

Given an integer array `nums`, rotate the array to the right by `k` steps, where `k` is non-negative.

Example 1:

Input: `nums = [1,2,3,4,5,6,7]`, `k = 3`
Output: `[5,6,7,1,2,3,4]`
Explanation:
rotate 1 steps to the right: `[7,1,2,3,4,5,6]`
rotate 2 steps to the right: `[6,7,1,2,3,4,5]`
rotate 3 steps to the right: `[5,6,7,1,2,3,4]`

1 2 3 4 5 6 7 k=3
5 6 7 1 2 3 4

$n=7$
 $k=143$
 $k=14$
 $k=21$
 $k=28$
 $k=35$
 $k=42$
 $k=49$
 $k=56$
 $k=63$
 $k=70$

$k=1$ 1 2 3 4 5 6 7
 $k=2$ 7 1 2 3 4 5 6
 $k=3$ 6 7 1 2 3 4 5
 $k=4$ 5 6 7 1 2 3 4
 $k=5$ 4 5 6 7 1 2 3
 $k=6$ 3 4 5 6 7 1 2
 $k=7$ 2 3 4 5 6 7 1
 $k=8$ 1 2 3 4 5 6 7
 $k=9$ 7 1 2 3 4 5 6
 $k=10$ 6 7 1 2 3 4 5
 $k=11$ 5 6 7 1 2 3 4

$k = k \% n$
 $k \leq n \rightarrow$ Rotation Rank
periodic nature
array length

1 2 3 4 5 6 7
5 6 7 1 2 3 4
7 1 2 3 4 5 6
6 7 1 2 3 4 5
5 6 7 1 2 3 4

$k=3$

$i=5$ arr[6] = arr[5]
 $i=4$ arr[5] = arr[4]
 $i=3$ arr[4] = arr[3]
 $i=2$ arr[3] = arr[2]
 $i=1$ arr[2] = arr[1]
 $i=0$ arr[1] = arr[0]
arr[0] = item

for $i = n-2; i >= 0; i--$
arr[i+1] = arr[i]
arr[0] = item

```
public static void Rotate(int[] arr, int k) {
    int n = arr.length;
    k = k % n; // 0 to n-1
    // logic
    for (int j = 1; j <= k; j++) {
        int item = arr[n-1];
        for (int i = n-2; i >= 0; i--) {
            arr[i+1] = arr[i];
        }
        arr[0] = item;
    }
}
```

Revised Algo

1 2 3 4 5 6 7
5 6 7 1 2 3 4
7 6 5 4 3 2 1

$k=2$
30 40 10 20
9 1 2 3

$i=2$ arr[3] = arr[2]
 $i=1$ arr[2] = arr[1]
 $i=0$ arr[1] = arr[0]

1 2 3 4 5 6 7
1 2 3 4 7 6 5
4 3 2 1 7 6 5
5 6 7 1 2 3 4

1 2 3 4 5 6 7
1 2 3 4 7 6 5
4 3 2 1 7 6 5
5 6 7 1 2 3 4

1 2 3 4 5 6 7
1 2 3 4 7 6 5
4 3 2 1 7 6 5
5 6 7 1 2 3 4

2+4+3+2+5 = 16

5 3 1 2 7 4 1 6

5 3 1 2 7 4 1 6

5 3 1 2 7 4 1 6

1 2 3 4 5 6 7
1 2 3 4 7 6 5
4 3 2 1 7 6 5
5 6 7 1 2 3 4

1 2 3 4 5 6 7
1 2 3 4 7 6 5
4 3 2 1 7 6 5
5 6 7 1 2 3 4

1 2 3 4 5 6 7
1 2 3 4 7 6 5
4 3 2 1 7 6 5
5 6 7 1 2 3 4

2 1 3 4 5 6
3 6 2 4 0 1 8 0 1 4 1 2 0
7 2 0 7 2 0 2 7 9 6
2 0 7 2 0 2 7 9 6

3 1 3 4 1 6
3 6 2 4 0 1 8 0 1 4 1 2 0
7 2 0 7 2 0 2 7 9 6
2 0 7 2 0 2 7 9 6

1 2 3 4 5 6
1 2 3 4 7 6 5
4 3 2 1 7 6 5
5 6 7 1 2 3 4

1 2 3 4 5 6
1 2 3 4 7 6 5
4 3 2 1 7 6 5
5 6 7 1 2 3 4