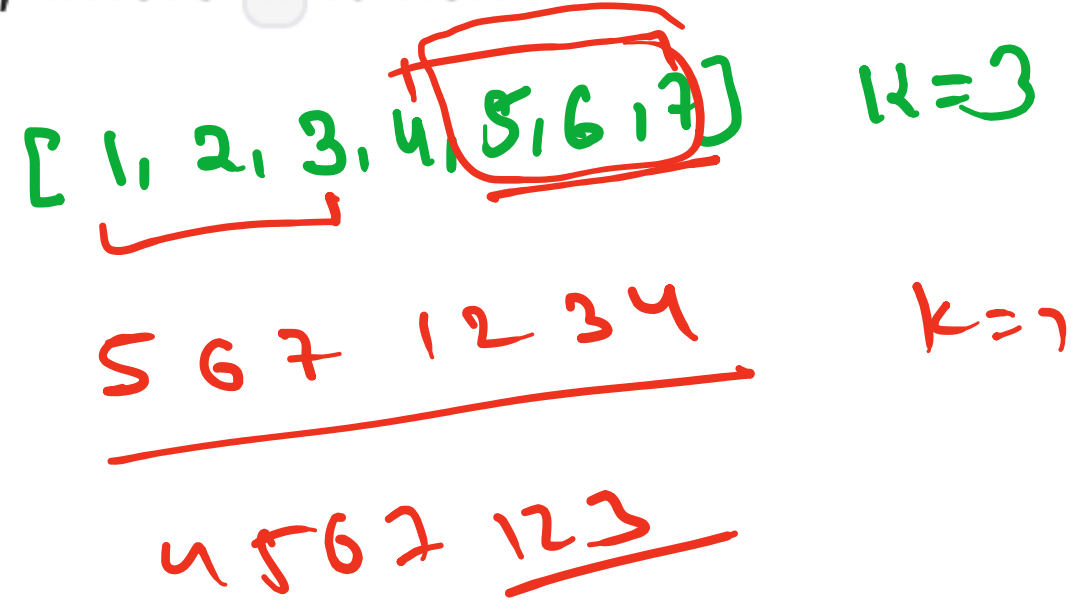
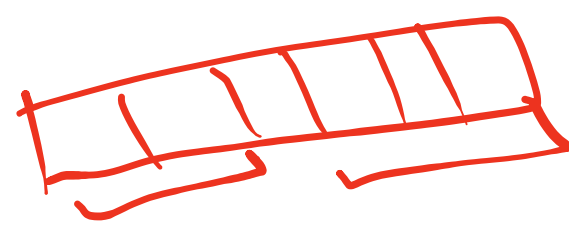
