LeetCode Solutions

First and Last Position of Element in Sorted Array:

```
class Solution:
    def searchRange(self, nums, target):
        def findLeft(nums, target):
            left, right = 0, len(nums) - 1
            while left <= right:
                mid = (left + right) // 2
                if nums[mid] < target:
                    left = mid + 1
                else:
                    right = mid - 1
            return left

        def findRight(nums, target):
            left, right = 0, len(nums) - 1
            while left <= right:
                mid = (left + right) // 2
                if nums[mid] <= target:
                    left = mid + 1
                else:
                    right = mid - 1
            return right

        left = findLeft(nums, target)
        right = findRight(nums, target)
        # Check if the target is in the range found
        if left <= right and left < len(nums) and nums[left] == target:
            return [left, right]
        else:
            return [-1, -1]
```

Two Sum:

```
class Solution:
    def twoSum(self, nums, target):
        num_to_index = {}  # Hash map to store number and its index

        for i, num in enumerate(nums):
            complement = target - num  # Calculate complement
            if complement in num_to_index:
                # If complement is in the map, return the indices
                return [num_to_index[complement], i]
            # Store the current number and its index in the map
            num_to_index[num] = i
```

Remove Element:

```
class Solution:
    def removeElement(self, nums, val):
        # Initialize a pointer for the position of elements not equal to val
        i = 0
        # Iterate through the array
        for num in nums:
            # If the current element is not equal to val, we keep it
            if num != val:
                nums[i] = num
                i += 1
        # i now holds the count of elements not equal to val
        return i
```

Next Permutation:

```
class Solution:
    def nextPermutation(self, nums):
        # Step 1: Find the longest decreasing suffix
        i = len(nums) - 2
        while i >= 0 and nums[i] >= nums[i + 1]:
            i -= 1

        # Step 2: If the entire array is in descending order, reverse it
        if i == -1:
            nums.reverse()
            return

        # Step 3: Find the next largest element in the suffix and swap
        j = len(nums) - 1
        while nums[j] <= nums[i]:
            j -= 1
        nums[i], nums[j] = nums[j], nums[i]

        # Step 4: Reverse the suffix
```

```
nums[i + 1:] = reversed(nums[i + 1:])
```