Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
def arraysIntersection(arr1, arr2, arr3):
    return sorted(set(arr1) & set(arr2) & set(arr3))


arr1 = [1, 2, 3, 4, 5]
arr2 = [1, 2, 5, 7, 9]
arr3 = [1, 3, 4, 5, 8]
result = arraysIntersection(arr1, arr2, arr3)
print(result)
```

input

```
[1, 5]


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
def maxChunksToSorted(arr):
    max_val = result = 0
    for i, val in enumerate(arr):
        max_val = max(max_val, val)
        if max_val == i:
            result += 1
    return result

arr = [4, 3, 2, 1, 0]
result = maxChunksToSorted(arr)
print(result)


```

input

```
1


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify

main.py

```python
from collections import Counter

def frequencySort(s):
    return ''.join(char * freq for char, freq in Counter(s).most_common())
s = "tree"
result = frequencySort(s)
print(result)

```

input

```
eetr


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def insertionSortList(head):
    dummy = ListNode(0)
    current = head
    while current:
        prev = dummy
        while prev.next and prev.next.val < current.val:
            prev = prev.next
        next_temp = current.next
        current.next = prev.next
        prev.next = current
        current = next_temp
    return dummy.next
head = ListNode(4, ListNode(2, ListNode(1, ListNode(3))))
result = insertionSortList(head)
while result:
    print(result.val, end=' -> ')
    result = result.next

```

input

```
1 -> 2 -> 3 -> 4 ->

...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def sortedArrayToBST(nums):
    if not nums:
        return None
    mid = len(nums) // 2
    root = TreeNode(nums[mid])
    root.left = sortedArrayToBST(nums[:mid])
    root.right = sortedArrayToBST(nums[mid+1:])
    return root
nums = [-10, -3, 0, 5, 9]
result = sortedArrayToBST(nums)
def preOrder(node):
    if node:
        print(node.val, end=' ')
        preOrder(node.left)
        preOrder(node.right)
preOrder(result)

```

input

```
0 -3 -10 9 5

...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
def merge(nums1, m, nums2, n):
    while m > 0 and n > 0:
        if nums1[m-1] > nums2[n-1]:
            nums1[m+n-1] = nums1[m-1]
            m -= 1
        else:
            nums1[m+n-1] = nums2[n-1]
            n -= 1
    nums1[:n] = nums2[:n]

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

```

input

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


...Program finished with exit code 0
Press ENTER to exit console.
```

main.py

```python
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def deleteDuplicates(head):
    current = head
    while current and current.next:
        if current.val == current.next.val:
            current.next = current.next.next
        else:
            current = current.next
    return head
head = ListNode(1, ListNode(1, ListNode(2)))
result = deleteDuplicates(head)
while result:
    print(result.val, end=' -> ')
    result = result.next

```

input

```
1 -> 2 ->

...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
def sortColors(nums):
    lo, hi, i = 0, len(nums) - 1, 0
    while i <= hi:
        if nums[i] == 0:
            nums[i], nums[lo] = nums[lo], nums[i]
            lo += 1
        elif nums[i] == 2:
            nums[i], nums[hi] = nums[hi], nums[i]
            hi -= 1
        i += 1

nums = [2, 0, 2, 1, 1, 0]
sortColors(nums)
print(nums)

```

input

```
[0, 0, 1, 1, 2, 2]


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
def searchRange(nums, target):
    def find(nums, target):
        lo, hi = 0, len(nums) - 1
        while lo <= hi:
            mid = (lo + hi) // 2
            if nums[mid] < target:
                lo = mid + 1
            else:
                hi = mid - 1
        return lo

    start = find(nums, target)
    end = find(nums, target + 1) - 1
    return (start, end) if start <= end else [-1, -1]

nums = [5, 7, 7, 8, 8, 10]
target = 8
result = searchRange(nums, target)
print(result)

```

input

```
(3, 4)


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | {} Beautify | Language Python 3

main.py

```python
def search(nums, target):
    lo, hi = 0, len(nums) - 1
    while lo <= hi:
        mid = (lo + hi) // 2
        if nums[mid] == target:
            return mid
        if nums[lo] <= nums[mid]:
            if nums[lo] <= target < nums[mid]:
                hi = mid - 1
            else:
                lo = mid + 1
        else:
            if nums[mid] < target <= nums[hi]:
                lo = mid + 1
            else:
                hi = mid - 1
    return -1

nums = [4, 5, 6, 7, 0, 1, 2]
target = 0
index = search(nums, target)
print(index)

```

input

```
4


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify

main.py

```python
def removeDuplicates(nums):
    if not nums:
        return 0
    j = 0
    for i in range(1, len(nums)):
        if nums[i] != nums[j]:
            j += 1
            nums[j] = nums[i]
    return j + 1

nums = [1, 1, 2]
new_length = removeDuplicates(nums)
print(nums[:new_length])

```

input

```
[1, 2]


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def mergeTwoLists(l1, l2):
    dummy = ListNode(0)
    tail = dummy
    while l1 and l2:
        if l1.val < l2.val:
            tail.next, l1 = l1, l1.next
        else:
            tail.next, l2 = l2, l2.next
        tail = tail.next
    tail.next = l1 if l1 else l2
    return dummy.next
l1 = ListNode(1, ListNode(2, ListNode(4)))
l2 = ListNode(1, ListNode(3, ListNode(4)))
result = mergeTwoLists(l1, l2)
while result:
    print(result.val, end=' -> ')
    result = result.next

```

input

```
1 -> 1 -> 2 -> 3 -> 4 -> 4 ->

...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
def merge(nums1, m, nums2, n):
    while m > 0 and n > 0:
        if nums1[m-1] > nums2[n-1]:
            nums1[m+n-1] = nums1[m-1]
            m -= 1
        else:
            nums1[m+n-1] = nums2[n-1]
            n -= 1
    nums1[:n] = nums2[:n]

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

```

input

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


...Program finished with exit code 0
Press ENTER to exit console.
```

Run | Debug | Stop | Share | Save | { } Beautify | Language Python 3

main.py

```python
import heapq
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def mergeKLists(lists):
    heap = [(lst.val, idx, lst) for idx, lst in enumerate(lists) if lst]
    heapq.heapify(heap)
    dummy = ListNode(0)
    tail = dummy
    while heap:
        val, idx, node = heapq.heappop(heap)
        tail.next = node
        tail = tail.next
        if node.next:
            heapq.heappush(heap, (node.next.val, idx, node.next))
    return dummy.next
l1 = ListNode(1, ListNode(4, ListNode(5)))
l2 = ListNode(1, ListNode(3, ListNode(4)))
l3 = ListNode(2, ListNode(6))
lists = [l1, l2, l3]
result = mergeKLists(lists)
while result:
    print(result.val, end=' -> ')
    result = result.next

```

input

```
1 -> 1 -> 2 -> 3 -> 4 -> 4 ->

...Program finished with exit code 0
Press ENTER to exit console.
```

File Explorer