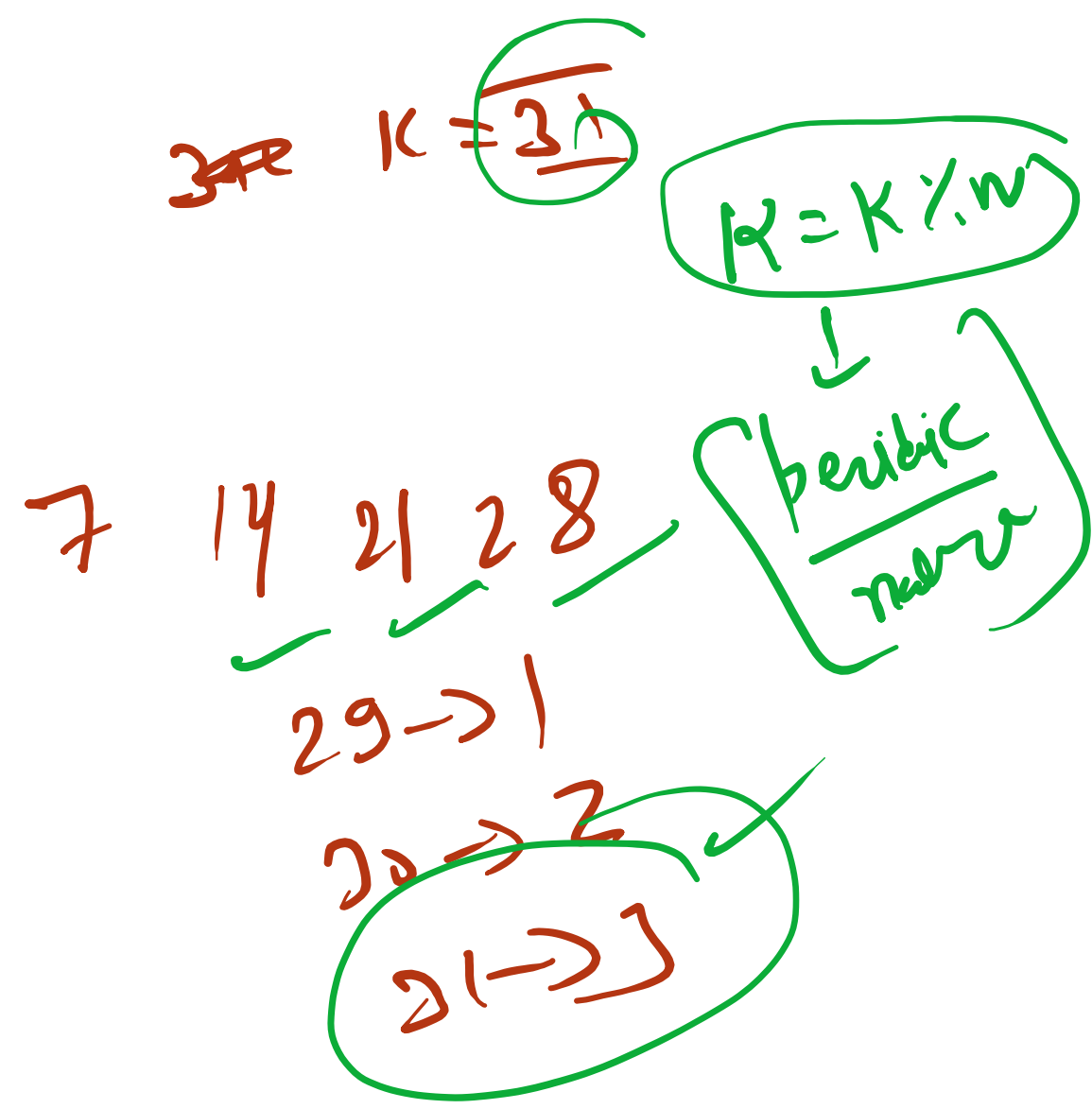
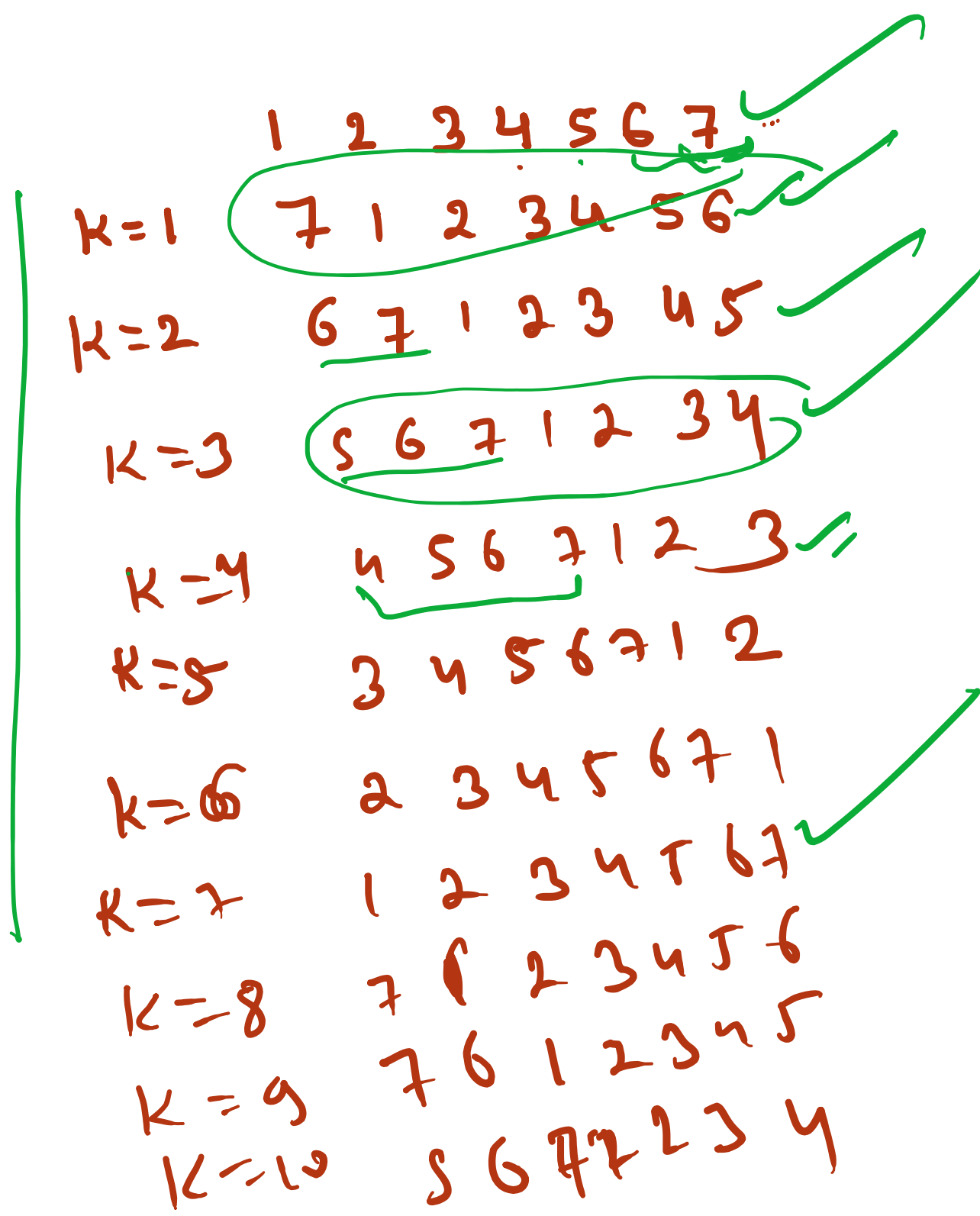


