Q7 Rotate an array.

(i) Clockwise direxn by k steps

i/p → {1, 2, 3, 4, 5, 6, 7}

k = 3

o/p → {5, 6, 7, 1, 2, 3, 4}

First of all we need to make sure that k lies blw 0 to ending index. This can be made sure by mod operation. We just have to follow 3 steps to rotate the array.

1) Reverse the whole array.
2) Reverse the left part of array i.e 0 to k-1 index.
3) Reverse the right part of array i.e k to ending index.

Dry run

{1, 2, 3, 4, 5, 6, 7} k = 3

1) Reverse the whole array

{7, 6, 5, 4, 3, 2, 1}

2) Reverse the left part of array i.e 0 to 2

{5, 6, 7, 4, 3, 2, 1}

3) Reverse the right part of array i.e 3 to 6

{5, 6, 7, 1, 2, 3, 4}

Code

```
void rotateClockwise (vector<int>& nums, int k){
    //k should be in range
    if (k > nums.size()){
        k = k % nums.size();
    }
    //Reverse array
    reverse (nums.begin(), nums.end());
    //Reverse left part
    reverse (nums.begin(), nums.begin()+k);
    //Reverse right part
    reverse (nums.begin()+k, nums.end());
    //print the array
    for (auto i: nums){
        cout << i << " ";
    }
}
```

(ii) Anticlockwise direxⁿ by k steps

i/p → {1,2,3,4,5} k = 2
o/p → {3,4,5,1,2}

First of all we need to make sure that the k lies in range 0 to end index. This can be made sure by mod operation. We just have to follow 3 steps to rotate array.

1) Reverse the whole array.
2) Reverse the left part of array i.e 0 to nums.size() − k −1 index.

3) Reverse the right part of array i.e nums.size() - k to num.size()-1 index (ending index).

Dry run
{1, 2, 3, 4, 5} k = 2

1) Reverse the whole array
{5, 4, 3, 2, 1}
2) Reverse the left part of array i.e 0 to 2nd index. 5-2-1 = 2
{3, 4, 5, 2, 1}
3) Reverse the right part of array i.e 3rd index to 4th index 5-1=4 ↔ 5-2=3
{3, 4, 5, 1, 2}

Code

```
void rotateAntiClockwise (vector<int>& nums, int k)
{
        // k should be in range
        if (k > nums.size()) {
                k = k % nums.size();
        }
        // Reverse the array
        reverse (nums.begin(), nums.end());
        // Reverse left part of array
        reverse (nums.begin(), nums.begin() +
                                nums.size() - k);
        // Reverse right part of array
        reverse (nums.begin() + nums.size() - k,
                                nums.end()
                );
        // Print the array
```

```
        for (auto i: nums) {
            cout << i << " ";
        }
    }
```

Time complexity = O(n)

Space complexity = O(1)