

1.Implement singly linked list in java

2.How to reverse a linked list in java

Question 1 and 2:

```
class myList{
    Node head=null;
    class Node{
        int data;
        Node next;

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

    void insertAtbeginning(int new_data){
        Node new_node= new Node(new_data);
        new_node.next=head;
        head=new_node;
    }
    void insertAt(int new_data,int pos){
        Node temp=head;
        Node new_node= new Node(new_data);
        int count=1;
        if(pos==1){
            new_node.next=head;
            head=new_node;
        }
        else{
            while(count!=pos-1){
                temp = temp.next;
                count++;
            }
            new_node.next=temp.next;
            temp.next=new_node;
        }
    }
}
```

```

void insertAtEnd(int new_data){
    Node new_node= new Node(new_data);
    Node temp=head;
    if(temp==null){
        head=new_node;
        return;
    }
    else{
        while(temp.next!=null){
            temp=temp.next;
        }
        temp.next=new_node;
    }
}

```

```

void delete(int key){
    Node temp=head;
    Node prev=null;
    if(temp!=null && temp.data==key){
        head=temp.next;
        return;
    }
    else{
        while(temp!=null && temp.data!=key){
            prev=temp;
            temp=temp.next;
        }
        if(temp==null){
            return;
        }
        else{
            prev.next=temp.next;
        }
    }
}

```

```

void deleteAtPosition(int pos){
    if(head==null)
        return;
    Node temp = head;
    if(pos==0){

```

```

        head=temp.next;
        return;
    }
    for(int i=0; temp!=null && i<pos-1 ;i++){
        temp= temp.next;
    }
    if(temp==null || temp.next==null)
        return;
    temp.next=temp.next.next;
}

```

```

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

```

```

int count(){
    Node temp= head;
    int count=0;
    while(temp!=null){
        count++;
        temp=temp.next;
    }
    return count;
}

```

```

public boolean search(int x){
    Node temp = head;
    if(temp.data==x)
        return true;
    while(temp!=null){
        if(temp.data == x){

```

```

        return true;
    }
    temp=temp.next;
}
return false;
}

```

```

void reverse(){
    Node temp=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;
}

```

```

public static void main(String args[]){
    myList l1= new myList();
    System.out.println("Inserting 11 in empty list");
    l1.insertAtbeginning(11);
    l1.display();
    System.out.println("Inserting 10 at the start of list");
    l1.insertAtbeginning(10);
    l1.display();
    System.out.println("Inserting 9 at the start of list");
    l1.insertAtbeginning(9);
    l1.display();
    System.out.println("Inserting 55 at 2nd position");
    l1.insertAt(55,2);
    l1.display();

    System.out.println("Deleting 10");
    l1.delete(10);
    l1.display();
}

