**1.Implement singly linked list in java**

class LinkedList {

Node head;

static class Node { //node created

int data;

Node next;

Node(int d) {

this.data = d;

this.next = null;

}

}

void display() { //transvers in the node

Node current = head;

while (current != null) {

System.out.print(current.data + "-->");

current = current.next;

}

System.out.println();

}

public static void main(String[] args) {

LinkedList l = new LinkedList();

l.head = new Node(11);

Node second = new Node(12);

Node third = new Node(13);

l.head.next = second;

second.next = third;

l.display();

}

}

**2.Implement Doubly linked list in java**

class DoublyLinkedList{

Node head;

Node tail;

static class Node {

int data;

Node prev;

Node next;

Node(int d) {

data = d;

prev = null;

next = null;

}

}

public void insertBeginning(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

tail = newNode;

} else {

newNode.next = head;

head.prev = newNode;

head = newNode;

}

}

public void insertEnd(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

tail = newNode;

} else {

tail.next = newNode;

newNode.prev = tail;

tail = newNode;

}

}

public void display() {

Node current = head;

while (current != null) {

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

current = current.next;

}

System.out.println();

}

public static void main(String[] args) {

DoublyLinkedList dll = new DoublyLinkedList();

System.out.println("Insert at beggining :");

dll.insertBeginning(1);

dll.insertBeginning(2);

dll.insertBeginning(3);

dll.display();

System.out.println("Insert at end :");

dll.insertEnd(4);

dll.insertEnd(5);

dll.insertEnd(6);

dll.display();

}

}

**3.How to reverse a linked list in java**

class Abhi {

static Node head;

static class Node{

int data;

Node next;

//Node prev;

Node(int d){

data = d;

next = null;

//prev = null;

}

}

static void display(){

Node curr = head;

Node prev = null;

Node temp = null;

System.out.println("Forward list");

while(curr != null){

System.out.print(curr.data+"->");

temp = curr.next;

curr.next = prev;

prev = curr;

curr = temp;

}

System.out.println("\nReverse list");

while(prev != null){

System.out.print(prev.data+"->");

prev = prev.next ;

}

}

public static void main(String[] args) {

head = new Node(10);

Node first = new Node(20);

Node second = new Node(30);

head.next = first;

first.next = second;

display();

}

}

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

class Abhi{

static Node head1;

static Node head2;

static class Node{

int data;

Node next;

//Node prev;

Node(int d){

data = d;

next = null;

//prev = null;

}

}

static void display(Node n){

Node curr = n;

Node prev = null;

Node temp = null;

System.out.println("Node");

while(curr != null){

System.out.print(curr.data+"->");

curr = curr.next;

}

System.out.println();

}

static Node mergeNode(Node h1, Node h2){

Node n = h1;

Node n2 = h2;

Node merge = new Node(0);

Node temp = merge;

while(n != null && n2 != null){

if(h1 == null){

temp.next = h2;

break;

}

if(h2 == null ){

temp.next = h1;

break;

}

if(n.data <= n2.data){

temp.next = n;

n = n.next;

}

else{

temp.next = n2;

n2 = n2.next;

}

temp=temp.next;

}

return merge.next;

}

public static void main(String[] args) {

head1 = new Node(10);

Node first = new Node(20);

Node second = new Node(30);

head1.next = first;

//fst.prev = head;

first.next = second;

head2 = new Node(5);

Node third = new Node(15);

Node fourth = new Node(25);

head2.next = third;

//fst.prev = head;

third.next = fourth;

//sec.prev = fst;

display(head1);

display(head2);

Node m = mergeNode(head1, head2);

display(m);

}

}

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

class Abhi{

static Node head1;

static Node head2;

static class Node{

int data;

Node next;

//Node prev;

Node(int d){

data = d;

next = null;

//prev = null;

}

}

static void display(Node n){

Node curr = n;

Node prev = null;

Node temp = null;

System.out.println("Node");

while(curr != null){

System.out.print(curr.data+"->");

curr = curr.next;

}

System.out.println();

}

static void midPoint(){

Node slow = head1;

Node fast = head1.next;

while(slow != null && fast != null){

slow = slow.next;

fast = fast.next.next;

}

System.out.println(slow.data);

}

public static void main(String[] args) {

head1 = new Node(10);

Node f = new Node(20);

Node s = new Node(30);

Node t = new Node(40);

Node fr = new Node(50);

Node fv = new Node(60);

Node six = new Node(70);

head1.next = fst;

f.next = s;

s.next = t;

t.next = fr;

fr.next = fv;

fv.next = six;

display(head1);

midPoint();

}

}

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

class Abhi {

static Node head1;

static Node head2;

static class Node{

int data;

Node next;

//Node prev;

Node(int d){

data = d;

next = null;

//prev = null;

}

}

static void display(Node n){

Node curr = n;

Node prev = null;

Node temp = null;

System.out.println("Node");

while(curr != null){

System.out.print(curr.data+"->");

curr = curr.next;

}

System.out.println();

}

static void detectLoop(){

Node slow = head1;

Node fast = head1.next;

int m=0;

while(slow != fast){

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

m++;

break;

}

slow = slow.next;

fast = fast.next.next;

}

if(m>0){

System.out.println("Loop Not detected");

}else{

System.out.println("Loop detected");

}

}

public static void main(String[] args) {

head1 = new Node(10);

Node fst = new Node(20);

