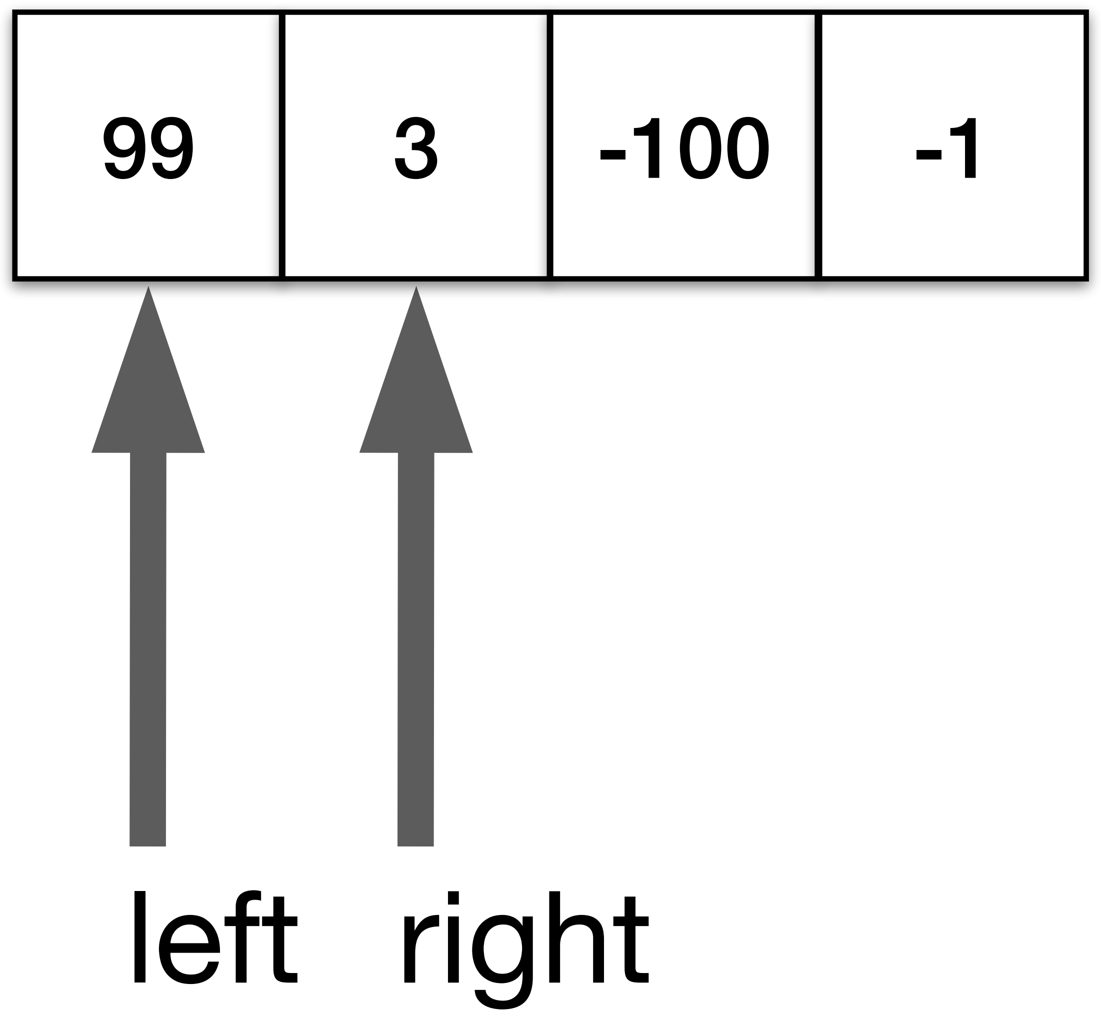
| | | | |
|---|---|---|---|
| 99 | 3 | -100 | -1 |

k = 2

Swapping -100 and 3 number.

First step of algorithm completed

# Rotate Array - Test Case 2 - Step 2



k = 2

Step 2: - Reverse the first k elements to restore their original order.

Swapping 99 and 3 number.

