Linked List Assignment Solution

Q1. Given a linked list and a key 'X' in, the task is to check if X is present in the linked list or not.
Solution:

```
public void elementCheck(Node head, int K){
    if(head==null)return;
    Node temp=head;
    while(temp!=null){
        if (temp.data==K) {
           System.out.println("Yes");
           return;
        }
        temp=temp.next;
    }
    System.out.println("No");
}
```

Q2. Insert a node at the given position in a linked list. We are given a pointer to a node, and the new node is inserted after the given node.
Solution:

```
public Node insertNode(int pos, int value) {
    Node newNode = new Node(value);
    if (pos < 1) System.out.println("position should be greater than 0");
    if (pos == 1) {
        newNode.next = head;
        head = newNode;
    }
    else {
        Node temp = head;
        int len = 1;
        while (len < pos && temp != null) {
            temp = temp.next;
            len++;
        }
        if (temp!=null) {
            newNode.next = temp.next.next;
            temp.next = newNode;
        }
        else System.out.println("Previous node is null");
    }
    return head;
}
```

Q3. Given the head of a sorted linked list, delete all duplicates such that each element appears only once. Return the linked list sorted as well.

Solution:

```
public Node duplicateRemover(Node head){
     if (head==null)return null;
     if (head.next==null)return head;
     Node temp=head;
     while(temp!=null){
         if (temp.next!=null && temp.data==temp.next.data){
             temp.next=temp.next.next;
         }
         temp=temp.next;
     }
     return head;
}
```

Q4. Given the head of a singly linked list, return true if it is a palindrome or false otherwise.

Solution:

```
public boolean isPalindrome(Node head){
     Node h1=head;
     Node h2=reverse(getRightHalf(head));
     while(h1!=null&&h2!=null){
         if (h1.data!= h2.data)return false;
         h1=h1.next;
         h2=h2.next;
     }
     return true;
}
private Node getRightHalf(Node head){
     if (head==null)return null;
     Node slow=head;
     Node fast=head.next;
     while (fast!=null && fast.next!=null){
         slow=slow.next;
         fast=fast.next.next;
     }
     return slow.next;
}
private Node reverse(Node head){
     if (head==null || head.next==null)return head;
     Node p=head;
     Node prev=null;
     while(p!=null){
         Node nextTemp=p.next;
         p.next=prev;
         prev=p;
         p=nextTemp;
     }
     return prev;
}
```

Q5. Given two numbers represented by two lists, write a function that returns the sum list. The sum list is a list representation of the addition of two input numbers.

Solution:

```
public class Llist5 {
    Node head;
    class Node {
        int data;
        Node next;
        Node(int d) {
            data = d;
            next = null;
        }
    }
    // for creating linked lists
    public void insertAtEnd(int newData) {
        Node newNode = new Node(newData);
        if (head == null) {
            head = new Node(newData);
            return;
        }
        Node temp = head;
        while (temp.next != null) {
            temp = temp.next;
        }
        temp.next = newNode;
    }
    // to display list nodes
    public void displayLL(Node head1) {
        Node cur = head1;
        while (cur != null) {
            System.out.print(cur.data + " ");
            cur = cur.next;
        }
    }
    public Node addTwoList(Node first,Node second){
        first=reverse(first);
        second=reverse(second);
        Node ans=add(first,second);
        return ans;
    }
    //to reverse the LinkedList
    private Node reverse(Node heads){
        Node curr=heads;
        Node prev=null;
        Node next=null;
        while (curr!=null){
            next=curr.next;
            curr.next=prev;
            prev=curr;
            curr=next;
        }
        return prev;
    }

    private Node add(Node first,Node second){
        int carry=0;
        Node ansHead = null;
```

```
        while(first != null || second != null|| carry != 0) {
            int val1 = 0;
            if(first != null){
                val1=first.data;
                first=first.next;
            }
            int val2 = 0;
            if(second != null){
                val2= second.data;
                second=second.next;
            }
            int sum = carry + val1 + val2;
            int digit = sum%10;

            ansHead=insertAtTail(ansHead, digit);

            carry = sum/10;
        }
        return ansHead;
    }
    //To add the addition result into new list
    private Node insertAtTail(Node head1,int val) {
        Node temp = new Node(val);
        temp.next=head1;
        head1 = temp;
        return head1;
    }
   public static void main(String[] args) {
        Llist5 list1 = new Llist5(); //list1 = 75946
        list1.insertAtEnd(7);
        list1.insertAtEnd(5);
        list1.insertAtEnd(9);
        list1.insertAtEnd(4);
        list1.insertAtEnd(6);

        Llist5 list2=new Llist5();    //list2 = 84
        list2.insertAtEnd(8);
        list2.insertAtEnd(4);

        Node ans=list1.addTwoList(list1.head, list2.head);
        System.out.println("Resultant List:");
        list1.displayLL(ans);
    }
}
```