DAY 5 LAB PROGRAMS

1. YOU ARE GIVEN AN ARRAY OF K LINKED-LISTS LISTS, EACH LINKED-LIST IS SORTED IN ASCENDING ORDER. MERGE ALL THE LINKED-LISTS INTO ONE SORTED LINKED-LIST AND RETURN IT.

Sol:-
```
from queue import PriorityQueue


class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
def mergeKLists(lists):
    dummy = ListNode(0)
    curr = dummy
    q = PriorityQueue()
for l in lists:
        if l:
            q.put((l.val, l))
while not q.empty():
        val, node = q.get()
        curr.next = ListNode(val)
        curr = curr.next
        node = node.next
        if node:
            q.put((node.val, node))
return dummy.next
```

2. GIVEN AN INTEGER ARRAY NUMS SORTED IN NON-DECREASING ORDER, REMOVE THE DUPLICATES INPLACE SUCH THAT EACH UNIQUE ELEMENT APPEARS ONLY ONCE. THE RELATIVE ORDER OF THE ELEMENTS SHOULD BE KEPT THE SAME.

Sol:-
```
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
```

4. SEARCH IN ROTATED SORTED ARRAY

Sol:-

```
def search(nums, target):
    left, right = 0, len(nums) - 1
    while left <= right:
        mid = left + (right - left) // 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))
```

5. FIND FIRST AND LAST POSITION OF ELEMENT IN SORTED ARRAY

Sol:-

```
class Solution:
    def searchRange(self, nums, target):
        def binarySearchLeft(nums, target):
```

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

6. Sort Colors Given an array nums with n objects colored red, white, or blue, sort them in-place so that objects of the same color are adjacent, with the colors in the order red, white, and blue. We will use the integers 0, 1, and 2 to represent the color red, white, and blue, respectively. You must solve this problem without using the library's sort function.

Sol:-

```
def sortColors(nums):
   red, white, blue = 0, 0, len(nums) - 1
while white <= blue:
    if nums[white] == 0:
        nums[red], nums[white] = nums[white], nums[red]
        red += 1
        white += 1
    elif nums[white] == 1:
        white += 1
    else:
        nums[white], nums[blue] = nums[blue], nums[white]
        blue -= 1
        left, right = 0, len(nums)
        while left < right:
            mid = left + (right - left) // 2
            if nums[mid] <= target:
                left = mid + 1
```

```
            else:
                right = mid
        return left
    left_idx = binarySearchLeft(nums, target)
    right_idx = binarySearchRight(nums, target
if left_idx <= right_idx:
        return [left_idx, right_idx]
    else:
        return [-1, -1]
nums = [5, 7, 7, 8, 8, 10]
target = 8
solution = Solution()
print(solution.searchRange(nums, target))
```

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
```

8. MERGE SORTED ARRAY

```
Sol:- def merge_sorted_arrays(nums1, m, nums2, n):
  nums1[m:] = nums2
  nums1.sort()
  return nums1
```

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

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
```

10. INSERTION SORT LIST GIVEN THE HEAD OF A SINGLY LINKED LIST, SORT THE LIST USING INSERTION SORT, AND RETURN THE SORTED LIST'S HEAD.

Sol:-

```
def improvedInsertionSortList(head):
    if not head or not head.next:
        return head
    dummy = ListNode(0)
    dummy.next = head
    last_sorted = head
    current = head.next
    while current:
        if last_sorted.val <= current.val:
```

```
            last_sorted = last_sorted.next
        else:
            prev = dummy
            while prev.next.val <= current.val:
                prev = prev.next
            last_sorted.next = current.next
            current.next = prev.next
            prev.next = current
        current = last_sorted.next
    return dummy.next
```