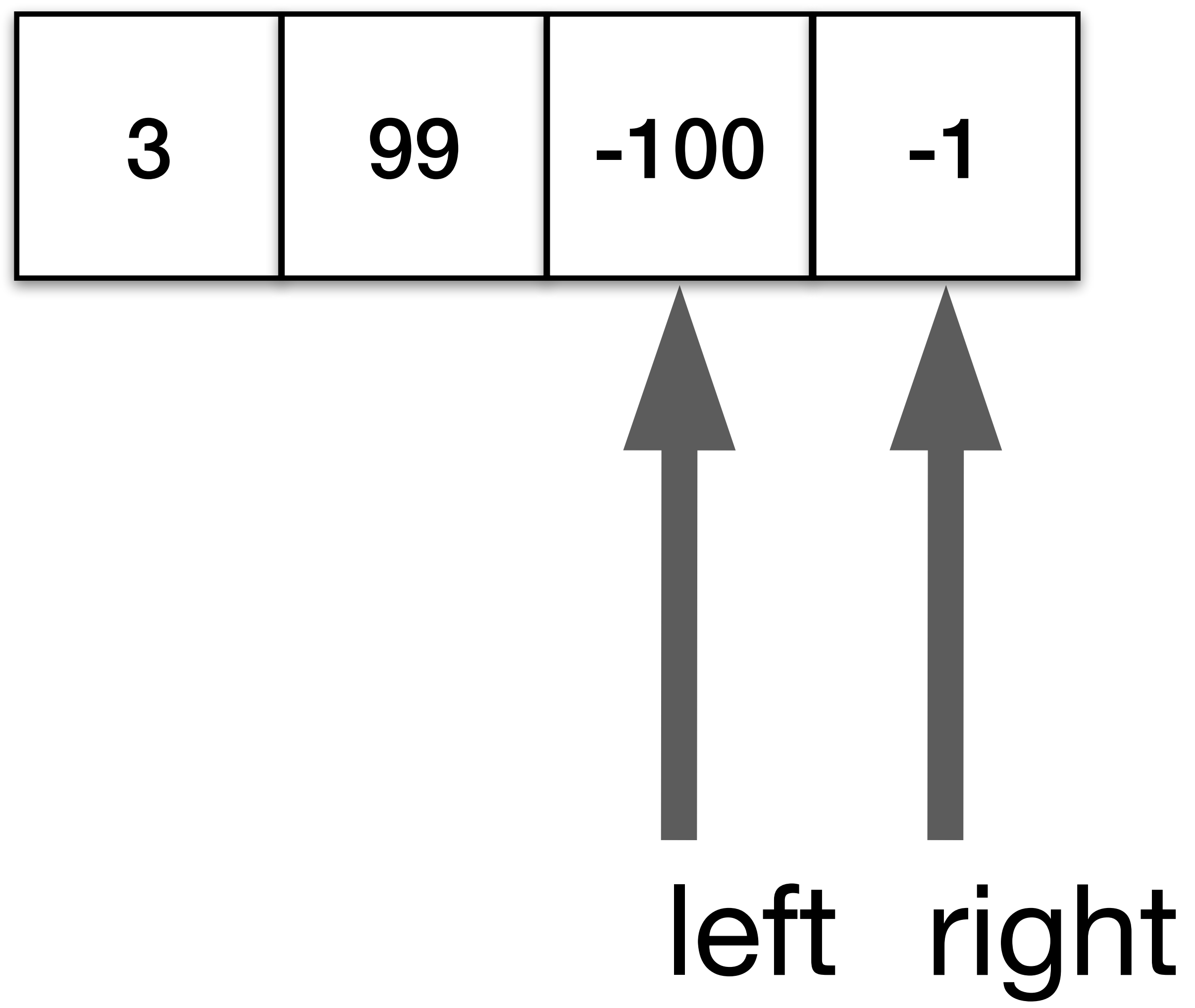
# Rotate Array - Test Case 2 - Step 2

| | | | |
|---|---|---|---|
| 3 | 99 | -100 | -1 |

k = 2

Second step of algorithm completed.

## Rotate Array - Test Case 2 - Step 3



k = 2

Step 3: -Reverse the remaining part of the array (from index k to end) to restore its order as well.

Swapping -100 and -1 number.

