* Remove duplicates from a sorted array :-

nums = [0, 0, 1, 1, 1, 2, 2, 3, 3, 4]

O/p :- k = 5 , nums = [0, 1, 2, 3, 4, _, _, _, _, _]

→ Arrays will always be sorted in ascending order.

→ We have to do it in-place. So we cannot create a new arr.

* Two pointer approach

$$\begin{array}{lccccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \text{nums} = [& 0, & 0, & 1, & 1, & 1, & 2, & 2, & 3, & 3, & 4 &] \end{array}$$

→ The input array has to be modified in-place such that the first k elements are unique.

→ So, initially k = 0 because we don't know how many duplicates are there in the array.

$$\begin{array}{cccccccccc} [0, & 0, & 1, & 1, & 1, & 2, & 2, & 3, & 3, & 4] \\ \uparrow & & & & & & & & & \\ k & & & & & & & & & \end{array}$$

→ k represents the position of a unique element.

→ Since we don't know how many dups exists, we initialized k = 0 which means that the first element is unique.

→ Now we need to iterate over the array from index 1 & compare with element at k.

$$\begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ [0, & 0, & 1, & 1, & 1, & 2, & 2, & 3, & 3, & 4] \\ \uparrow & \uparrow & & & & & & & & \\ k & i & & & & & & & & \end{array}$$

→ Here, nums[k] == nums[i]. So we have found a duplicate.

→ Ideally, the element next to k should be unique.

→ So we search for unique element & put it just next to k.

$$\begin{array}{cccccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & \\ [& 0, & 0, & 1, & 1, & 1, & 2, & 2, & 3, & 3, & 4 &] \\ & \uparrow & & \uparrow & & & & & & & & \\ & k & & i & & & & & & & & \end{array}$$

→ Found unique, so putting it next to k.

$$\begin{array}{cccccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & \\ [& 0, & 1, & 1, & 1, & 1, & 2, & 2, & 3, & 3, & 4 &] \\ & & \uparrow & \uparrow & & & & & & & & \\ & & k & i & & & & & & & & \end{array}$$

→ Initially k=0 which represented that the 0th element is unique.

→ Now we have place 1 next to k on index 1.

→ This means that we have unique elements from index [0, 1].

→ So, we point k to 1. & so on.

→ Thus, every unique element will be placed next to k & k will point to the updated unique element.

→ Once the loop is completed, we know that the elements from range [0, k] are unique.

→ Hence, we return (k+1) because 0-indexed array.

TC :- O(n)

SC :- O(1)