Given an integer array `nums`, rotate the array to the right by `k` steps, where `k` is non-negative.

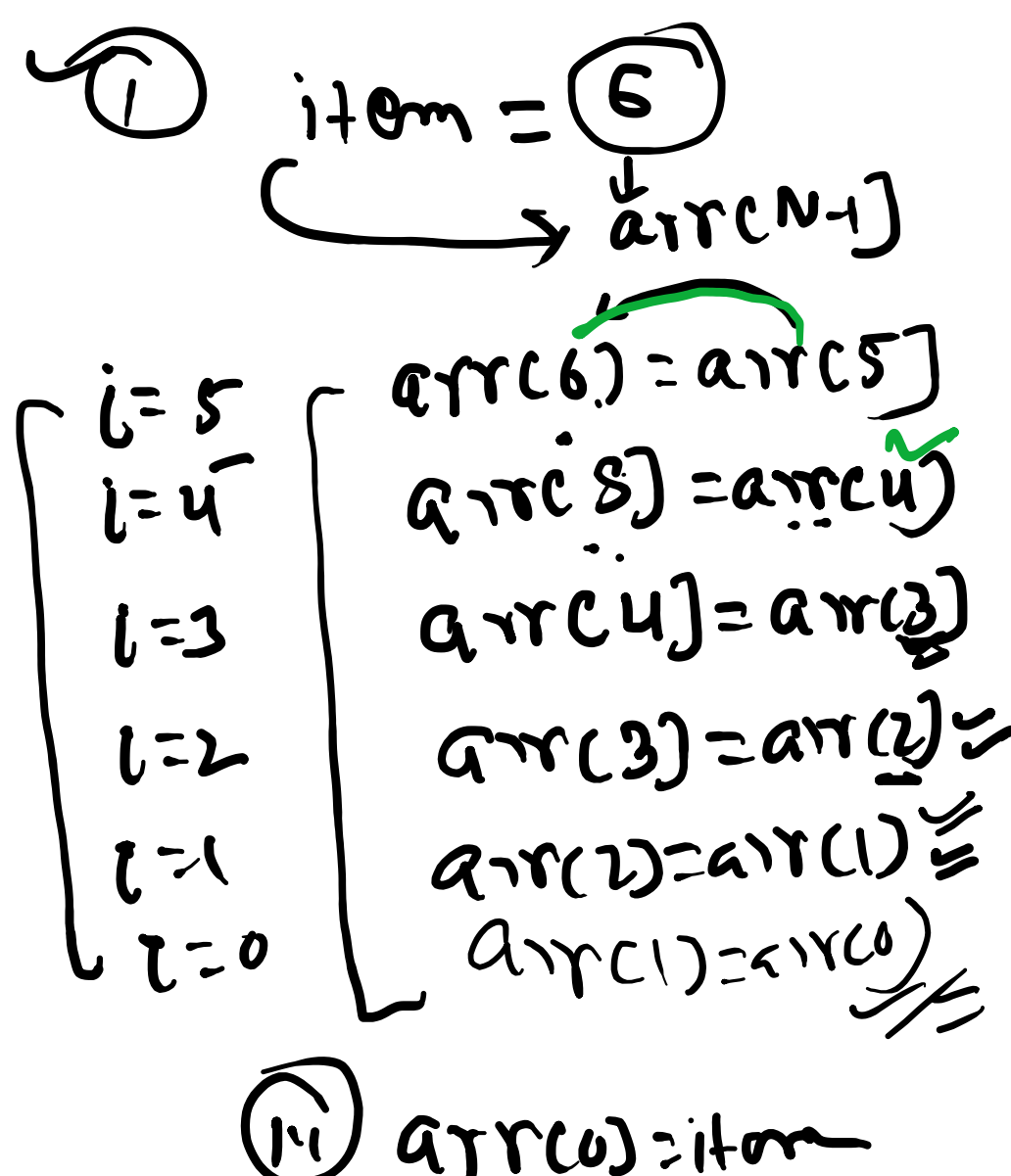
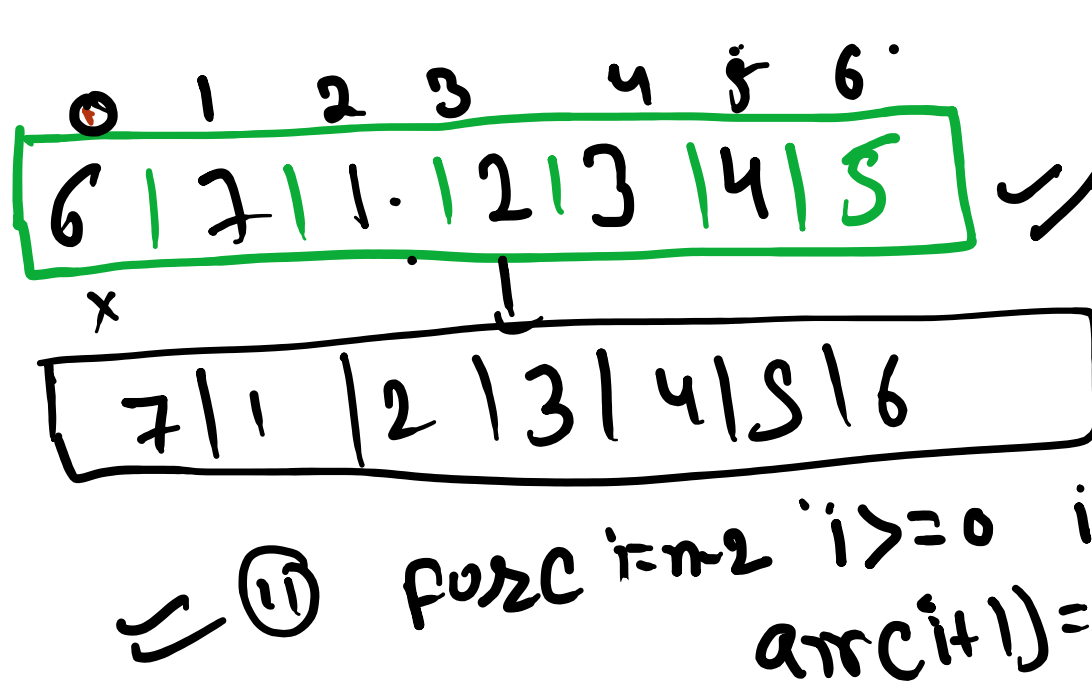