## Rotate Array - Test Case 2 - Step 3

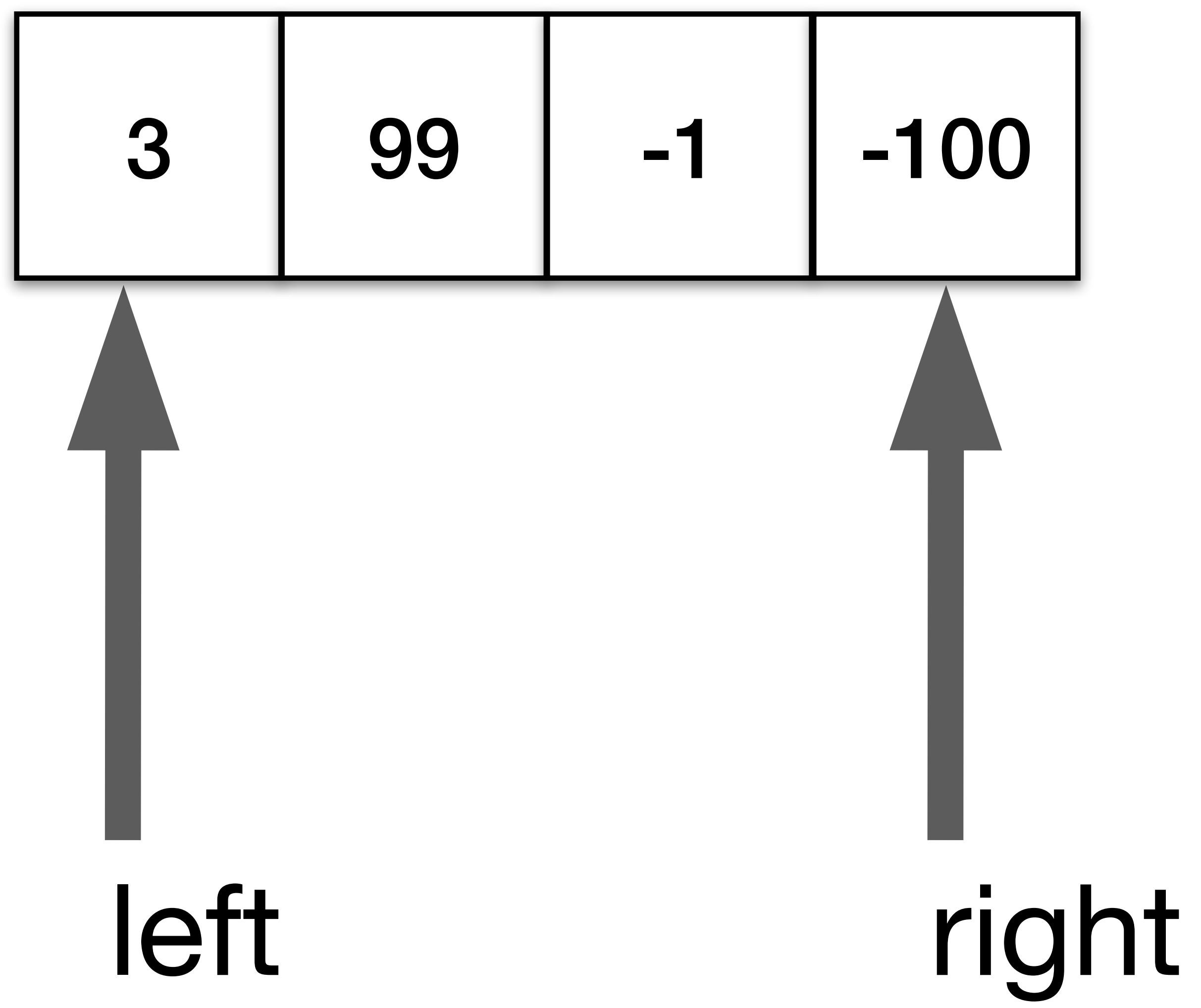
|  |  |  |  |
|---|---|---|---|
| 3 | 99 | -1 | -100 |

