### DSA Assignment

### Problem 1: Reverse a singly linked list

class ListNode:

def \_\_init\_\_(self, val=0, next=None):

self.val = val

self.next = next

def reverseList(head):

prev = None

current = head

while current:

next\_node = current.next

current.next = prev

prev = current

current = next\_node

return prev

# Helper function to print the list

def printList(head):

while head:

print(head.val, end=" -> ")

head = head.next

print("None")

# Example usage

head = ListNode(1, ListNode(2, ListNode(3, ListNode(4, ListNode(5)))))

reversed\_head = reverseList(head)

printList(reversed\_head)

### Problem 2: Merge two sorted linked lists into one sorted linked list

def mergeTwoLists(l1, l2):

dummy = ListNode()

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

# Example usage

l1 = ListNode(1, ListNode(3, ListNode(5)))

l2 = ListNode(2, ListNode(4, ListNode(6)))

merged\_head = mergeTwoLists(l1, l2)

printList(merged\_head)

### Problem 3: Remove the nth node from the end of a linked list

def removeNthFromEnd(head, n):

dummy = ListNode(0, head)

first = dummy

second = dummy

for \_ in range(n + 1):

first = first.next

while first:

first = first.next

second = second.next

second.next = second.next.next

return dummy.next

# Example usage

head = ListNode(1, ListNode(2, ListNode(3, ListNode(4, ListNode(5)))))

new\_head = removeNthFromEnd(head, 2)

printList(new\_head)

### Problem 4: Find the intersection point of two linked lists

def getIntersectionNode(headA, headB):

if not headA or not headB:

return None

a, b = headA, headB

while a != b:

a = a.next if a else headB

b = b.next if b else headA

return a

# Example usage

intersect = ListNode(3, ListNode(4))

headA = ListNode(1, ListNode(2, intersect))

headB = ListNode(9, ListNode(8, intersect))

intersection\_node = getIntersectionNode(headA, headB)

print(intersection\_node.val if intersection\_node else "No intersection")

### Problem 5: Remove duplicates from a sorted linked list

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

# Example usage

head = ListNode(1, ListNode(1, ListNode(2, ListNode(3, ListNode(3)))))

new\_head = deleteDuplicates(head)

printList(new\_head)

### Problem 6: Add two numbers represented by linked lists

def addTwoNumbers(l1, l2):

dummy = ListNode()

current = dummy

carry = 0

while l1 or l2 or carry:

val1 = l1.val if l1 else 0

val2 = l2.val if l2 else 0

carry, out = divmod(val1 + val2 + carry, 10)

current.next = ListNode(out)

current = current.next

l1 = l1.next if l1 else None

l2 = l2.next if l2 else None

return dummy.next

# Example usage

l1 = ListNode(2, ListNode(4, ListNode(3)))

l2 = ListNode(5, ListNode(6, ListNode(4)))

result = addTwoNumbers(l1, l2)

printList(result)

### Problem 7: Swap nodes in pairs in a linked list

def swapPairs(head):

dummy = ListNode(0, head)

current = dummy

while current.next and current.next.next:

first = current.next

second = current.next.next

first.next = second.next

current.next = second

current.next.next = first

current = current.next.next

return dummy.next

# Example usage

head = ListNode(1, ListNode(2, ListNode(3, ListNode(4))))

new\_head = swapPairs(head)

printList(new\_head)

### Problem 8: Reverse nodes in a linked list in groups of k

def reverseKGroup(head, k):

dummy = ListNode(0, head)

group\_prev = dummy

while True:

kth = group\_prev

for \_ in range(k):

kth = kth.next

if not kth:

return dummy.next

group\_next = kth.next

prev, curr = kth.next, group\_prev.next

for \_ in range(k):

curr.next, prev, curr = prev, curr, curr.next

temp = group\_prev.next

group\_prev.next, group\_prev = kth, temp

return dummy.next

# Example usage

head = ListNode(1, ListNode(2, ListNode(3, ListNode(4, ListNode(5)))))

new\_head = reverseKGroup(head, 3)

printList(new\_head)

