**Assignment Questions 13**

**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

Solve:-

/\*\*

\* Definition for singly-linked list.

\* public class ListNode {

\* intval;

\* ListNode next;

\* ListNode() {}

\* ListNode(int val) { this.val = val; }

\* ListNode(int val, ListNode next) { this.val = val; this.next = next; }

\* }

\*/

class Solution {

public ListNodeaddTwoNumbers(ListNode l1, ListNode l2) {

int carry =0;

ListNodenewHead = new ListNode(0);

ListNode p1 = l1, p2 = l2, p3=newHead;

while(p1 != null || p2 != null)

{

if(p1 != null)

{

carry += p1.val;

p1 = p1.next;

}

if(p2 != null)

{

carry += p2.val;

p2 = p2.next;

}

p3.next = new ListNode(carry%10);

p3 = p3.next;

carry /= 10;

}

if(carry==1)

p3.next=new ListNode(1);

return newHead.next;

}

}

**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

Solve:-

class Solution {

public ListNodedeleteDuplicates(ListNode head) {

ListNodecurr = head;

while(head != null &&head.next != null) {

if(head.val == head.next.val) head.next = head.next.next;

else head = head.next;

}

return curr;

}

}

**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.aside>

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.

Solve:-

class Solution {

public ListNodereverseKGroup(ListNode head, int k) {

ListNode dummy = new ListNode(0);

dummy.next = head;

ListNodeprevGroupTail = dummy;

while (head != null) {

ListNodegroupStart = head;

ListNodegroupEnd = getGroupEnd(head, k);

if (groupEnd == null)

break;

prevGroupTail.next = reverseList(groupStart, groupEnd.next);

prevGroupTail = groupStart;

head = prevGroupTail.next;

}

ListNodenewHead = dummy.next;

return newHead;

}

private ListNodegetGroupEnd(ListNode head, int k) {

while (head != null && --k > 0)

head = head.next;

return head;

}

private ListNodereverseList(ListNode head, ListNode stop) {

ListNodeprev = stop;

while (head != stop) {

ListNode next = head.next;

head.next = prev;

prev = head;

head = next;

}

return prev;

}

}

**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.

Solve:-

class LinkedList {

static Node head;

class Node {

int data;

Node next;

Node(int d) {

data = d;

next = null;

}

}

Node kAltReverse(Node node, int k) {

Node current = node;

Node next = null, prev = null;

int count = 0;

/\*1) reverse first k nodes of the linked list \*/

while (current != null && count < k) {

next = current.next;

current.next = prev;

prev = current;

current = next;

count++;

}

if (node != null) {

node.next = current;

}

count = 0;

while (count < k - 1 &&current != null) {

current = current.next;

count++;

}

if (current != null) {

current.next = kAltReverse(current.next, k);

}

return prev;

}

void printList(Node node) {

while (node != null) {

System.out.print(node.data + " ");

node = node.next;

}

}

void push(int newdata) {

Node mynode = new Node(newdata);

mynode.next = head;

head = mynode;

}

public static void main(String[] args) {

LinkedList list = new LinkedList();

// Creating the linkedlist

for (int i = 20; i> 0; i--) {

list.push(i);

}

System.out.println("Given Linked List :");

list.printList(head);

head = list.kAltReverse(head, 3);

System.out.println("");

System.out.println("Modified Linked List :");

list.printList(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

Solve:-

class GFG

{

static class Node

{

int key;

Node next;

};

static Node deleteLast(Node head, int key)

{

Node x = null;

Node temp = head;

while (temp != null)

{

if (temp.key == key)

x = temp;

temp = temp.next;

}

if (x != null)

{

x.key = x.next.key;

// Store and unlink next

temp = x.next;

x.next = x.next.next;

// Free memory for next

}

return head;

}

static Node newNode(int key)

{

Node temp = new Node();

temp.key = key;

temp.next = null;

return temp;

}

static void printList( Node node)

{

while (node != null)

{

System.out.printf(" %d ", node.key);

node = node.next;

}

}

// Driver code/

public static void main(String args[])

{

// /Start with the empty list /

Node head = newNode(1);

head.next = newNode(2);

head.next.next = newNode(3);

head.next.next.next = newNode(5);

head.next.next.next.next = newNode(2);

head.next.next.next.next.next = newNode(10);

System.out.printf("Created Linked List: ");

printList(head);

deleteLast(head, 2);

System.out.printf("\nLinked List after Deletion of 2: ");

printList(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.

**Example**

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

Solve:-

/\*\*

\* Definition for singly-linked list.

\* public class ListNode {

\* intval;

\* ListNode next;

\* ListNode() {}

\* ListNode(int val) { this.val = val; }

\* ListNode(int val, ListNode next) { this.val = val; this.next = next; }

\* }

\*/

class Solution {

public ListNodemergeTwoLists(ListNode list1, ListNode list2) {

if(list1!=null && list2!=null){

if(list1.val<list2.val){

list1.next=mergeTwoLists(list1.next,list2);

return list1;

}

else{

list2.next=mergeTwoLists(list1,list2.next);

return list2;

}

}

if(list1==null)

return list2;

return list1;

}

}

**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

Solve:-

class Solution {

public ListNodereverseList(ListNode head) {

ListNodeprev = null;

ListNode current = head;

while(current != null) {

ListNode next = current.next;

current.next = prev;

prev = current;

current = next;

}

return prev;

}

}