Example 1:

Input: `nums = [1,2,3,4,5,6,7]`, `k = 3`

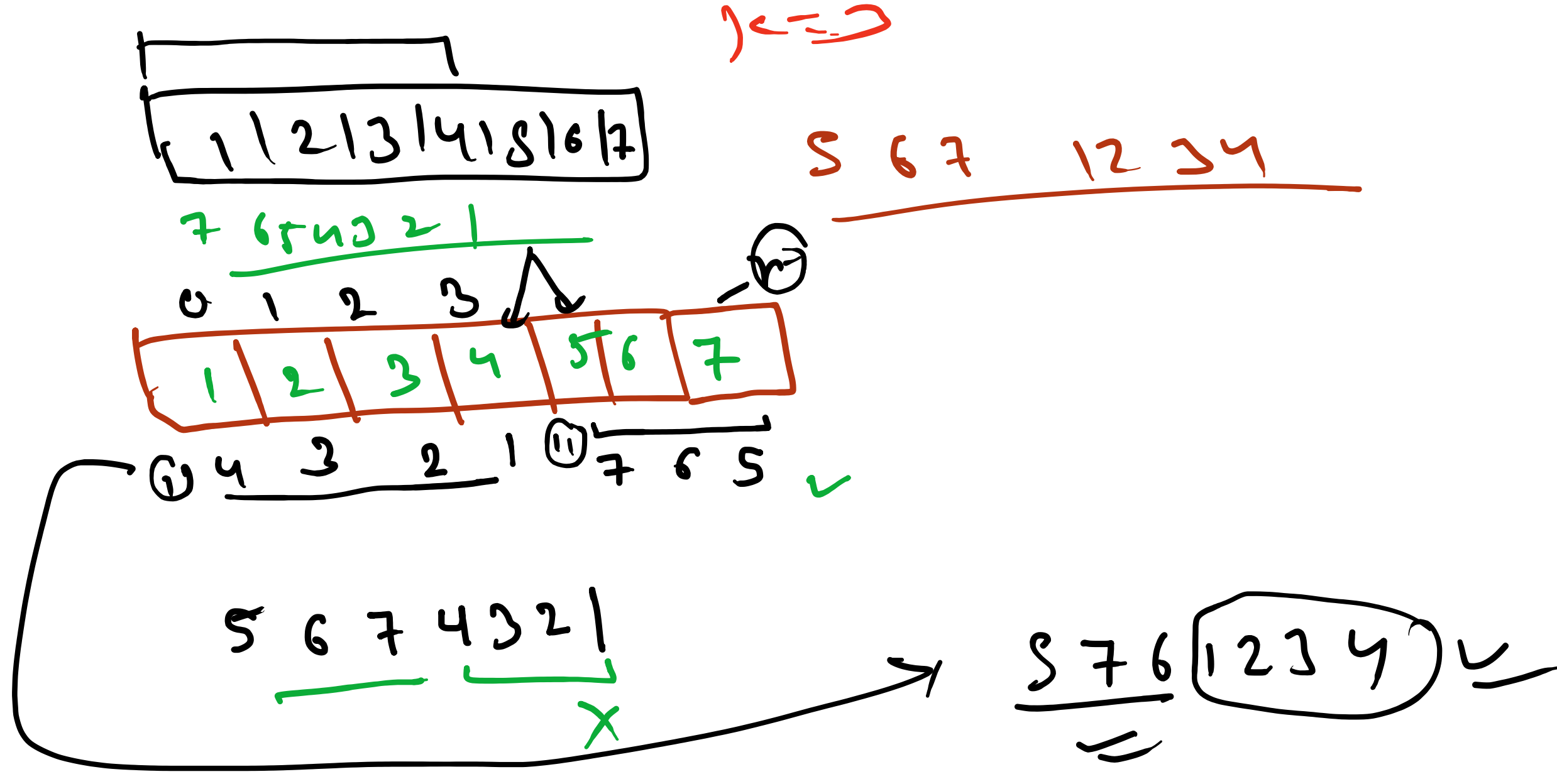


$arr[i] = arr[i+1]$   
 $arr[n-1] = arr[0]$

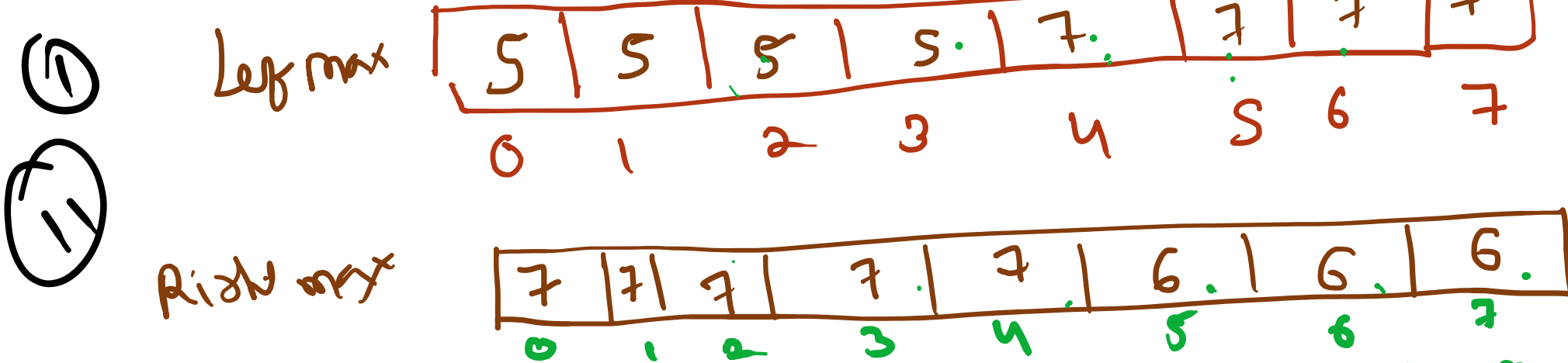
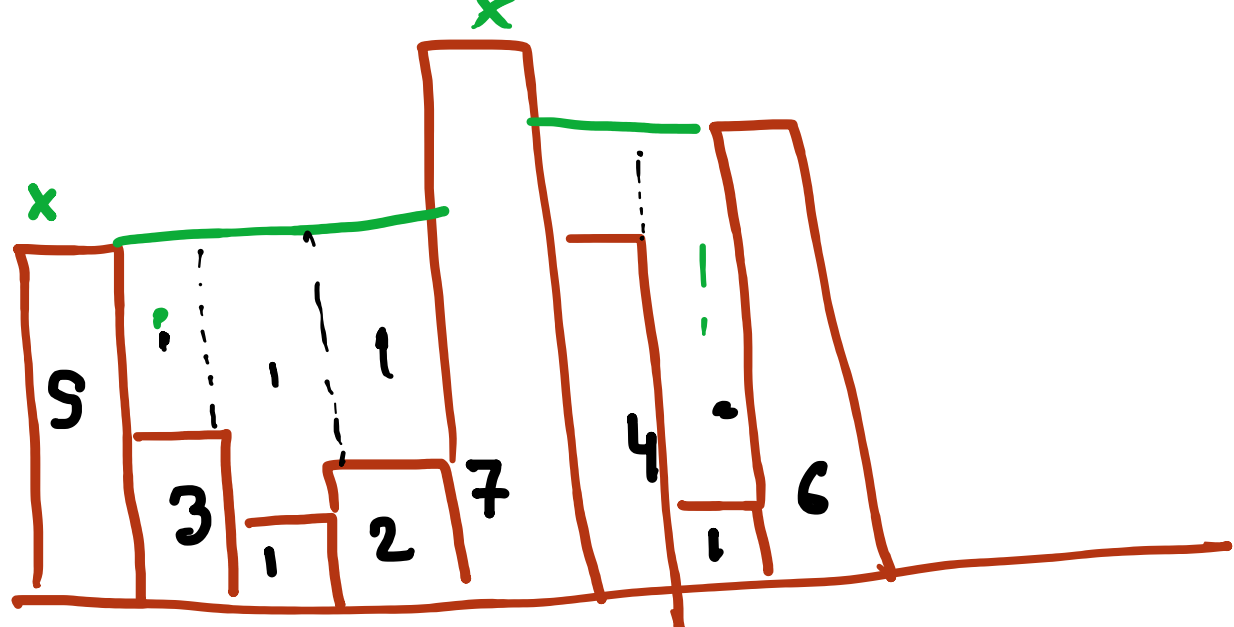
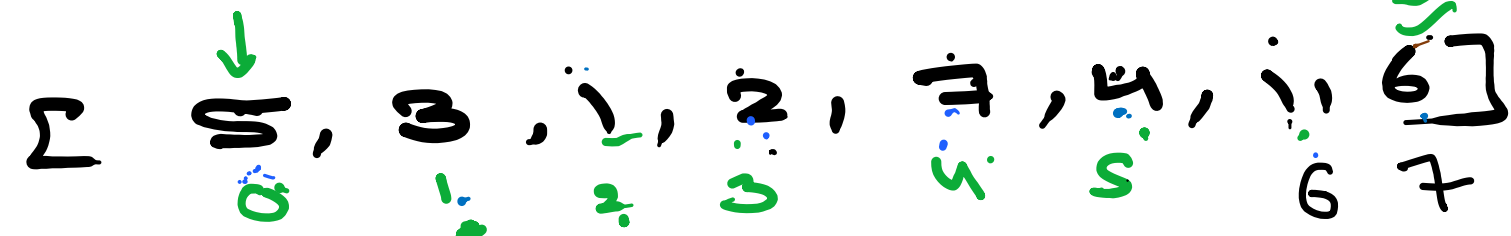


```
public static void Rotate(int[] arr, int k) {  
    int n = arr.length;  
    k = k % n;  
    for (int i = 1; i <= k; i++) {  
        int item = arr[n - 1];  
        for (int j = n - 2; j >= 0; j--) {  
            arr[j + 1] = arr[j];  
        }  
        arr[0] = item;  
    }  
}
```