k = 2

Third step of algorithm completed

This is the output

## Step1 :- Rotate Array - Dynamic params for template



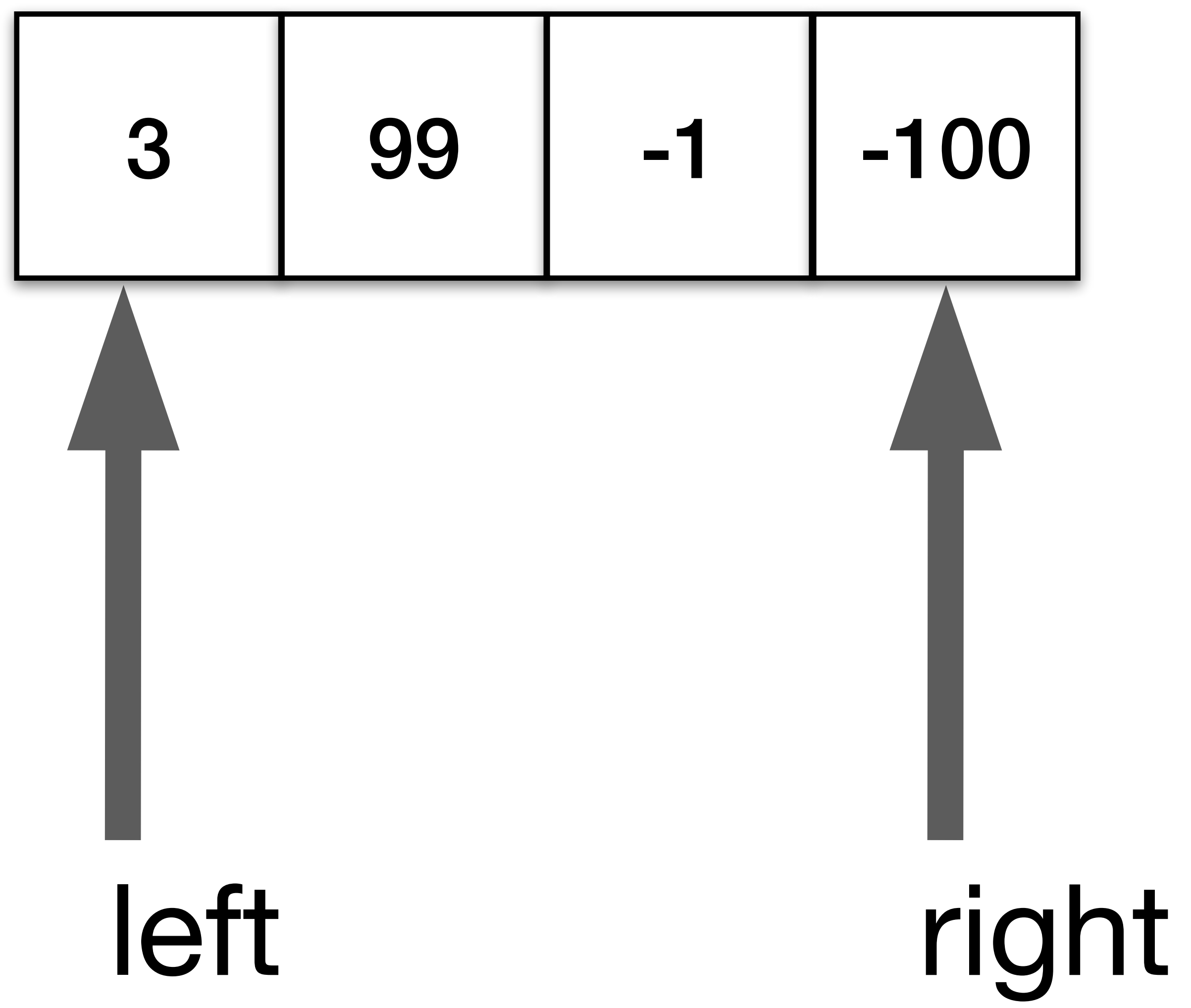
k = 2

Template : - Base structure opposite direction pointer template

We need to swap the two numbers.

```
temp = arr[left]
arr[left] = arr[right]
arr[right] = temp
```

