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 c
ondition 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

code:-

```
class GFG
{
static class Node
{
 int data;
 Node next;
};
static Node insert(Node root, int item)
{
 Node ptr, temp;
 temp = new Node();
 temp.data = item;
 temp.next = null;

 if (root == null)
  root = temp;
 else {
  ptr = root;
  while (ptr.next != null)
   ptr = ptr.next;

  ptr.next = temp;
 }
 return root;
}
static Node newList(Node root1, Node root2)
{
 Node ptr1 = root1, ptr2 = root2, ptr;
 Node root = null, temp;

 while (ptr1 != null) {
  temp = new Node();
  temp.next = null;
  if (ptr1.data < ptr2.data)
   temp.data = ptr2.data;
  else
   temp.data = ptr1.data;

  if (root == null)
   root = temp;
```

```
  else {
   ptr = root;
   while (ptr.next != null)
    ptr = ptr.next;

   ptr.next = temp;
  }
  ptr1 = ptr1.next;
  ptr2 = ptr2.next;
 }
 return root;
}

static void display(Node root)
{
 while (root != null)
 {
  System.out.print( root.data + "->");
  root = root.next;
 }
 System.out.println();
}
public static void main(String args[])
{
 Node root1 = null, root2 = null, root = null;
 root1=insert(root1, 5);
 root1=insert(root1, 2);
 root1=insert(root1, 3);
 root1=insert(root1, 8);
 System.out.print("First List: ");
 display(root1);
 root2=insert(root2, 1);
 root2=insert(root2, 7);
 root2=insert(root2, 4);
 root2=insert(root2, 5);
 System.out.print( "Second List: ");
 display(root2);
 root = newList(root1, root2);
 System.out.print("New List: ");
 display(root);
}
}
```

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

code:-

```
class LinkedList {
 Node head; // head of list
 class Node {
  int data;
  Node next;
  Node(int d)
  {
   data = d;
   next = null;
  }
 }

 void removeDuplicates()
 {
  Node curr = head;
  while (curr != null) {
   Node temp = curr;
   while (temp != null && temp.data == curr.data) {
    temp = temp.next;
   }
   curr.next = temp;
   curr = curr.next;
  }
 }
 public void push(int new_data)
 {
  Node new_node = new Node(new_data);
  new_node.next = head;
  head = new_node;
 }
 void printList()
 {
  Node temp = head;
  while (temp != null) {
   System.out.print(temp.data + " ");
   temp = temp.next;
  }
  System.out.println();
 }
 public static void main(String args[])
 {
  LinkedList llist = new LinkedList();
  llist.push(20);
  llist.push(13);
  llist.push(13);
  llist.push(11);
  llist.push(11);
```

```
  llist.push(11);
  System.out.println("List before removal of duplicates");
  llist.printList();
  llist.removeDuplicates();
  System.out.println("List after removal of elements");
  llist.printList();
 }
}
```

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 th e linked list. If the number of nodes is not a multiple of k then left-out nodes, in the end, should be conside red 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.
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.

code:-

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
```

```
    int length (ListNode* head)
    {
        int len = 0;
        while(head != NULL)
        {
            len++;
            head = head -> next;
        }
        return len;
    }

    ListNode* reverseKGroup(ListNode* head, int k) {

        //head = 1

        int len = length(head); //Calculate length of LL
        if(len < k) //As mentioned in aue, if len < k don't reverse
        {
            return head;
        }
        int cnt = 0;
        ListNode* curr = head; //1 --- After 1st step, curr = 2
        ListNode* prev = NULL; //NULL
        ListNode* forward = NULL;

        while(curr != NULL && cnt < k)  //Reverseing 'k' nodes initially
        {
            forward = curr -> next; //2 --- 3
            curr -> next = prev; //1 -> 2 is broken and NULL <- 1 --- 2 -> 1
            prev = curr; //prev = 1 --- prev = 2
            curr = forward; // curr = 2 --- curr = 3
            cnt++;
        }
        if(forward != NULL)
        {
            head -> next = reverseKGroup(forward, k); //Recursively calling for remaining nodes
        }
        //I've stored it in head -> next bcz, head = 1 and I've coneected it with 4, head of the new LL

        return prev; // return prev bcz, 2 is the head of our final LL and it is stored in 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.

code:-

```
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;
 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();
 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 d
uplicates.
Examples:
Input: 1->2->3->5->2->10, key = 2
Output: 1->2->3->5->10

code:-

```
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;
  temp = x.next;
  x.next = x.next.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;
 }
}
public static void main(String args[])
{
 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 lis ts (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

code:-

```
class Node {
 int key;
 Node next;

 public Node(int key) {
  this.key = key;
  next = null;
 }
}

public class Main {
 public static Node newNode(int key) {
  return new Node(key);
 }
 public static void main(String[] args) {
  Node a = new Node(5);
  a.next = new Node(10);
  a.next.next = new Node(15);
  a.next.next.next = new Node(40);

  Node b = new Node(2);
  b.next = new Node(3);
  b.next.next = new Node(20);
```

```
  List<Integer> v = new ArrayList<>();
  while (a != null) {
   v.add(a.key);
   a = a.next;
  }

  while (b != null) {
   v.add(b.key);
   b = b.next;
  }

  Collections.sort(v);
  Node result = new Node(-1);
  Node temp = result;
  for (int i = 0; i < v.size(); i++) {
   result.next = new Node(v.get(i));
   result = result.next;
  }

  temp = temp.next;
  System.out.print("Resultant Merge Linked List is : ");
  while (temp != null) {
   System.out.print(temp.key + " ");
   temp = temp.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

code:-

```
class LinkedList {
 static Node head;
 static class Node {
  int data;
  Node next, prev;

  Node(int d)
  {
   data = d;
   next = prev = null;
  }
 }
 void reverse()
 {
  Node temp = null;
  Node current = head;
  while (current != null) {
   temp = current.prev;
```

```
   current.prev = current.next;
   current.next = temp;
   current = current.prev;
  }
  if (temp != null) {
   head = temp.prev;
  }
 }
 void push(int new_data)
 {
  Node new_node = new Node(new_data);
  new_node.prev = null;
  new_node.next = head;
  if (head != null) {
   head.prev = new_node;
  }
  head = new_node;
 }
 void printList(Node node)
 {
  while (node != null) {
   System.out.print(node.data + " ");
   node = node.next;
  }
 }
 public static void main(String[] args)
 {
  LinkedList list = new LinkedList();
  list.push(2);
  list.push(4);
  list.push(8);
  list.push(10);
  System.out.println("Original linked list ");
  list.printList(head);
  list.reverse();
  System.out.println("");
  System.out.println("The reversed Linked List is ");
  list.printList(head);
 }
}
```

Question 8.
Given a doubly linked list and a position. The task is to delete a node from given position in a doubly linke
d 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

code:-

```
class Node
{
 int data;
 Node next, prev;
}

class GFG
{
 static Node deleteNode(Node del)
 {
  if (head == null || del == null)
   return null;
  if (head == del)
   head = del.next;
  if (del.next != null)
   del.next.prev = del.prev;
  if (del.prev != null)
   del.prev.next = del.next;

  del = null;

  return head;
 }
 static void deleteNodeAtGivenPos(int n)
 {
  if (head == null || n <= 0)
   return;

  Node current = head;
  int i;
  for (i = 1; current != null && i < n; i++)
  {
   current = current.next;
  }
  if (current == null)
   return;
  deleteNode(current);
 }
 static void push(int new_data)
 {
  Node new_node = new Node();
  new_node.data = new_data;
  new_node.prev = null;
  new_node.next = head;
  if (head != null)
   head.prev = new_node;
  head = new_node;
 }
 static void printList()
```

```
 {
  Node temp = head;
  if (temp == null)
   System.out.print("Doubly Linked list empty");

  while (temp != null)
  {
   System.out.print(temp.data + " ");
   temp = temp.next;
  }
  System.out.println();
 }
 public static void main(String[] args)
 {
  push(5);
  push(2);
  push(4);
  push(8);
  push(10);
  System.out.println("Doubly linked "+"list before deletion:");
                printList();
  int n = 2;
  deleteNodeAtGivenPos(n);
  System.out.println("Doubly linked "+"list after deletion:");
  printList();
 }
}
```