### Problem 9: Determine if a linked list is a palindrome

def isPalindrome(head):

fast = slow = head

while fast and fast.next:

fast = fast.next.next

slow = slow.next

prev = None

while slow:

next\_node = slow.next

slow.next = prev

prev = slow

slow = next\_node

left, right = head, prev

while right:

if left.val != right.val:

return False

left = left.next

right = right.next

return True

# Example usage

head = ListNode(1, ListNode(2, ListNode(2, ListNode(1))))

print(isPalindrome(head))

### Deployment Instructions

1. **Choose a Cloud Platform**: You can use platforms like Heroku, AWS, Google Cloud, or Azure. Here, I'll use Heroku as an example.
2. **Install Heroku CLI**: Follow the instructions on the [Heroku website](https://devcenter.heroku.com/articles/heroku-cli) to install the Heroku CLI.
3. **Create a `requirements.txt` file**: This file should list all the dependencies of your project. You can generate it using:
4. **Create a `Procfile`**: This file should specify the command to run your app. For example:
5. **Initialize a Git Repository**:
6. **Deploy to Heroku**:

### Problem 10: Rotate a linked list to the right by k places

class ListNode:

def \_\_init\_\_(self, val=0, next=None):

self.val = val

self.next = next

def rotateRight(head, k):

if not head or not head.next or k == 0:

return head

# Find the length of the list

length = 1

tail = head

while tail.next:

tail = tail.next

length += 1

# Connect the tail to the head to make it a circular list

tail.next = head

# Find the new tail: (length - k % length - 1)th node

# and the new head: (length - k % length)th node

k = k % length

new\_tail = head

for \_ in range(length - k - 1):

new\_tail = new\_tail.next

new\_head = new\_tail.next

# Break the circle

new\_tail.next = None

return new\_head

### Problem 11: Flatten a multilevel doubly linked list

class Node:

def \_\_init\_\_(self, val, prev=None, next=None, child=None):

self.val = val

self.prev = prev

self.next = next

self.child = child

def flatten(head):

if not head:

return head

pseudoHead = Node(0, None, head, None)

prev = pseudoHead

stack = []

stack.append(head)

while stack:

curr = stack.pop()

prev.next = curr

curr.prev = prev

if curr.next:

stack.append(curr.next)

if curr.child:

stack.append(curr.child)

curr.child = None

prev = curr

pseudoHead.next.prev = None

return pseudoHead.next

### Problem 12: Rearrange a linked list such that all even positioned nodes are placed at the end

class ListNode:

def \_\_init\_\_(self, val=0, next=None):

self.val = val

self.next = next

def rearrangeEvenOdd(head):

if not head:

return None

odd = head

even = head.next

evenHead = even

while even and even.next:

odd.next = even.next

odd = odd.next

even.next = odd.next

even = even.next

odd.next = evenHead

return head

### Problem 13: Given a non-negative number represented as a linked list, add one to it

class ListNode:

def \_\_init\_\_(self, val=0, next=None):

self.val = val

self.next = next

def addOne(head):

def reverse(node):

prev = None

while node:

next\_node = node.next

node.next = prev

prev = node

node = next\_node

return prev

head = reverse(head)

curr = head

carry = 1

while curr:

curr.val += carry

if curr.val < 10:

carry = 0

break

else:

curr.val = 0

carry = 1

if not curr.next and carry:

curr.next = ListNode(1)

carry = 0

curr = curr.next

return reverse(head)

### Problem 14: Given a sorted array and a target value, return the index if the target is found. If not, return the index where it would be inserted

def searchInsert(nums, target):

left, right = 0, len(nums) - 1

while left <= right:

mid = (left + right) // 2

if nums[mid] == target:

return mid

elif nums[mid] < target:

left = mid + 1

else:

right = mid - 1

return left

### Problem 15: Find the minimum element in a rotated sorted array

def findMin(nums):

left, right = 0, len(nums) - 1

while left < right:

mid = (left + right) // 2

if nums[mid] > nums[right]:

left = mid + 1

else:

right = mid

return nums[left]