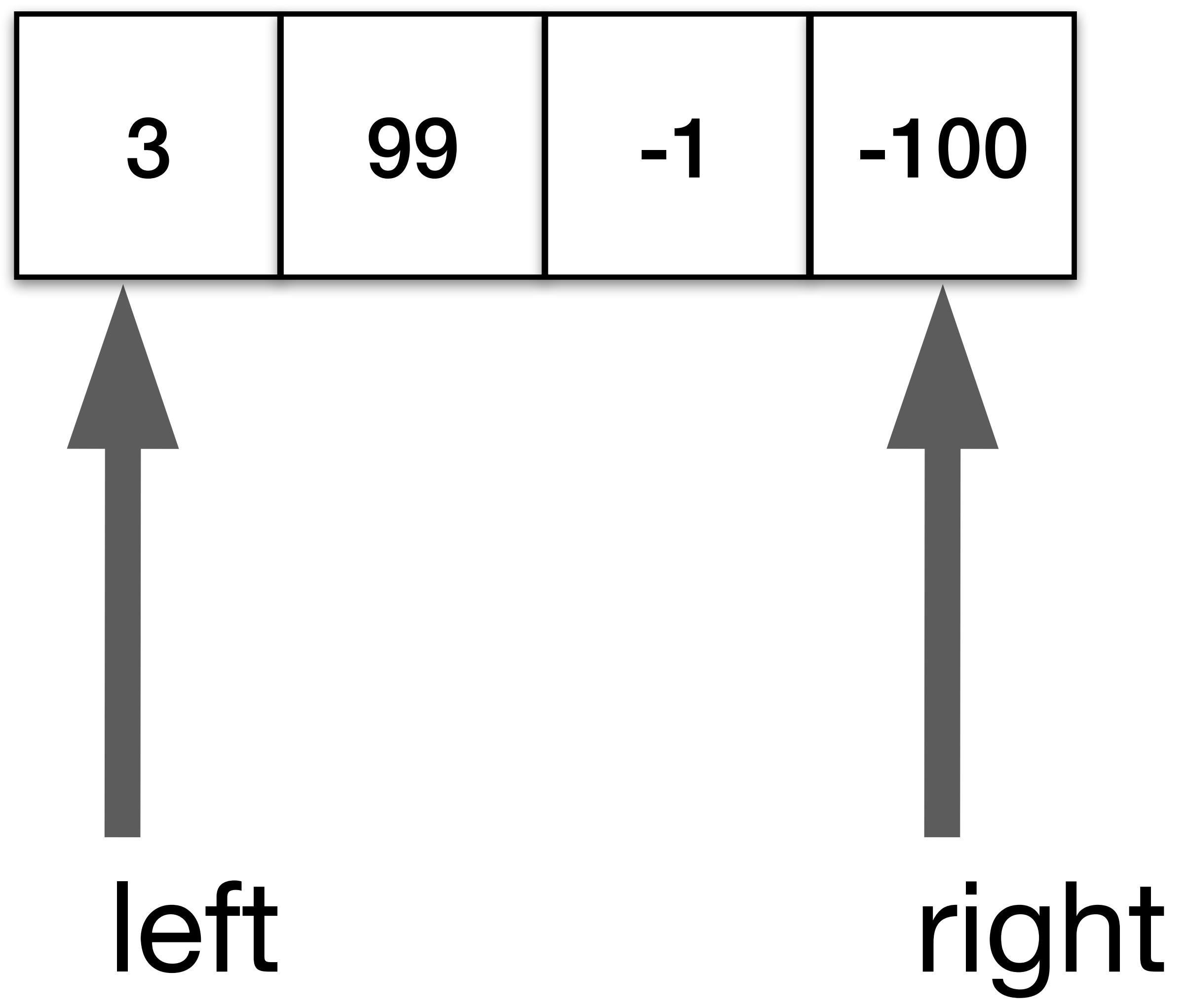
# Rotate Array - Dynamic param for I/O



k = 2

There is no dynamic parameter for base template required and in this problem we do not required output parameter as well.