Node sec = new Node(30);

Node thr = new Node(40);

Node fr = new Node(50);

Node fv = new Node(60);

Node six = new Node(70);

head1.next = fst;

fst.next = sec;

sec.next = thr;

thr.next = fr;

fr.next = fv;

fv.next = six;

display(head1);

detectLoop();

}

}

**7.Find start node of loop in linkedlist**

class Abhi {

static Node head1;

static Node head2;

static class Node{

int data;

Node next;

//Node prev;

Node(int d){

data = d;

next = null;

//prev = null;

}

}

static void display(Node n){

Node curr = n;

Node prev = null;

Node temp = null;

System.out.println("Node");

while(curr != null){

System.out.print(curr.data+"->");

curr = curr.next;

}

System.out.println();

}

static void detectStartofLoop(){

Node slow = head1;

Node fast = head1.next;

int m=0;

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

slow = slow.next;

fast = fast.next.next;

if(slow == fast){

m++;

break;

}

}

if(m>0){

while(slow != fast){

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

//m++;

break;

}

slow = slow.next;

fast = fast.next.next;

}

}

System.out.println("Starting point = "+ slow.next.data);

}

public static void main(String[] args) {

head1 = new Node(50);

Node fst = new Node(30);

Node sec = new Node(20);

Node thr = new Node(40);

Node fr = new Node(50);

Node fv = new Node(60);

Node six = new Node(70);

head1.next = fst;

fst.next = sec;

sec.next = thr;

thr.next = fr;

fr.next = fv;

fv.next = six;

six.next = head1;

detectStartofLoop();

}

}

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

class Abhi {

static Node head1;

static Node head2;

static class Node{

int data;

Node next;

//Node prev;

Node(int d){

data = d;

next = null;

//prev = null;

}

}

static void display(Node n){

Node curr = n;

Node prev = null;

Node temp = null;

System.out.println("Node");

while(curr != null){

System.out.print(curr.data+"->");

curr = curr.next;

}

System.out.println();

}

static void midPoint(){

Node slow = head1;

int m=1;

if(slow == null){

m=0;

}

else if (slow.next == null){

m=1;

}

else{

while(slow.next != null){

m++;

slow = slow.next;

}

}

System.out.println("Last Element is "+slow.data+" at position of " + m);

}

static void getElement(int n){

Node curr = head1;

Node p = null;

Node temp = null;

System.out.println("Node");

while(curr != null){

temp = curr.next;

curr.next = p;

p = curr;

curr = temp;

}

if (n==0){

System.out.println(p.data);

}else{

int i =0;

while(p != null && i< n-1 ){

p=p.next;

i++;

}

System.out.println(p.data);

}

}

public static void main(String[] args) {

head1 = new Node(10);

Node fst = new Node(20);

Node sec = new Node(30);

Node thr = new Node(40);

Node fr = new Node(50);

Node fv = new Node(60);

Node six = new Node(70);

head1.next = fst;

fst.next = sec;

sec.next = thr;

thr.next = fr;

fr.next = fv;

fv.next = six;

display(head1);

getElement(2);

}

}

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

class Abhi {

static Node head1;

static Node head2;

static class Node{

int data;

Node next;

Node(int d){

data = d;

next = null;

}

}

static void display(Node n){

Node curr = n;

Node prev = null;

Node temp = null;

System.out.println("Node");

while(curr != null){

System.out.print(curr.data+"->");

curr = curr.next;

}

System.out.println();

}

static boolean isPalindrom(Node h1){

Node curr = h1;

String s1 = "";

String s2 = "";

Node temp = null;

Node p = null;

while(curr != null){

s1 = s1 + String.valueOf(curr.data);

temp = curr.next;

curr.next = p;

p = curr;

curr = temp;

}

while(p != null){

s2 = s2 + String.valueOf(p.data);

p = p.next;

}

System.out.println(s1+"\n"+s2);

return s1.equals(s2);

}

public static void main(String[] args) {

head1 = new Node(1);

Node fst = new Node(2);

Node sec = new Node(1);

head1.next = fst;

fst.next = sec;

display(head1);

System.out.println("LinkedList is Palindrome = "+isPalindrom(head1));

}

}

**10.Add two numbers represented by linked list in java**

class Abhi {

static Node head1;

static Node head2;

static class Node{

int data;

Node next;

Node(int d){

data = d;

next = null;

}

}

static void display(Node n){

Node curr = n;

Node prev = null;

Node temp = null;

System.out.println("Node");

while(curr != null){

System.out.print(curr.data+"->");

curr = curr.next;

}

System.out.println();

}

static void addOfTwoLL(Node h1,Node h2){

Node curr = h1;

Node p = h2;

String s1 = "";

String s2 = "";

while(curr != null){

s1 = s1 + String.valueOf(curr.data);

curr = curr.next;

}

while(p != null){

s2 = s2 + String.valueOf(p.data);

p = p.next;

}

int i = Integer.valueOf(s1);

int j = Integer.valueOf(s2);

System.out.println(i+j);

}

public static void main(String[] args) {

head1 = new Node(1);

Node fst = new Node(2);

Node sec = new Node(3);

head1.next = fst;

fst.next = sec;

head2 = new Node(1);

Node fst2 = new Node(2);

Node sec2 = new Node(3);

head2.next = fst2;

fst2.next = sec2;

display(head1);

addOfTwoLL(head1,head2);

}

}