ASSIGNMENT 3

Q1

```
public class LinkedList{
	Node head;
	static class Node{
		int data;
		Node link;

	public Node(int d){
		data d;
		link = null;
		}

	}

	public void display(){
		Node n = head;
		while(n != null){
			System.out.print(n.data + "  ");
			n = n.link;
			}
	}

	public static void main(String args[]){
		LinkedList l1 = new LinkedList();

		l1.head = new Node(11);
		Node second = new Node(22);
		Node third = new Node(33);

		l1.head.link = second;
		second.link = third;

		l1.display();
		}
	}
```

Q2

```
class DLL{
	Node head;
	static class Node{
		int data;
		Node prev;
		Node next;

		Node(int d)
		{
			data = d;
			prev = next= null;
		}
	}

	public void insert(int new_data){
		Node new_node = new Node(new_data);
		new_node.next = head;
		new_node.prev = null;

		if(head != null)
			head.prev = new_node;

		head = new_node;
	}


	public void insertAfter(Node prev, int new_data)
	{
		if(prev == null)

			return;

		Node new_node = new Node(new_data);
		new_node.next = prev.next;
		prev.next = new_node;
		new_node.prev = prev;
		new_node.next.prev = new_node;

	}


	public void append(int new_data)
```

```
{
        Node new_node = new Node(new_data);
        Node n = head;

        if(head == null){
                new_node.prev = null;
                head = new_node;
                return;
        }

        while(n.next != null){
                n= n.next;
        }

        n.next = new_node;
        new_node.prev = n;

}


public void deleteNode(Node del)
{

        if(head == null)
                return;


        if(head == del)
        {
                head = del.next;  //head = head.next;???
                head.prev = null;
        }

        if(del.next != null)              {
                del.next.prev = del.prev;
        }

        if(del.prev != null){
                del.prev.next = del.next;

        }


}
```

```
	public void revdisplay(Node n)
	{
		Node p = null;
		System.out.println("Forward direction : ");
		while(n != null)
		{

			System.out.print(n.data + " ");
			p=n;
			n = n.next;
		}

		System.out.println("-----------------------------------");
		System.out.println("Backward direction : ");
		while(p != null) //backward print while loop
		{
			System.out.print( p.data +" ");
			p = p.prev;
		}
	}

	public static void main(String args[])
	{
			DLL d1 = new DLL();
			d1.insert(5);
			d1.insert(10);
			d1.insert(15);
			//d1.revdisplay(d1.head);
			d1.insertAfter(d1.head, 7);
			d1.insertAfter(d1.head.next, 8);
			d1.append(2);
			//d1.revdisplay(d1.head);
			d1.deleteNode(head.next.next);
			d1.revdisplay(d1.head);
	}
}
```

Q3

```
public class ReversedLinkedList {
```

```
Node head;

static class Node {
    int data;
    Node next;

    Node(int d) {
        this.data = d;
        next = null;
    }
}

public void display() {
    Node temp = head;
    while (temp != null) {
        System.out.print(temp.data + " --> ");
        temp = temp.next;
    }
    System.out.println("null");
}

public void reverse() {
    Node prev = null;
    Node current = head;
    Node nextNode = null;

    while (current != null) {
        nextNode = current.next;
        current.next = prev;
        prev = current;
        current = nextNode;
    }

    head = prev;
}

public static void main(String args[]) {
    ReversedLinkedList r = new ReversedLinkedList();
    r.head = new Node(11);
    Node second = new Node(22);
    Node third = new Node(33);

    r.head.next = second;
    second.next = third;
```

```
        System.out.println("Original list:");
        r.display();

        r.reverse();

        System.out.println("Reversed list:");
        r.display();
    }
}
```

Q4

```
public class D3Que5 {
    static class Node{
        int data;
        Node link;
        Node(int x){
            data = x;
            link = null;
        }
    }
    static void pushNode(Node[] head, int data){
        Node new_node = new Node(data);
        new_node.link = head[0];
        head[0] = new_node;
    }
    static int getMiddle(Node head){
        Node ptr1 = head;
        Node ptr2 =head;
        while(ptr2 != null && ptr2.link != null){
            ptr2 = ptr2.link.link;
            ptr1 = ptr1.link;
        }
        return ptr1.data;
    }
    public static void main(String[] args){
        Node[] head = new Node[1];
        for (int i=0; i<7; i++){
            pushNode(head,i);
        }
        System.out.println("Middle Value:" +getMiddle(head[0]));
    }
}
```

Q5

```
public class D3Que5 {
    static class Node{
        int data;
        Node link;
        Node(int x){
            data = x;
            link = null;
        }
    }
    static void pushNode(Node[] head, int data){
        Node new_node = new Node(data);
        new_node.link = head[0];
        head[0] = new_node;
    }
    static int getMiddle(Node head){
        Node ptr1 = head;
        Node ptr2 =head;
        while(ptr2 != null && ptr2.link != null){
            ptr2 = ptr2.link.link;
            ptr1 = ptr1.link;
        }
        return ptr1.data;
    }
    public static void main(String[] args){
        Node[] head = new Node[1];
        for (int i=0; i<7; i++){
            pushNode(head,i);
        }
        System.out.println("Middle Value:" +getMiddle(head[0]));
    }
}
```