## Step 2 :- Rotate Array - Dynamic param for Algorithm Step 1



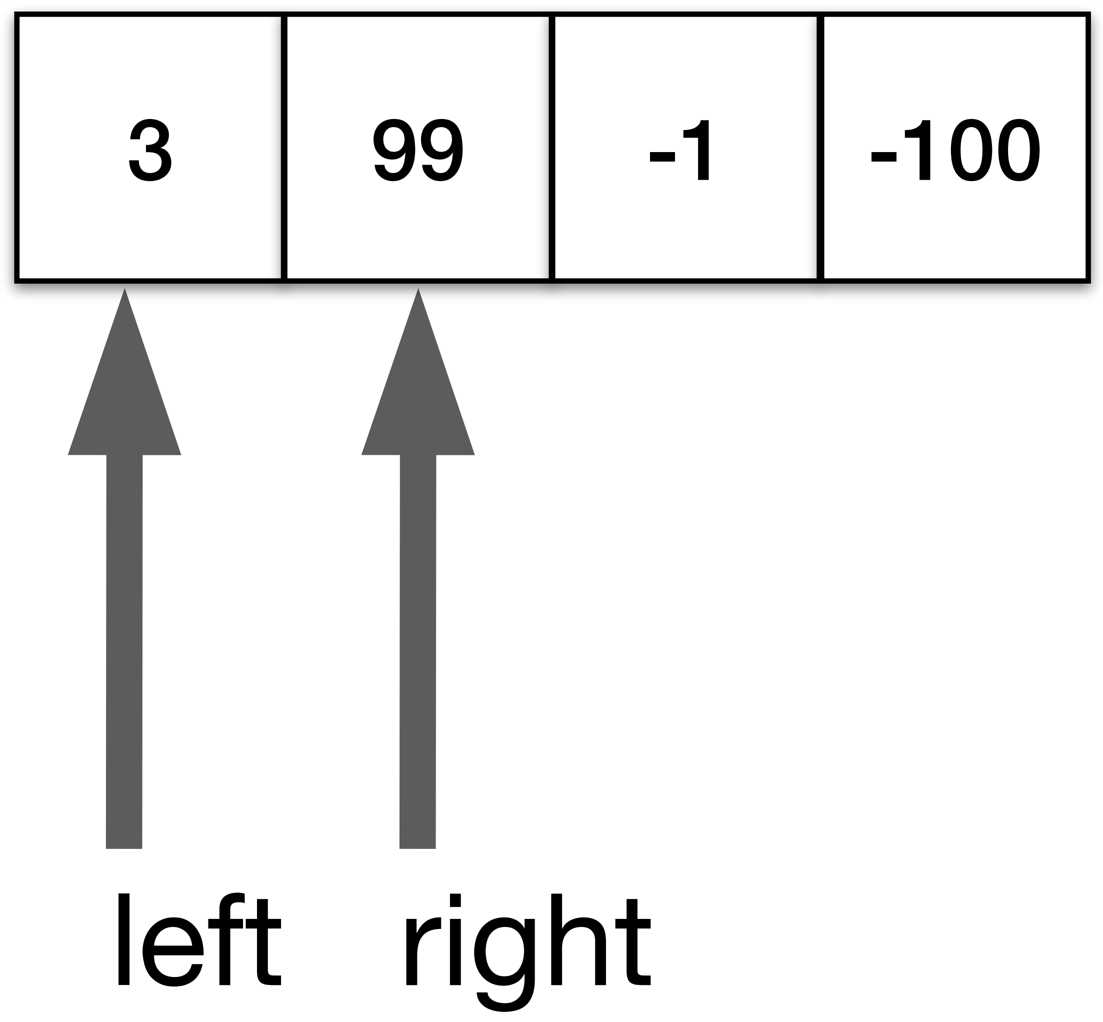
k = 2

Step 1: - Reverse the entire array to bring the last k elements to the front in reversed order.