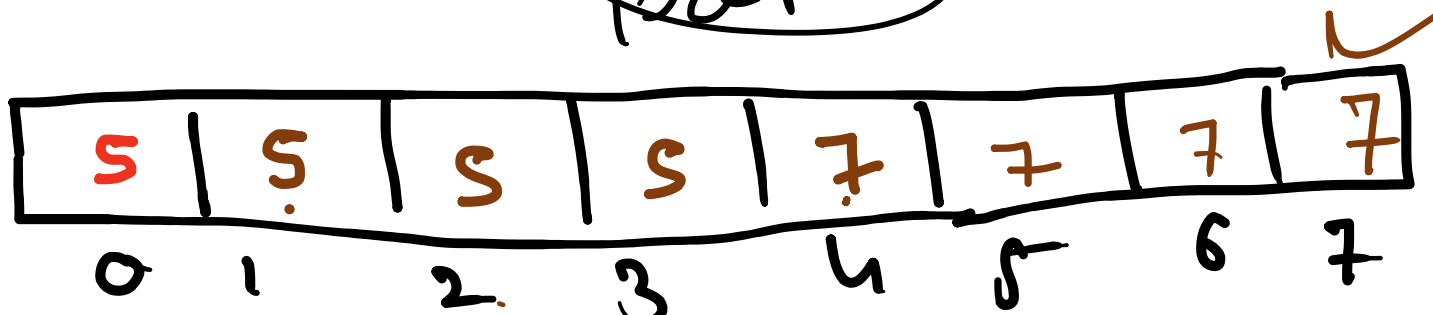
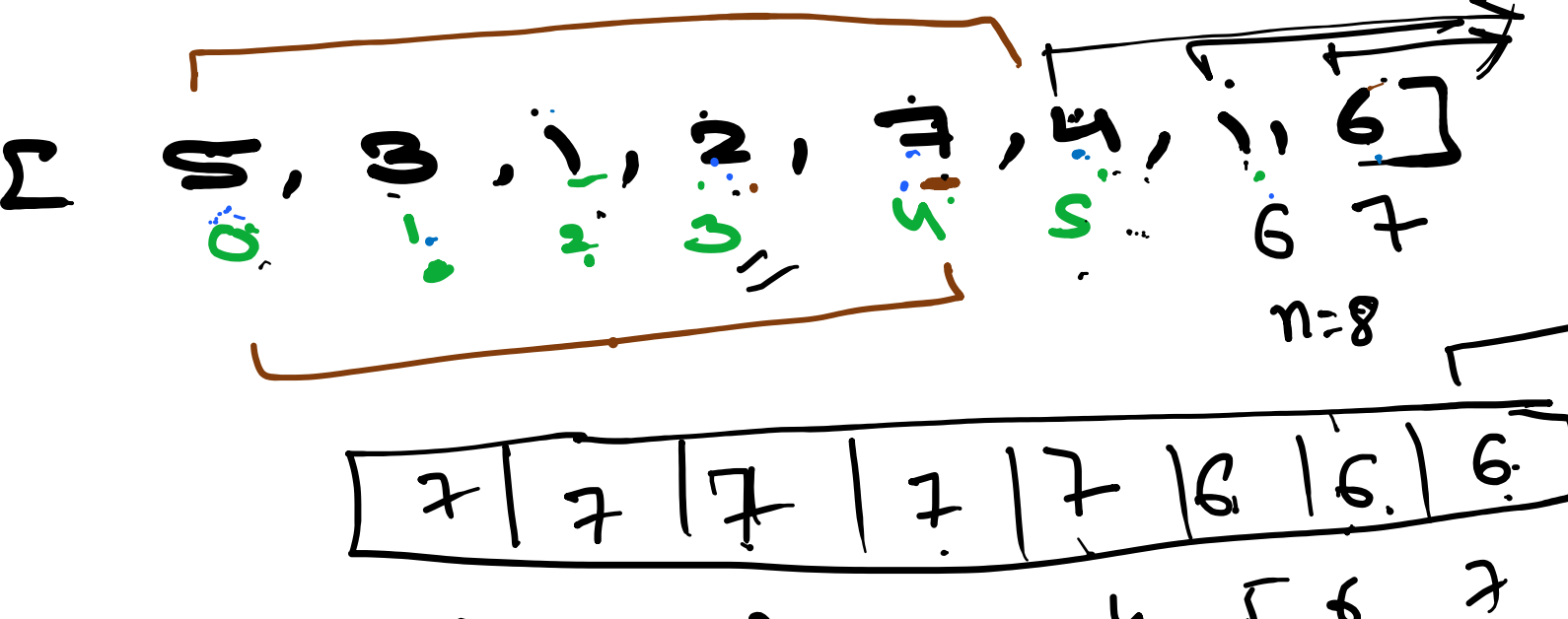
Last  $k = n - 1 - (n - k - 1)$   
Starting  $n - k = 0$  to  $n - k - 1$



