October 2

Problem: Move Zeroes

Problem Statement: Given an integer array nums, move all 0s to the end of it while maintaining the relative order of the non-zero elements. Note that you must do this in-place without making a copy of the array.

Link to problem:

https://leetcode.com/problems/move-zeroes/

Example 1:

Input: nums = [0,1,0,3,12]

Output: [1,3,12,0,0]

Example 2:

Input: nums = [0]

Output: [0]

Solution:

```
class Solution {

   public void moveZeroes(int[] nums) {
      int nonZero = 0, zero = 0; // Initialize pointers for non-zero and zero elements

      while(nonZero < nums.length) { // Iterate through the array

         if(nums[nonZero] != 0) {
            // Check if the current element is non-zero

            // Swap non-zero element with the element at 'zero' index
            int temp = nums[zero];
            nums[zero] = nums[nonZero];
            nums[nonZero] = temp;

            zero++; // Move the zero pointer to the next position

         }
         nonZero++; // Move the non-zero pointer to the next position

      }
   }
}
```

Explanation:

- We use two pointers: `nonZero` to traverse the array and `zero` to keep track of the position where the next non-zero element should be placed.
- As we iterate through the array, when we find a non-zero element, we swap it with the element at the `zero` pointer.
- After the swap, we increment the `zero` pointer to move to the next position for potential non-zero elements.
- The `nonZero` pointer always moves forward, ensuring we check every element in the array.
- This approach maintains the order of non-zero elements while moving all `0`s to the end of the array.

Time Complexity:

- O(n), where n is the number of elements in the array. We traverse the array once.

Space Complexity:

- O(1), as we are using only a few extra variables (nonZero and zero) for storage.