From left = 0 to right = are.length -1

Swap all numbers with base template

## Step 3 :- Rotate Array - Dynamic param for Algorithm Step 2



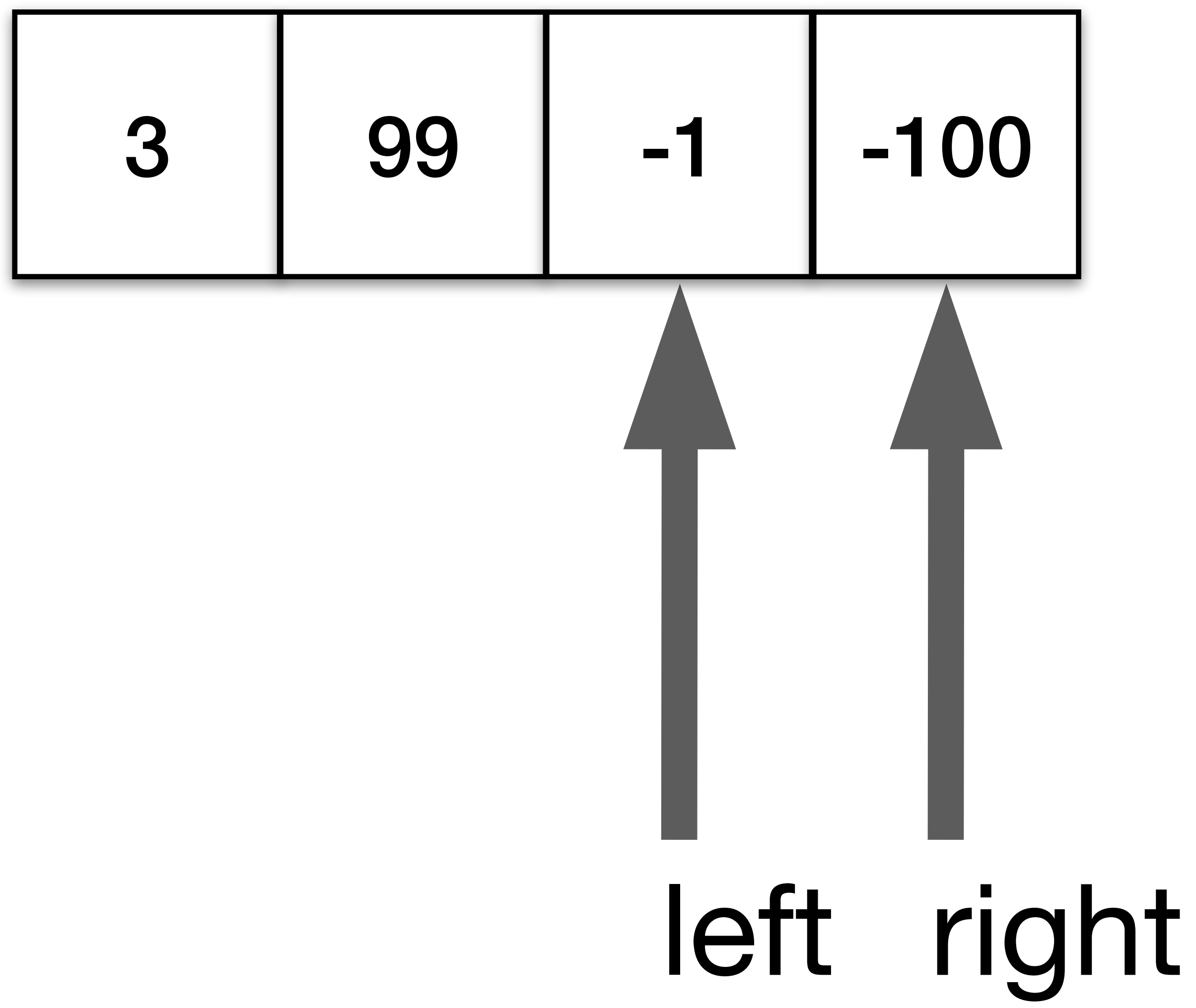
k = 2

Step 2: -Reverse the first k elements to restore their original order.

From left = 0 to k

Swap all numbers with base template

## Step 4 :- Rotate Array - Dynamic param for Algorithm Step 3



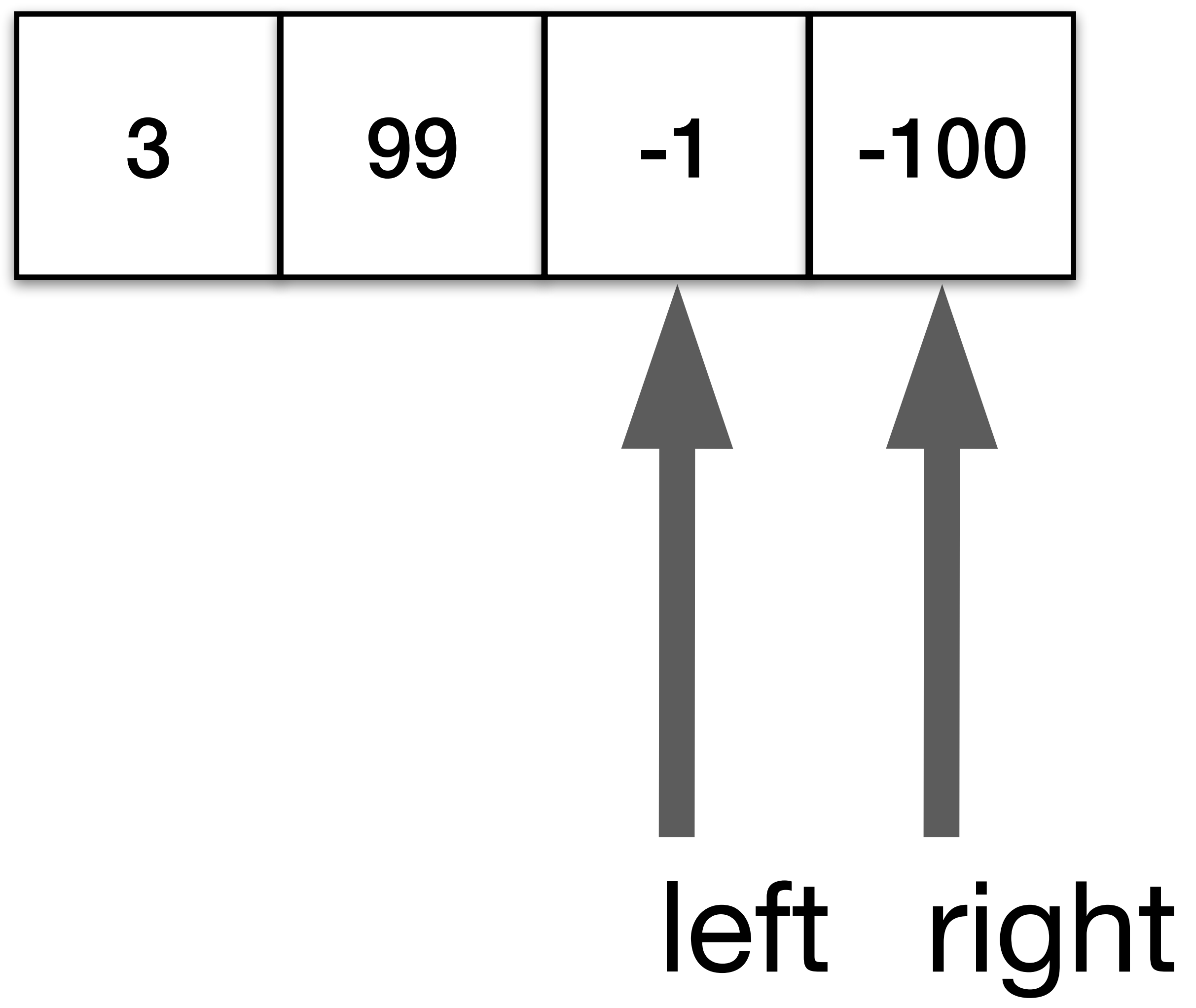
k = 2

Step 3: -Reverse the remaining part of the array (from index k to end) to restore its order as well.