Given `n` non-negative integers representing an elevation map where the width of each bar is `1`, compute how much water it can trap after raining.



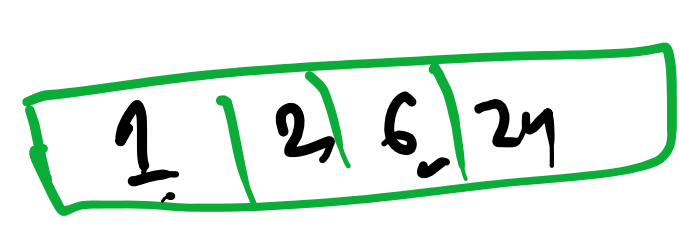
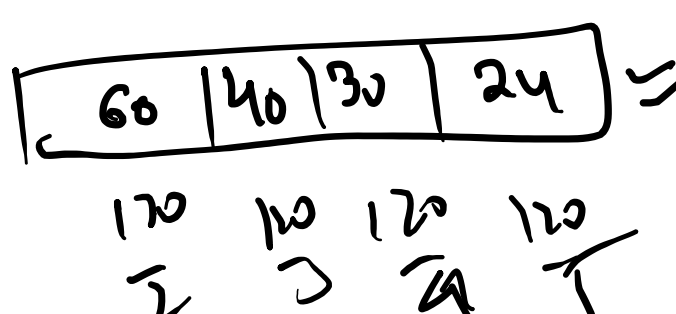
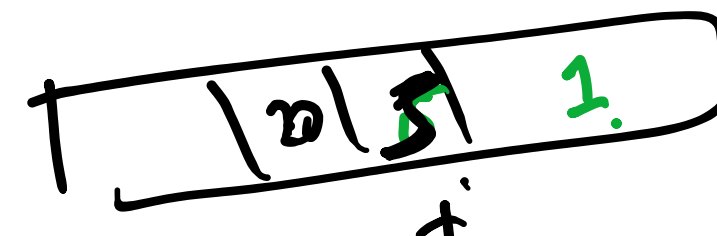
Handwritten code for calculating the water trapped at each index `i` using the left and right maximums.



Handwritten code for calculating the water trapped at each index `i` using the left and right maximums.

Handwritten code for calculating the water trapped at each index `i` using the left and right maximums.

$[2, 3, 4, 5]$



Handwritten code for calculating the water trapped at each index `i` using the left and right maximums.

Handwritten code for calculating the water trapped at each index `i` using the left and right maximums.