ASSIGNMENT-XIII

**Question 1**

Given two linked list of the same size, the task is to create a new linked list using those linked lists. The condition is that the greater node among both linked list will be added to the new linked list.

**Examples:**

Input: list1 = 5->2->3->8

list2 = 1->7->4->5

Output: New list = 5->7->4->8

Input:list1 = 2->8->9->3

list2 = 5->3->6->4

Output: New list = 5->8->9->4

**Ans:** class ListNode:

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

self.val = val

self.next = next

def mergeLists(list1, list2):

newList = None

current = None

ptr1 = list1

ptr2 = list2

while ptr1 and ptr2:

if ptr1.val >= ptr2.val:

newNode = ListNode(ptr1.val)

ptr1 = ptr1.next

else:

newNode = ListNode(ptr2.val)

ptr2 = ptr2.next

if newList is None:

newList = newNode

current = newNode

else:

current.next = newNode

current = current.next

while ptr1:

newNode = ListNode(ptr1.val)

current.next = newNode

current = current.next

ptr1 = ptr1.next

while ptr2:

newNode = ListNode(ptr2.val)

current.next = newNode

current = current.next

ptr2 = ptr2.next

return newList

**Question 2**

Write a function that takes a list sorted in non-decreasing order and deletes any duplicate nodes from the list. The list should only be traversed once.

For example if the linked list is 11->11->11->21->43->43->60 then removeDuplicates() should convert the list to 11->21->43->60.

**Example 1:**

Input:

LinkedList:

11->11->11->21->43->43->60

Output:

11->21->43->60

**Example 2:**

Input:

LinkedList:

10->12->12->25->25->25->34

Output:

10->12->25->34

**Ans**: class ListNode:

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

self.val = val

self.next = next

def removeDuplicates(head):

if not head:

return head

current = head

while current.next:

if current.val == current.next.val:

current.next = current.next.next

else:

current = current.next

return head

**Question 3**

Given a linked list of size **N**. The task is to reverse every **k** nodes (where k is an input to the function) in the linked list. If the number of nodes is not a multiple of *k* then left-out nodes, in the end, should be considered as a group and must be reversed (See Example 2 for clarification).

**Example 1:**

Input:

LinkedList: 1->2->2->4->5->6->7->8

K = 4

Output:4 2 2 1 8 7 6 5

Explanation:

The first 4 elements 1,2,2,4 are reversed first

and then the next 4 elements 5,6,7,8. Hence, the

resultant linked list is 4->2->2->1->8->7->6->5.

**Example 2:**

Input:

LinkedList: 1->2->3->4->5

K = 3

Output:3 2 1 5 4

Explanation:

The first 3 elements are 1,2,3 are reversed

first and then elements 4,5 are reversed.Hence,

the resultant linked list is 3->2->1->5->4.

**Ans:** class ListNode:

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

self.val = val

self.next = next

def reverseKNodes(head, k):

if not head or k == 1:

return head

dummy = ListNode(0)

dummy.next = head

prev = dummy

curr = head

count = 0

while curr:

count += 1

if count % k == 0:

prev = reverseGroup(prev, curr.next)

curr = prev.next

else:

curr = curr.next

return dummy.next

def reverseGroup(prev, next\_node):

last = prev.next

curr = last.next

while curr != next\_node:

last.next = curr.next

curr.next = prev.next

prev.next = curr

curr = last.next

return last

**Question 4**

Given a linked list, write a function to reverse every alternate k nodes (where k is an input to the function) in an efficient way. Give the complexity of your algorithm.

**Example:**

Inputs: 1->2->3->4->5->6->7->8->9->NULL and k = 3

Output: 3->2->1->4->5->6->9->8->7->NULL.

**Ans**: class ListNode:

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

self.val = val

self.next = next

def reverseAlternateKNodes(head, k):

if not head or k <= 1:

return head

prev = None

curr = head

next\_node = None

count = 0

reverse = True

while curr:

count += 1

next\_node = curr.next

if count % k == 0 and reverse:

if prev:

prev.next = None

head, curr = reverseGroup(head, curr)

prev = curr

else:

prev = curr

curr = next\_node

reverse = not reverse

return head

def reverseGroup(head, tail):

prev = None

curr = head

next\_node = None

while curr != tail.next:

next\_node = curr.next

curr.next = prev

prev = curr

curr = next\_node

head.next = curr

return prev, head

**Question 5**

Given a linked list and a key to be deleted. Delete last occurrence of key from linked. The list may have duplicates.

**Examples**:

Input: 1->2->3->5->2->10, key = 2

Output: 1->2->3->5->10

**Ans:** class ListNode:

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

self.val = val

self.next = next

def deleteLastOccurrence(head, key):

if not head:

return None

prev = None

lastOccur = None

current = head

while current:

if current.val == key:

lastOccur = current

prev = current

current = current.next

if not lastOccur:

return head

if lastOccur == head:

head = head.next

else:

prev.next = lastOccur.next

return head

**Question 6**

Given two sorted linked lists consisting of **N** and **M** nodes respectively. The task is to merge both of the lists (in place) and return the head of the merged list.

**Examples:**

Input: a: 5->10->15, b: 2->3->20

Output: 2->3->5->10->15->20

Input: a: 1->1, b: 2->4

Output: 1->1->2->4

**Ans**: class ListNode:

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

self.val = val

self.next = next

def mergeTwoLists(a, b):

dummy = ListNode(0)

curr = dummy

while a and b:

if a.val <= b.val:

curr.next = a

a = a.next

else:

curr.next = b

b = b.next

curr = curr.next

if a:

curr.next = a

if b:

curr.next = b

return dummy.next

**Question 7**

Given a **Doubly Linked List**, the task is to reverse the given Doubly Linked List.

**Example:**

Original Linked list 10 8 4 2

Reversed Linked list 2 4 8 10

**Ans:** class ListNode:

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

self.val = val

self.prev = prev

self.next = next

def reverseDLL(head):

if not head:

return None

prev = None

current = head

while current:

next\_node = current.next

current.next = prev

current.prev = next\_node

prev = current

current = next\_node

head = prev

return head

**Question 8**

Given a doubly linked list and a position. The task is to delete a node from given position in a doubly linked list.

**Example 1:**

Input:

LinkedList = 1 <--> 3 <--> 4

x = 3

Output:1 3

Explanation:After deleting the node at

position 3 (position starts from 1),

the linked list will be now as 1->3.

**Example 2:**

Input:

LinkedList = 1 <--> 5 <--> 2 <--> 9

x = 1

Output:5 2 9

**Ans:** class ListNode:

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

self.val = val

self.prev = prev

self.next = next

def deleteNode(head, position):

if not head:

return None

if position == 1:

new\_head = head.next

if new\_head:

new\_head.prev = None

return new\_head

current = head

count = 1

while current and count < position:

current = current.next

count += 1

if not current:

return head

prev\_node = current.prev

next\_node = current.next

if prev\_node:

prev\_node.next = next\_node

if next\_node:

next\_node.prev = prev\_node

return head