Assignment Question 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.

Examples:

Input: 14->21->11->30->10, X = 14

Output: Yes

Explanation: 14 is present in the linked list.

Input: 6->21->17->30->10->8, X = 13

Output: No

Solution:

public class find {

    public class Node {

        int data;

        Node next;

        Node(int data)  {

        this.data = data;

        next = null;

        }

    }

    private static Node head;

    public void add(int data)   {

        Node newnode = new Node(data);

        newnode.next = head;

        head = newnode;

    }

    public static String find(int x)

    {

        Node current = head;

        while (current != null) {

            if (current.data == x)

                return "Yes";

            current = current.next;

        }

        return "No";

    }

    public void Display(){

        Node temp=head;

        if(temp==null){

            System.out.println("Linked list is Empty");

            return;

        }

        while(temp!=null ){

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

            temp=temp.next;

        }System.out.println();

    }

    public static void main(String args[]){

        find list = new find();

        list.add(14);

        list.add(21);

        list.add(11);

        list.add(30);

        list.add(10);

        System.out.println("Current Linked List");

        list.Display();

        System.out.println("14 is available in given list: "+find(14));

        System.out.println("100 is available in given list: "+find(100));

    }

}

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.

public class LinkedListInsertionAtMiddle {

    public class Node {

        int data;

        Node next;

        Node(int data)  {

        this.data = data;

        next = null;

        }

    }

    public void Display(){

        Node temp=head;

        if(temp==null){

            System.out.println("Linked list is Empty");

            return;

        }

        while(temp!=null ){

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

            temp=temp.next;

        }System.out.println();

    }

    private static Node head;

    public void insertAtBegin(int data){

        Node newNode= new Node(data);

        if(head==null){

            head=newNode;

            return;

        }

        newNode.next=head;//linked nodes

        head=newNode;//head position changed

    }

    public void insertionAtMiddle(int data,int index){

        if(index==0){

            insertAtBegin(data);

            return;

        }

        Node newNode= new Node(data);

        Node temp=head;

        int i=0;

        while(i<(index-1)){

            temp=temp.next;

            i++;

        }

        newNode.next=temp.next;

        temp.next=newNode;

    }

    public static void main(String[] args) {

        LinkedListInsertionAtMiddle list =new LinkedListInsertionAtMiddle();

        list.insertAtBegin(23);

        list.insertAtBegin(88);

        list.insertAtBegin(923);

        list.insertAtBegin(3);

        list.Display();

        list.insertionAtMiddle(99, 2);

        list.insertionAtMiddle(34,0);

        list.Display();

    }

}

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.

public class linklistUnique {

    public class Node {

        int data;

        Node next;

        Node(int data)  {

        this.data = data;

        next = null;

        }

    }

    public void Display(){

        Node temp=head;

        if(temp==null){

            System.out.println("Linked list is Empty");

            return;

        }

        while(temp!=null ){

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

            temp=temp.next;

        }System.out.println();

    }

    public void add(int data)   {

        Node newnode = new Node(data);

        newnode.next = head;

        head = newnode;

    }

    private static Node head;

    public static void main(String[] args) {

        linklistUnique list=new linklistUnique();

        list.add(300);

        list.add(77);

        list.add(66);

        list.add(23);

        list.add(23);

        list.add(11);

        list.Display();

        list.removeDuplicate();

        list.Display();

    }

    private void removeDuplicate() {

        if(head == null || head.next == null){

            return ;

        }

        Node temp = head;

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

            if(temp.data == temp.next.data){

                temp.next = temp.next.next;

            }

            else{

                temp = temp.next;

            }

        }

        return;

    }

}

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

public class palindromeCheck {

    public class Node {

        int data;

        Node next;

        Node(int data)  {

        this.data = data;

        next = null;

        }

    }

    public void Display(){

        Node temp=head;

        if(temp==null){

            System.out.println("Linked list is Empty");

            return;

        }

        while(temp!=null ){

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

            temp=temp.next;

        }System.out.println();

    }

    public void add(int data)   {

        Node newnode = new Node(data);

        newnode.next = head;

        head = newnode;

    }

    private static Node head;

    public static void main(String[] args) {

        palindromeCheck list=new palindromeCheck();

         list.add(11);

         list.add(22);

        list.add(11);

        list.Display();

        System.out.println("Given list is palindrome: "+list.palindromeCheck());

    }

    private boolean palindromeCheck() {

       //Brute force

        // String s1="";

        // String s2="";

        // Node temp=head;

        // if(temp==null){

        //     return true;

        // }

        // while(temp!=null){

        //     s1=temp.data+s1;

        //     s2=s2+temp.data;

        //     temp=temp.next;

        // }

        // //System.out.println(s1+" "+s2);

        // if(s1.equals(s2)){

        //     return true;

        // }

        // return false;

        Node temp=head;

        int num=0;

        int n=0;

        int m=1;

        if(temp==null){

            return true;

        }

        while(temp!=null){

            num=num\*10+temp.data;

            n=n+temp.data\*m;

            m\*=10;

            temp=temp.next;

        }

        return num==n;

    }

}

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.

public class twoLinkedListSum {

    public class Node {

        int data;

        Node next;

        Node(int data)  {

        this.data = data;

        next = null;

        }

    }

    public void Display(){

        Node temp=head;

        if(temp==null){

            System.out.println("Linked list is Empty");

            return;

        }

        while(temp!=null ){

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

            temp=temp.next;

        }System.out.println();

    }

    public void add(int data)   {

        Node newnode = new Node(data);

        newnode.next = head;

        head = newnode;

    }

    private  Node head;

    public static void main(String[] args) {

        twoLinkedListSum list=new twoLinkedListSum();

        list.add(1);

         list.add(2);

        list.add(1);

        System.out.println("Linked list first is: ");

        list.Display();

        twoLinkedListSum list2=new twoLinkedListSum();

        System.out.println("Linked list second is: ");

        list2.add(4);

         list2.add(5);

         list2.Display();

       System.out.println("After sum both linked list we get: ");

       list=sum(list,list2);

       list.Display();

    }

    private static twoLinkedListSum sum(twoLinkedListSum list, twoLinkedListSum list2) {

        Node head1=list.head;

        Node head2=list2.head;

        String s1="",s2="";

        twoLinkedListSum newlist=new twoLinkedListSum();

        while(head1!=null || head2!=null){

            if(head1!=null){

                s1+=head1.data;

                head1=head1.next;

            }

            if(head2!=null){

                s2+=head2.data;

                head2=head2.next;            }

        }

        int sum=Integer.parseInt(s2)+Integer.parseInt(s1);

        if(sum==0){

            newlist.add(0);

            return newlist;

        }

        while(sum!=0){

            newlist.add(sum%10);

            sum/=10;

        }

        return newlist;

    }

}