In [13]:

```
**Q1.** Given an array of integers nums and an integer target, return indices of the two

You may assume that each input would have exactly one solution, and you may not use the s

You can return the answer in any order.

**Example:**
Input: nums = [2,7,11,15], target = 9
Output0 [0,1]

**Explanation:** Because nums[0] + nums[1] == 9, we return [0, 1][
```

```
f twoSum(nums, target):
      hashmap = {}
      for i in range(len(nums)):
          hashmap[nums[i]] = i
      for i in range(len(nums)):
          complement = target - nums[i]
          if complement in hashmap and hashmap[complement] != i:
              return [i, hashmap[complement]]


ms = [2,7,11,15]
rget = 9
= twoSum(nums , target)
int(x)
```

```
[0, 1]
```

In [2]:

```
# **Q2.** Given an integer array nums and an integer val, remove all occurrences of val

# Consider the number of elements in nums which are not equal to val be k, to get accept

# - Change the array nums such that the first k elements of nums contain the elements wh
# - Return k.

# **Example :**
# Input: nums = [3,2,2,3], val = 3
# Output: 2, nums = [2,2,_*,_*]

# **Explanation:** Your function should return k = 2, with the first two elements of num


def removeElement(nums, val):
        # Counter for keeping track of elements other than val
        count = 0
        # empty list for append element without value
        l = []
        # Loop through all the elements of the array
        for i in range(len(nums)):
            # If the element is not val
            if nums[i] != val:
                nums[count] = nums[i]
                count += 1
                l.append(nums[i])
        return count , l

nums = [3,2,2,3]
val = 3
x = removeElement(nums,val)
print(x)
```

```
(2, [2, 2])
```

In [3]:

```
# **Q3.** Given a sorted array of distinct integers and a target value, return the index

# You must write an algorithm with O(log n) runtime complexity.

# **Example 1:**
# Input: nums = [1,3,5,6], target = 5

# Output: 2



def searchInsert(nums, target):
        start, end = 0, len(nums) - 1
        ans = len(nums) # Default answer when target is greater than all elements

        while start <= end:
            mid = (start + end) // 2

            if nums[mid] == target:
                return mid
            elif nums[mid] < target:
                start = mid + 1
            else:
                ans = mid # Update the answer to the current index
                end = mid - 1

        return ans


nums = [1,3,5,6]
target = 5
x = searchInsert(nums,target)
print(x)
```

2

In [4]:

```
# **Q4.** You are given a large integer represented as an integer array digits, where ea

# Increment the large integer by one and return the resulting array of digits.

# **Example 1:**
# Input: digits = [1,2,3]
# Output: [1,2,4]

# **Explanation:** The array represents the integer 123.

# Incrementing by one gives 123 + 1 = 124.
# Thus, the result should be [1,2,4].



def plusOne(digits):
        strings = ""
        for number in digits:
            strings += str(number)

        temp = str(int(strings) +1)

        return [int(temp[i]) for i in range(len(temp))]

digits = [1,2,3]
x = plusOne(digits)
print(x)
```

```
[1, 2, 4]
```

In [5]:

```
#
# **Q5.** You are given two integer arrays nums1 and nums2, sorted in non-decreasing ord

Merge nums1 and nums2 into a single array sorted in non-decreasing order.

The final sorted array should not be returned by the function, but instead be stored ins

**Example 1:**
Input: nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3
Output: [1,2,2,3,5,6]

**Explanation:** The arrays we are merging are [1,2,3] and [2,5,6].
The result of the merge is [1,2,2,3,5,6] with the underlined elements coming from nums1.

#

def merge(nums1, m, nums2, n):
        i = m - 1
        j = n - 1
        k = m + n - 1

        while j >= 0:
            if i >= 0 and nums1[i] > nums2[j]:
                nums1[k] = nums1[i]
                i -= 1
            else:
                nums1[k] = nums2[j]
                j -= 1
            k -= 1

nums1 = [1,2,3,0,0,0]
m = 3
nums2 = [2,5,6]
n = 3
x = merge(nums1, m, nums2, n)
print(nums1)
```

```
[1, 2, 2, 3, 5, 6]
```

In [6]:

```
# **Q6.** Given an integer array nums, return true if any value appears at least twice i

# **Example 1:**
# Input: nums = [1,2,3,1]

# Output: true


def containsDuplicate(nums):
    return len(set(nums)) != len(nums)

nums = [1,2,3,1]
x = containsDuplicate(nums)
print(x)
```

```
True
```

In [7]:

```
#  **Q7.** Given an integer array nums, move all 0's to the end of it while maintaining

# Note that you must do this in-place without making a copy of the array.

# **Example 1:**
# Input: nums = [0,1,0,3,12]
# Output: [1,3,12,0,0]


def moveZeroes(nums):
        n=len(nums)
        for i in range(0,n):
            if nums[i]==0:
                nums.remove(nums[i])
                nums.append(0)
        return nums

nums = [0,1,0,3,12]
x = moveZeroes(nums)
print(x)
```

```
[1, 3, 12, 0, 0]
```

In [8]:

```
#  **Q8.** You have a set of integers s, which originally contains all the numbers from 

# You are given an integer array nums representing the data status of this set after the

# Find the number that occurs twice and the number that is missing and return them in th

# **Example 1:**
# Input: nums = [1,2,2,4]
# Output: [2,3]

def findErrorNums(nums):

        n, a, b = len(nums), sum(nums), sum(set(nums))

        s = n*(n+1)//2

        return [a-b, s-b]

nums = [1,2,2,4]
x = findErrorNums(nums)
print(x)
```

```
[2, 3]
```

In [ ]: