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

Solution:-

```
import java.util.*;

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 void display(Node root)
{
    while (root != null)
    {
        System.out.print( root.data + " - > ");
        root = root.next;
    }
    System.out.print("null");
}
 static Node newList(Node root1, Node root2)
{
    Node ptr1 = root1, ptr2 = root2;
```

```
    Node root = null;
    while (ptr1 != null)
    {
        int currMax = ((ptr1.data < ptr2.data)
                    ? ptr2.data
                    : ptr1.data);
        if (root == null)
        {
            Node temp = new Node();
            temp.data = currMax;
            temp.next = null;
            root = temp;
        }
         else
        {
            root = insert(root, currMax);
        }
         ptr1 = ptr1.next;
        ptr2 = ptr2.next;
    }
    return root;
}

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);

    // Second linked list
    root2 = insert(root2, 1);
    root2 = insert(root2, 7);
    root2 = insert(root2, 4);
    root2 = insert(root2, 5);
     root = newList(root1, root2);
     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.

Input:

LinkedList:

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

Output:

11->21->43->60

Solution:-

```
import java.io.*;
class LinkedList {
	Node head; // head of list
	class Node {
		int data;
		Node next;
		Node(int d)
		{
			data = d;
			next = null;
		}
	}

	void removeDuplicates()
	{
		/*Another reference to head*/
		Node curr = head;

		/* Traverse list till the last node */
		while (curr != null) {
			Node temp = curr;
			/*Compare current node with the next node and
			keep on deleting them until it matches the
```

```
                current node data */
                while (temp != null && temp.data == curr.data) {
                        temp = temp.next;
                }
                /*Set current node next to the next different
                element denoted by temp*/
                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 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.

Solution:-

```
class Solution
{
    public static Node reverse(Node node, int k)
    {



      int c=0,z=0;
      Node temp=node;
```

```
Node x=node;
Node y=null;
Node head=null;
Node cut=null;
boolean flag=true;

while(true){

   if(c==k||temp==null){
     if(z==0){
     head=y;
     cut=temp;
     }
     else{
       node.next=y;
       node=cut;
       cut=temp;
     }
     z++;
    flag=false;

    c=0;
```

```
        }

        if(temp==null)
        break;

        x=temp.next;
        if(flag)
        temp.next=y;
        else
        temp.next=null;
        y=temp;
        temp=x;
        c++;
        flag=true;

      }

      return head;
    }
}
```

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.

Solution:-

```
// Java program to reverse alternate k nodes in a linked list


class LinkedList {


    static Node head;


    class Node {


        int data;

        Node next;


        Node(int d) {

            data = d;

            next = null;

        }
```

```
}

/* Reverses alternate k nodes and
returns the pointer to the new head node */
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++;
        }

        /* 2) Now head points to the kth node. So change next
        of head to (k+1)th node*/
        if (node != null) {
                node.next = current;
        }
```

```
        /* 3) We do not want to reverse next k nodes. So move the current
        pointer to skip next k nodes */
        count = 0;
        while (count < k - 1 && current != null) {
                current = current.next;
                count++;
        }

        /* 4) Recursively call for the list starting from current->next.
        And make rest of the list as next of first node */
        if (current != null) {
                current.next = kAltReverse(current.next, k);
        }

        /* 5) prev is new head of the input list */
        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.

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

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

Solution:-

```
class Test
{
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;
        }
        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[])
{
        // /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.

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

Solution:-

```
import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

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

Solution:-

```
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);



        prev is always NULL */
        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 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.

Solution:-

```
class Solution
{


   Node deleteNode(Node head,int x)
   {
   // Your code here
   Node temp= head;
   if(x ==1){
      head = head.next;
```

```
        head.prev.next = null;
        head.prev = null;
        return head;
    }

    while(x-2>0){
        temp=temp.next;
        x--;
    }
    Node res = temp.next.next;
    temp.next = res;
    if(res!=null)
    res.prev = temp;
    return head;
    }
}
```