

1.Easy\06.Rotate_array_left&right_by_k_places.cpp

```
1  /*
2  QUESTION:-
3
4  Given an integer array nums, rotate the array to the right by k steps, where k is non-
negative.
5
6  Example 1:
7
8  Input: nums = [1,2,3,4,5,6,7], k = 3
9  Output: [5,6,7,1,2,3,4]
10 Explanation:
11 rotate 1 steps to the right: [7,1,2,3,4,5,6]
12 rotate 2 steps to the right: [6,7,1,2,3,4,5]
13 rotate 3 steps to the right: [5,6,7,1,2,3,4]
14 Example 2:
15
16 Input: nums = [-1,-100,3,99], k = 2
17 Output: [3,99,-1,-100]
18 Explanation:
19 rotate 1 steps to the right: [99,-1,-100,3]
20 rotate 2 steps to the right: [3,99,-1,-100]
21 */
22
23 /*
24 APPROACH:-
25 To rotate the array k places to right follow below steps
26 -> Reverse first n-k elements
27 -> Reverse last k elements
28 -> Reverse the entire array
29
30 To rotate the array k places to left follow below steps
31 -> Reverse first k elements
32 -> Reverse last n-k elements
33 -> Reverse the entire array
34 */
35
36 // CODE:-
37
38 // RIGHT ROTATE:-
39 void rightRotate(int arr[], int n, int k)
40 {
41     k = k % n; // to keep k within the range
42     reverse(arr, arr + (n - k));
43     reverse(arr + (n - k), arr + n);
44     reverse(arr, arr + n);
45 }
46
47 // LEFT ROTATE:-
48 void leftRotate(int arr[], int n, int k)
49 {
50     k = k % n; // to keep k within the range
51     reverse(arr, arr + k);
```

```
52     reverse(arr + k, arr + n);  
53     reverse(arr, arr + n);  
54 }  
55  
56 // TIME COMPLEXITY = O(N)  
57 // SPACE COMPLEXITY = O(0)
```