### Problem 16: Search for a target value in a 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

### Problem 17: Find the peak element in an array

def findPeakElement(nums):

left, right = 0, len(nums) - 1

while left < right:

mid = (left + right) // 2

if nums[mid] > nums[mid + 1]:

right = mid

else:

left = mid + 1

return left

### Problem 18: Count the number of negative numbers in a sorted matrix

def countNegatives(grid):

count = 0

for row in grid:

for num in row:

if num < 0:

count += 1

return count

### Problem 19: Determine if a target value is present in a 2D matrix

def searchMatrix(matrix, target):

if not matrix or not matrix[0]:

return False

rows, cols = len(matrix), len(matrix[0])

left, right = 0, rows \* cols - 1

while left <= right:

mid = (left + right) // 2

mid\_value = matrix[mid // cols][mid % cols]

if mid\_value == target:

return True

elif mid\_value < target:

left = mid + 1

else:

right = mid - 1

return False

### Problem 20: Find the median of two sorted arrays

def findMedianSortedArrays(nums1, nums2):

nums = sorted(nums1 + nums2)

n = len(nums)

if n % 2 == 1:

return float(nums[n // 2])

else:

return (nums[n // 2 - 1] + nums[n // 2]) / 2.0

### Problem 21: Find the smallest letter greater than the target

def nextGreatestLetter(letters, target):

for letter in letters:

if letter > target:

return letter

return letters[0]

### Problem 22: Sort colors (red, white, and blue)

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[mid], nums[high] = nums[high], nums[mid]

high -= 1

### Problem 23: Find the kth largest element in an unsorted array

import heapq

def findKthLargest(nums, k):

return heapq.nlargest(k, nums)[-1]

### Problem 24: Reorder array in-place

def wiggleSort(nums):

nums.sort()

for i in range(1, len(nums) - 1, 2):

nums[i], nums[i + 1] = nums[i + 1], nums[i]

### Problem 25: Calculate the sum of all elements in an array

def arraySum(nums):

return sum(nums)

### Problem 26: Find the maximum element in an array of integers

def find\_max(arr):

return max(arr)

# Example usage

arr = [3, 7, 2, 9, 4, 1]

print(find\_max(arr)) # Output: 9

### Problem 27: Implement linear search to find the index of a target element in an array

def linear\_search(arr, target):

for i in range(len(arr)):

if arr[i] == target:

return i

return -1

# Example usage

arr = [5, 3, 8, 2, 7, 4]

target = 8

print(linear\_search(arr, target)) # Output: 2

### Problem 28: Calculate the factorial of a given number

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n-1)

# Example usage

n = 5

print(factorial(n)) # Output: 120

### Problem 29: Check if a given number is a prime number

def is\_prime(n):

if n <= 1:

return False

for i in range(2, int(n\*\*0.5) + 1):

if n % i == 0:

return False

return True

# Example usage

n = 7

print(is\_prime(n)) # Output: True

### Problem 30: Generate the Fibonacci series up to a given number n

def fibonacci(n):

fib\_series = [0, 1]

while fib\_series[-1] + fib\_series[-2] <= n:

fib\_series.append(fib\_series[-1] + fib\_series[-2])

return fib\_series

# Example usage

n = 8

print(fibonacci(n)) # Output: [0, 1, 1, 2, 3, 5, 8]

### Problem 31: Calculate the power of a number using recursion

def power(base, exponent):

if exponent == 0:

return 1

else:

return base \* power(base, exponent - 1)

# Example usage

base = 3

exponent = 4

print(power(base, exponent)) # Output: 81

### Problem 32: Reverse a given string

def reverse\_string(s):

return s[::-1]

# Example usage

s = "hello"

print(reverse\_string(s)) # Output: "olleh"