```

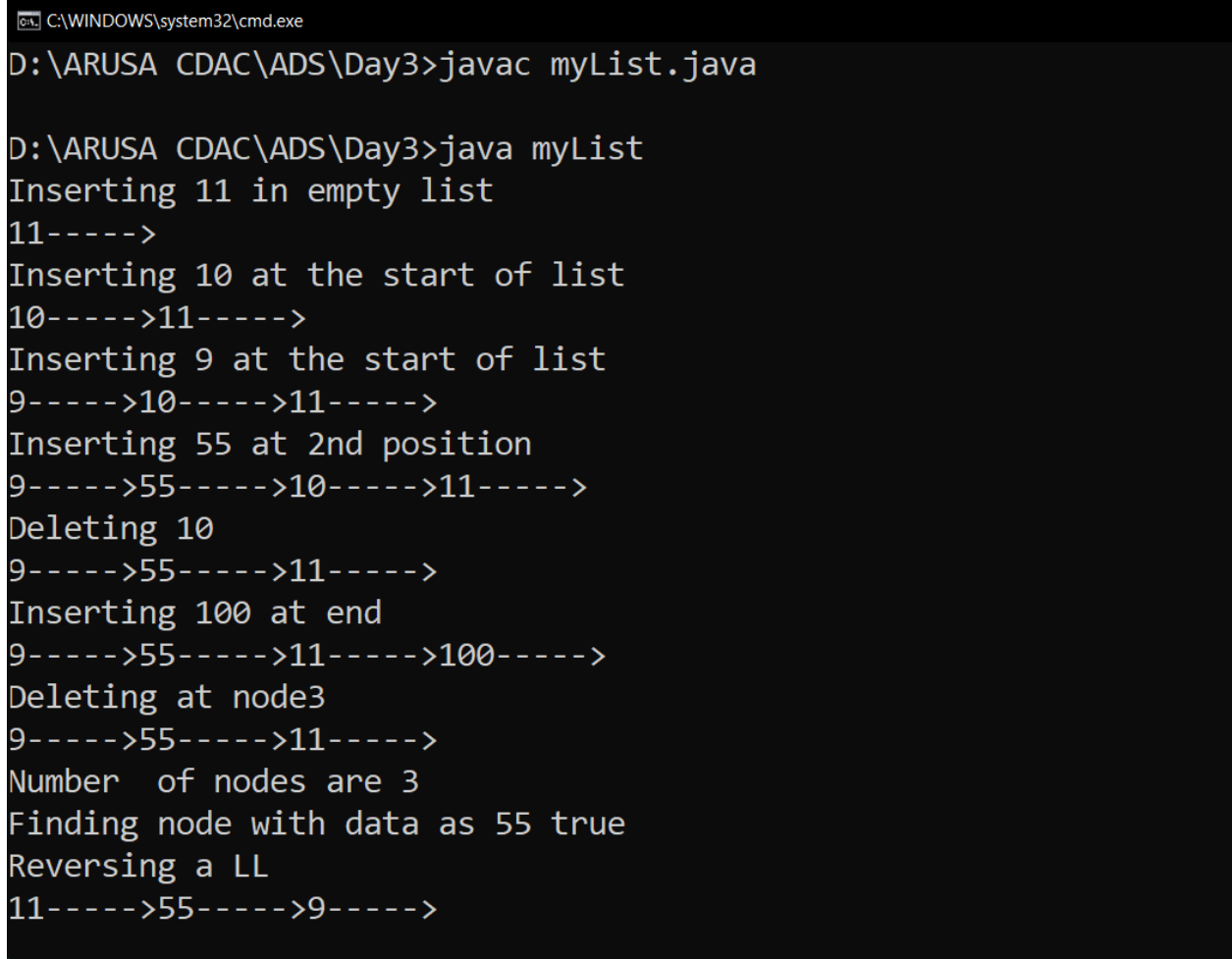
```

        System.out.println("Inserting 100 at end");
        l1.insertAtEnd(100);
        l1.display();

        System.out.println("Deleting at node3");
        l1.deleteAtPosition(3);
        l1.display();

        System.out.println("Number of nodes are " +l1.count());
        System.out.println("Finding node with data as 55 "+l1.search(55));
        l1.reverse();
        System.out.println("Reversing a LL");
        l1.display();
    }
}

```



```

C:\WINDOWS\system32\cmd.exe
D:\ARUSA CDAC\ADS\Day3>javac myList.java

D:\ARUSA CDAC\ADS\Day3>java myList
Inserting 11 in empty list
11----->
Inserting 10 at the start of list
10----->11----->
Inserting 9 at the start of list
9----->10----->11----->
Inserting 55 at 2nd position
9----->55----->10----->11----->
Deleting 10
9----->55----->11----->
Inserting 100 at end
9----->55----->11----->100----->
Deleting at node3
9----->55----->11----->
Number of nodes are 3
Finding node with data as 55 true
Reversing a LL
11----->55----->9----->

```

2.Implement Doubly linked list in java

```
class DLL{
    Node head;//instance
    static class Node{

        int data;
        Node next;

        Node(int d)
        {
            data=d;
            next=null;
        }
    }
    void display()
    {
        Node n = head;
        while(n!= null)
        {
            System.out.print(n.data+ "---> ");
            n=n.next;
        }
    }

    void insert(int new_data)
    {
        Node new_node = new Node(new_data);
        new_node.next = head;
        head = new_node;
    }

    void append(int new_data)
    {
        //new node is created
        Node new_node = new Node(new_data);
        //check list is empty ?
        if(head == null)
        {
            head = new_node;
            return;
        }
    }
}
```

```

    }
    //if list is not empty
    new_node.next = null;//condition for last node in the list

    Node last = head;//traverse ke liya new reference create kiya hai
    //last.next: last node tak le jayega
    while(last.next != null)
    {
        last = last.next;// shifting to next node
    }
    last.next = new_node;// connect to new node
    return;
}

```

```

void insertAfter(Node prev_node,int new_data)
{
    //check list is empty ?
    if(prev_node == null)
    {
        System.out.println("Empty list !!!");
        return;
    }
    //if list is not empty
    Node new_node = new Node(new_data);
    new_node.next = prev_node.next;
    prev_node.next = new_node;
}

