

Q1. Given an array of integers `nums` and an integer `target`, return indices of the two numbers such that they add up to `target`.

You may assume that each input would have exactly one solution, and you may not use the same element twice.

You can return the answer in any order.

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

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

PROGRAM:-

```
def twoSum(nums, target):  
    hashMap = {}  
  
    for i, num in enumerate(nums):  
        diff = target - num  
  
        if diff in hashMap:  
            return [hashMap[diff], i]  
  
        hashMap[num] = i  
  
    return []  
  
nums = [2, 7, 11, 15]  
  
target = 9  
  
result = twoSum(nums, target)  
  
print(result)
```

Output: `[0, 1]`

Q2. Given an integer array nums and an integer val, remove all occurrences of val in nums in-place. The order of the elements may be changed. Then return the number of elements in nums which are not equal to val.

Consider the number of elements in nums which are not equal to val be k, to get accepted, you need to do the following things:

- Change the array nums such that the first k elements of nums contain the elements which are not equal to val. The remaining elements of nums are not important as well as the size of nums.
- 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 nums being 2. It does not matter what you leave beyond the returned k (hence they are underscores)

PROGRAM:-

```
def removeElement(nums, val):  
  
    i = 0  
  
    for j in range(len(nums)):  
  
        if nums[j] != val:  
  
            nums[i] = nums[j]  
  
            i += 1  
  
    return i  
  
nums = [3, 2, 2, 3]  
  
val = 3  
  
result = removeElement(nums, val)  
  
print(result)  
  
print(nums)
```

OUTPUT:-

Output: 2

Output: [2, 2, 2, 3] (remaining elements are not important)

Q3. Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

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

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

Output: 2

PROGRAM:-

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

```
nums = [1, 3, 5, 6]

target = 5

result = searchInsert(nums, target)

print(result)
```

Output: 2

Q4. You are given a large integer represented as an integer array `digits`, where each `digits[i]` is the *i*th digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading 0's.

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]`.

PROGRAM:-

```
def plusOne(digits):

    carry = 1

    for i in range(len(digits) - 1, -1, -1):

        digits[i] += carry

        if digits[i] < 10:

            carry = 0

            break

        digits[i] %= 10

    if carry:
```

```

        digits.insert(0, carry)

    return digits

digits = [1, 2, 3]

result = plusOne(digits)

print(result)

```

Output: [1, 2, 4]

Q5. You are given two integer arrays `nums1` and `nums2`, sorted in non-decreasing order, and two integers `m` and `n`, representing the number of elements in `nums1` and `nums2` respectively.

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 inside the array `nums1`. To accommodate this, `nums1` has a length of `m + n`, where the first `m` elements denote the elements that should be merged, and the last `n` elements are set to 0 and should be ignored. `nums2` has a length of `n`.

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`.

PROGRAM:-

```

def merge(nums1, m, nums2, n):

    p1, p2, p = m - 1, n - 1, m + n - 1

    while p1 >= 0 and p2 >= 0:

        if nums1[p1] > nums2[p2]:

            nums1[p] = nums1[p1]

            p1 -= 1

```

```

        else:
            nums1[p] = nums2[p2]

            p2 -= 1

        p -= 1

    nums1[:p2 + 1] = nums2[:p2 + 1]

    return nums1

nums1 = [1, 2, 3, 0, 0, 0]

m = 3

nums2 = [2, 5, 6]

n = 3

result = merge(nums1, m, nums2, n)

print(result)

```

Output: [1, 2, 2, 3, 5, 6]

Q6. Given an integer array `nums`, return `true` if any value appears at least twice in the array, and return `false` if every element is distinct.

Example 1: Input: `nums = [1,2,3,1]`

Output: `true`

PROGRAM:-

```

def containsDuplicate(nums) :

    seen = set()

    for num in nums:

```

```

        if num in seen:

            return True

        seen.add(num)

    return False

nums = [1, 2, 3, 1]

result = containsDuplicate(nums)

print(result)

```

Output: True

Q7. Given an integer array nums, move all 0's to the end of it while maintaining the relative order of the nonzero elements.

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]

PROGRAM:-

```

def moveZeroes(nums):

    left, right = 0, 0

    while right < len(nums):

        if nums[right] != 0:

            nums[left], nums[right] = nums[right],
nums[left]

            left += 1

        right += 1

    nums[left:] = [0] * (len(nums) - left)

```

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

```
moveZeroes(nums)
```

```
print(nums)
```

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

Q8. You have a set of integers *s*, which originally contains all the numbers from 1 to *n*. Unfortunately, due to some error, one of the numbers in *s* got duplicated to another number in the set, which results in repetition of one number and loss of another number.

You are given an integer array *nums* representing the data status of this set after the error.

Find the number that occurs twice and the number that is missing and return them in the form of an array.

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

PROGRAM:-

```
def findErrorNums(nums):  
  
    numSet = set()  
  
    duplicate, missing = 0, 0  
  
    for num in nums:  
  
        if num in numSet:  
  
            duplicate = num  
  
        numSet.add(num)  
  
    for i in range(1, len(nums) + 1):  
  
        if i not in numSet:  
  
            missing = i
```



```
        return [duplicate, missing]

nums = [1, 2, 2, 4]

result = findErrorNums(nums)

print(result)
```

OUTPUT:-

```
[2, 3]
```