Q6

```
public class DetectLoop{
            Node head;
```

```
    static class Node{
        int data;
        Node next;
        Node(int d){
            data = d;
            next = null;
        }
    }
    public static boolean checkCycle(Node head){
        if(head == null || head.next == null){
            return false;
        }
        Node slow = head;
        Node fast = head.next;
        while(slow != fast){
            if(fast == null || fast.next ==null){
                return false;
            }
            slow = slow.next;
            fast = fast.next.next;
        }
        return true;
    }


    public static void main(String[] args){
        DetectLoop list1 = new DetectLoop();
        list1.head = new Node(1);
        list1.head.next = new Node(2);
        list1.head.next.next = new Node(3);
        list1.head.next.next.next = new Node(10);
        list1.head.next.next.next.next = list1.head.next;

        System.out.println("Result: "+checkCycle(list1.head));
        System.out.println();
        System.out.println("(True = Cycle detected / False = Not Circular)");
    }
}
```

Q7

```
class ListNode {
    int val;
    ListNode next;
    ListNode(int x) {
        val = x;
        next = null;
    }
}

public class LinkedListCycle {
    public ListNode detectCycle(ListNode head) {
        if (head == null || head.next == null)
            return null;

        ListNode slow = head;
        ListNode fast = head;

        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;

            if (slow == fast) {
                ListNode start = head;
                while (start != slow) {
                    start = start.next;
                    slow = slow.next;
                }
                return start;
            }
        }
        return null;
    }

    public static void main(String[] args) {
        ListNode head = new ListNode(3);
        head.next = new ListNode(2);
        head.next.next = new ListNode(0);
        head.next.next.next = new ListNode(-4);
        head.next.next.next.next = head.next;

        LinkedListCycle solution = new LinkedListCycle();
        ListNode startNode = solution.detectCycle(head);
        if (startNode != null) {
            System.out.println("Start node of the loop is: " + startNode.val);
        } else {
            System.out.println("No loop found in the linked list.");
        }
    }
}
```

Q8

```
class ListNode {
    int val;
    ListNode next;
    ListNode(int x) {
        val = x;
        next = null;
    }
}

public class NthFromEnd {
    public int nthFromEnd(ListNode head, int n) {
        ListNode slow = head;
        ListNode fast = head;


        for (int i = 0; i < n; i++) {
            if (fast == null) return -1;
            fast = fast.next;
        }


        while (fast != null) {
            slow = slow.next;
            fast = fast.next;
        }


        if (slow != null)
            return slow.val;
        else
            return -1;
    }
    public static void main(String[] args) {
        ListNode head = new ListNode(1);
        head.next = new ListNode(2);
        head.next.next = new ListNode(3);
        head.next.next.next = new ListNode(4);
        head.next.next.next.next = new ListNode(5);

        int n = 2;

        NthFromEnd solution = new NthFromEnd();
        int nthFromEnd = solution.nthFromEnd(head, n);

        if (nthFromEnd != -1) {
            System.out.println("The " + n + "th element from the end is: " + nthFromEnd);
        } else {
            System.out.println("Invalid input or element not found.");
        }
    }
}
```

Q9

```
public boolean isPalindrome(ListNode head) {
        List<Integer> list = new ArrayList();
        while(head != null) {
            list.add(head.val);
            head = head.next;
        }

        int left = 0;
        int right = list.size()-1;
        while(left < right && list.get(left) == list.get(right)) {
            left++;
            right--;
        }
        return left >= right;
    }
```

Q10

```
class Node{
    int data;
    Node next;
    public Node(int data){
        this.data = data;
        this.next = null;
    }
}
class LinkList{
    Node head;
    LinkList(){
        this.head = null;
    }
    LinkList(int data){
        this.head = new Node(data);
    }
    public void add(int data){
        Node newNode = new Node(data);
        if(head == null){
            head = newNode;
            return;
        }
        Node current = head;
        while(current.next != null){
            current = current.next;
        }
        current.next = newNode;
    }
}
/**
 * ReverseList
 */
public class AddListsNumber {
    public static int toNumber(Node head){
        Node current = head;
        int num = 0;
        while(current != null){
            num = 10 * num + current.data;
            current = current.next;
        }
        return num;
    }
    public static void main(String[] args) {
        LinkList list1 = new LinkList();
        LinkList list2 = new LinkList();
        list1.add(1);
        list1.add(2);
        list1.add(3);
    ///////////////////////
        list2.add(1);
        list2.add(2);
        list2.add(3);
        Node head1=  list1.head;
        Node head2 = list2.head;
        int num1 = toNumber(head1);
        int num2 = toNumber(head2);

        System.out.println(num1 + num2);
    }
}
```