1.Implement singly linked list in java

--------------------------------------

class Node {

int data;

Node next;

Node(int data) {

this.data = data;

this.next = null;

}

}

class SinglyLinkedList {

Node head;

SinglyLinkedList() {

this.head = null;

}

// Method to insert a node at the end of the linked list

public void insert(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

} else {

Node temp = head;

while (temp.next != null) {

temp = temp.next;

}

temp.next = newNode;

}

}

// Method to display the linked list

public void display() {

Node temp = head;

while (temp != null) {

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

temp = temp.next;

}

System.out.println();

}

}

-------------------------------------

-------------------------------------

2.Implement Doubly linked list in java

--------------------------------------

class DoublyNode {

int data;

DoublyNode prev;

DoublyNode next;

DoublyNode(int data) {

this.data = data;

this.prev = null;

this.next = null;

}

}

class DoublyLinkedList {

DoublyNode head;

DoublyLinkedList() {

this.head = null;

}

// Method to insert a node at the end of the linked list

public void insert(int data) {

DoublyNode newNode = new DoublyNode(data);

if (head == null) {

head = newNode;

} else {

DoublyNode temp = head;

while (temp.next != null) {

temp = temp.next;

}

temp.next = newNode;

newNode.prev = temp;

}

}

// Method to display the linked list

public void display() {

DoublyNode temp = head;

while (temp != null) {

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

temp = temp.next;

}

System.out.println();

}

}

-------------------------------------

-------------------------------------

3.How to reverse a linked list in java

--------------------------------------

public class ReverseLinkedList {

public static Node reverse(Node head) {

Node prev = null;

Node current = head;

Node next = null;

while (current != null) {

next = current.next;

current.next = prev;

prev = current;

current = next;

}

head = prev;

return head;

}

}

-------------------------------------

-------------------------------------

4.How to merge two linked list in sorted order in java

------------------------------------------------------

public class MergeSortedLinkedList {

public static Node merge(Node head1, Node head2) {

if (head1 == null) return head2;

if (head2 == null) return head1;

if (head1.data < head2.data) {

head1.next = merge(head1.next, head2);

return head1;

} else {

head2.next = merge(head1, head2.next);

return head2;

}

}

}

-------------------------------------

-------------------------------------

5.How to find middle element of linked list in java

---------------------------------------------------

public class FindMiddleElement {

public static Node findMiddle(Node head) {

if (head == null) return null;

Node slow = head, fast = head;

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

slow = slow.next;

fast = fast.next.next;

}

return slow;

}

}

-------------------------------------

-------------------------------------

6.How to detect a loop in linked list in java

---------------------------------------------

public class DetectLoop {

public static boolean detectLoop(Node head) {

if (head == null || head.next == null) return false;

Node slow = head, fast = head;

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

slow = slow.next;

fast = fast.next.next;

if (slow == fast) return true;

}

return false;

}

}

-------------------------------------

-------------------------------------

7.Find start node of loop in linkedlist

---------------------------------------

public class StartNodeOfLoop {

public static Node findStartNode(Node head) {

if (head == null || head.next == null) return null;

Node slow = head, fast = head;

boolean loopExists = false;

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

slow = slow.next;

fast = fast.next.next;

if (slow == fast) {

loopExists = true;

break;

}

}

if (!loopExists) return null;

slow = head;

while (slow != fast) {

slow = slow.next;

fast = fast.next;

}

return slow;

}

}

-------------------------------------

-------------------------------------

8.How to find nth element from end of linked list

-------------------------------------------------

public class NthFromEnd {

public static Node findNthFromEnd(Node head, int n) {

if (head == null) return null;

Node fast = head, slow = head;

for (int i = 0; i < n; i++) {

if (fast == null) return null; // n is greater than the length of the list

fast = fast.next;

}

while (fast != null) {

slow = slow.next;

fast = fast.next;

}

return slow;

}

}

-------------------------------------

-------------------------------------

9.How to check if linked list is palindrome in java

---------------------------------------------------

public class IsPalindrome {

public static boolean isPalindrome(Node head) {

if (head == null || head.next == null) return true;

Node slow = head, fast = head;

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

slow = slow.next;

fast = fast.next.next;

}

Node secondHalf = reverse(slow);

while (secondHalf != null) {

if (head.data != secondHalf.data) return false;

head = head.next;

secondHalf = secondHalf.next;

}

return true;

}

public static Node reverse(Node head) {

Node prev = null;

Node current = head;

Node next = null;

while (current != null) {

next = current.next;

current.next = prev;

prev = current;

current = next;

}

return prev;

}

}

-------------------------------------

-------------------------------------

10.Add two numbers represented by linked list in java

-----------------------------------------------------

public class AddTwoNumbers {

public static Node addTwoNumbers(Node l1, Node l2) {

Node dummy = new Node(0);

Node p = l1, q = l2, current = dummy;

int carry = 0;

while (p != null || q != null) {

int x = (p != null) ? p.data : 0;

int y = (q != null) ? q.data : 0;

int sum = carry + x + y;

carry = sum / 10;

current.next = new Node(sum % 10);

current = current.next;

if (p != null) p = p.next;

if (q != null) q = q.next;

}

if (carry > 0) {

current.next = new Node(carry);

}

return dummy.next;

}

}

-------------------------------------

-------------------------------------