In [4]:

```
#1. Merge Two Sorted Lists
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
def mergeTwoLists(list1, list2):
    dummy = ListNode()
    tail = dummy
    while list1 and list2:
        if list1.val < list2.val:
            tail.next = list1
            list1 = list1.next
        else:
            tail.next = list2
            list2 = list2.next
        tail = tail.next

    if list1:
        tail.next = list1
    elif list2:
        tail.next = list2

    return dummy.next

list1 = ListNode(1, ListNode(2, ListNode(4)))
list2 = ListNode(1, ListNode(3, ListNode(4)))
merged = mergeTwoLists(list1, list2)
while merged:
    print(merged.val, end=' ')
    merged = merged.next
```

```
1 1 2 3 4 4
```

In [5]:

```
#2. Merge k Sorted Lists
from heapq import heappop, heappush

class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
def mergeKLists(lists):
    min_heap = []
    dummy = ListNode()
    current = dummy
    for index, node in enumerate(lists):
        if node:
            heappush(min_heap, (node.val, index, node))
    while min_heap:
        val, index, node = heappop(min_heap)
        current.next = node
        current = current.next
        if node.next:
            heappush(min_heap, (node.next.val, index, node.next))

    return dummy.next

lists = [ListNode(1, ListNode(4, ListNode(5))), ListNode(1, ListNode(3,
merged = mergeKLists(lists)
while merged:
    print(merged.val, end=' ')
    merged = merged.next
```

```
1 1 2 3 4 4 5 6
```

In [6]:

```
#3. Remove Duplicates from Sorted Array
def removeDuplicates(nums):
    if not nums:
        return 0

    k = 1
    for i in range(1, len(nums)):
        if nums[i] != nums[i-1]:
            nums[k] = nums[i]
            k += 1
    return k

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

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

In [7]:

```
#4. Search in Rotated Sorted Array
def search(nums, target):
    left, right = 0, len(nums) - 1
    while left <= right:
        mid = (left + right) // 2
        if nums[mid] == target:
            return mid
        if nums[left] <= nums[mid]:
            if nums[left] <= target < nums[mid]:
                right = mid - 1
            else:
                left = mid + 1
        else:
            if nums[mid] < target <= nums[right]:
                left = mid + 1
            else:
                right = mid - 1

    return -1

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

```
4
```

In [8]:

```
#5. Find First and Last Position of Element in Sorted Array
def searchRange(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, right = findLeft(nums, target), findRight(nums, target)
    if left <= right:
        return [left, right]
    return [-1, -1]

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

```
[3, 4]
```

In [9]:

```
#6. Sort Colors
def sortColors(nums):
    low, mid, high = 0, 0, len(nums) - 1

    while mid <= high:
        if nums[mid] == 0:
            nums[low], nums[mid] = nums[mid], nums[low]
            low += 1
            mid += 1
        elif nums[mid] == 1:
            mid += 1
        else:
            nums[high], nums[mid] = nums[mid], nums[high]
            high -= 1

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

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

In [10]:

```
#7. Remove Duplicates from Sorted List
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, ListNode(3, ListNode(3)))))
result = deleteDuplicates(head)
output = []
while result:
    output.append(result.val)
    result = result.next
print(output)
```

```
[1, 2, 3]
```

In [11]:

```
#8. Merge Sorted Array
def merge(nums1, m, nums2, n):
    i, j, k = m - 1, n - 1, 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
merge(nums1, m, nums2, n)
print(nums1)
```

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

In [12]:

```
#9. Convert Sorted Array to Binary Search Tree
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]
root = sortedArrayToBST(nums)
def inorder_traversal(root):
    return inorder_traversal(root.left) + [root.val] + inorder_traversa
print(inorder_traversal(root))
```

```
[-10, -3, 0, 5, 9]
```

In [13]:

```
#10. Insertion Sort List
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)
output = []
while result:
    output.append(result.val)
    result = result.next
print(output)
```

```
[1, 2, 3, 4]
```

In [ ]:

```
pp
```