From k to right = arr.length -1

Swap all numbers with base template

## Rotate Array - Corner Case Scenario

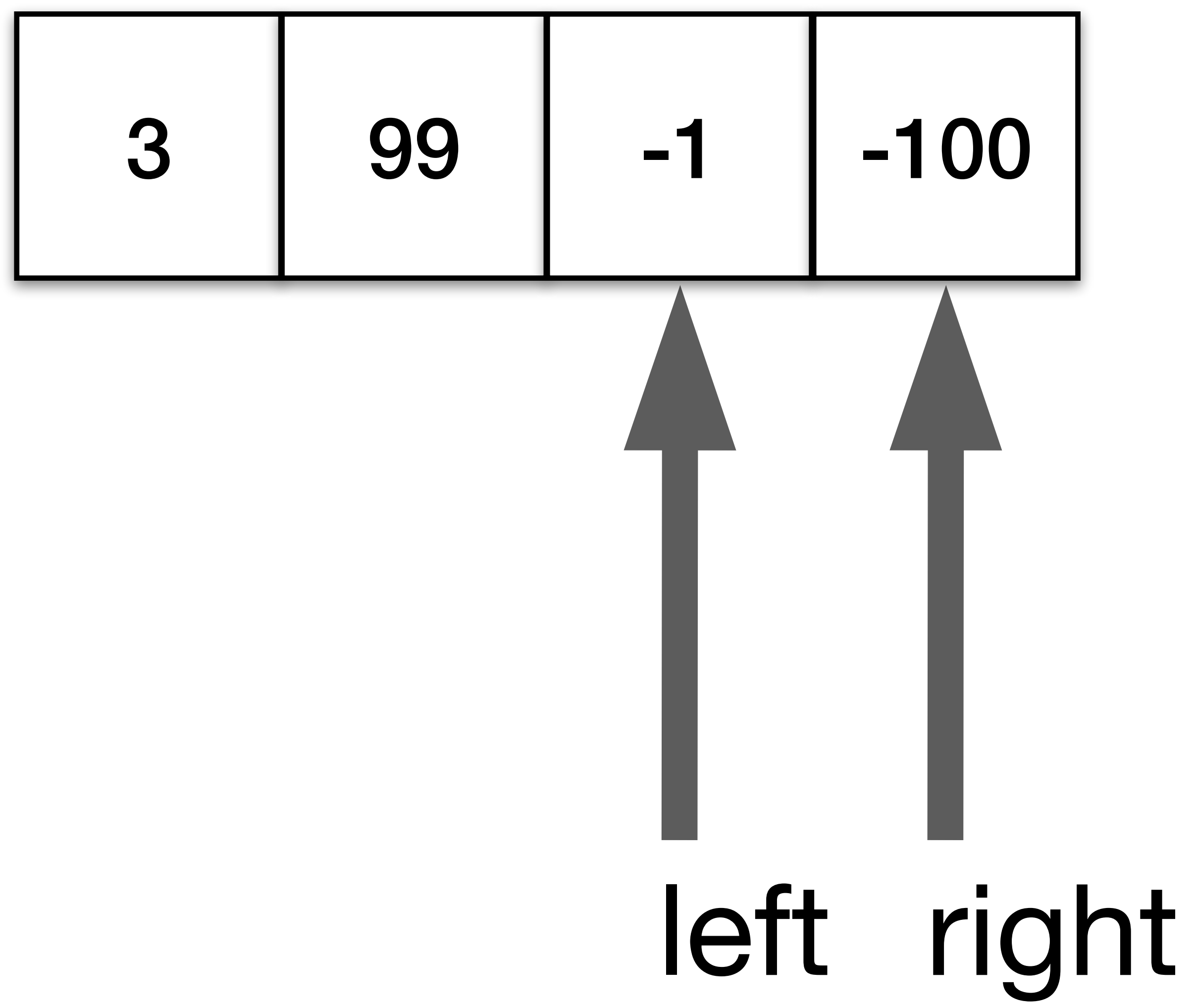


k = 2

Step 3: -Reverse the remaining part of the array (from index k to end) to restore its order as well.

Swap all numbers with base template

## Step 5 :- Rotate Array - Dynamic param for Corner Case Scenario



k = 2

Step 3: -Reverse the remaining part of the array (from index k to end) to restore its order as well.

From k to right = arr.length -1

Swap all numbers with base template