```

```

void deleteNode(int key)//19
{
    Node temp = head, prev = null;
    if(temp != null && temp.data == key)
    {
        head = temp.next;
        return;
    }
    //Last or intermediate
    while(temp != null && temp.data != key)
    {
        prev = temp;
        temp = temp.next;
    }
}

```

```

        if(temp == null)
            return;
        prev.next = temp.next;
    }

void deletenode(int position)
{
    if(head == null)
        return;
    Node temp=head;
    //Deletion at the begining
    if(position == 0)
    {
        head=temp.next;
        return;
    }
    //Deletion at the intermediate node
    for(int i=0;temp != null && i<position-1;i++)
    {
        temp=temp.next;
    }
    if(temp == null || temp.next == null)
        return;
    Node last = temp.next.next;
    temp.next = last;
}

public static void main(String args[]){

    DLL L1 = new DLL();
    L1.head = new Node(9);

    L1.display();
    L1.insert(7);
    System.out.println();
    //L1.display();

    L1.display();
    L1.insert(5);
    System.out.println();
    //L1.display();
}

```



```
L1.display();  
L1.insert(10);  
System.out.println();  
//L1.display();
```

```
L1.insert(15);  
System.out.println();  
L1.display();
```

```
L1.append(23);  
System.out.println();  
L1.display();
```

```
L1.append(33);  
System.out.println();  
L1.display();
```

```
L1.insertAfter(L1.head.next, 53);  
System.out.println();  
L1.display();
```

```
L1.insertAfter(L1.head, 63);  
System.out.println();  
L1.display();
```

```
L1.insertAfter(L1.head.next.next, 53);  
System.out.println();  
L1.display();
```

```
L1.deleteNode(53);  
System.out.println();  
L1.display();
```

```
}
```

```
}
```

```

D:\ARUSA CDAC\ADS\Day3>java DLL
9--->
7---> 9--->
5---> 7---> 9--->

15---> 10---> 5---> 7---> 9--->
15---> 10---> 5---> 7---> 9---> 23--->
15---> 10---> 5---> 7---> 9---> 23---> 33--->
15---> 10---> 53---> 5---> 7---> 9---> 23---> 33--->
15---> 63---> 10---> 53---> 5---> 7---> 9---> 23---> 33--->
15---> 63---> 10---> 53---> 53---> 5---> 7---> 9---> 23---> 33--->
15---> 63---> 10---> 53---> 5---> 7---> 9---> 23---> 33--->
D:\ARUSA CDAC\ADS\Day3>

```

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

```

import java.io.*;
import java.math.*;
import java.security.*;
import java.text.*;
import java.util.*;
import java.util.concurrent.*;
import java.util.regex.*;

public class Solution {

    static class SinglyLinkedListNode {
        public int data;
        public SinglyLinkedListNode next;

        public SinglyLinkedListNode(int nodeData) {
            this.data = nodeData;
            this.next = null;
        }
    }

    static class SinglyLinkedList {
        public SinglyLinkedListNode head;
        public SinglyLinkedListNode tail;

        public SinglyLinkedList() {
            this.head = null;
            this.tail = null;
        }
    }

```

```
}
```

```
public void insertNode(int nodeData) {
```

```
    SinglyLinkedListNode node = new SinglyLinkedListNode(nodeData);
```

```
    if (this.head == null) {
```

```
        this.head = node;
```

```
    } else {
```

```
        this.tail.next = node;
```

```
    }
```

```
    this.tail = node;
```

```
}
```

```
}
```

```
public static void printSinglyLinkedList(SinglyLinkedListNode node, String sep, BufferedWriter  
bufferedWriter) throws IOException {
```

```
    while (node != null) {
```

```
        bufferedWriter.write(String.valueOf(node.data));
```

```
        node = node.next;
```

```
        if (node != null) {
```

```
            bufferedWriter.write(sep);
```

```
        }
```

```
    }
```

```
}
```

```
static SinglyLinkedListNode mergeLists(SinglyLinkedListNode head1, SinglyLinkedListNode head2) {
```

```
    if(head1==null) return head2;
```

```
    else if(head2==null) return head1;
```

```
    SinglyLinkedListNode ansHead, temp;
```

```
    if(head1.data< head2.data) {
```

```
        ansHead=head1;
```

```
        head1=head1.next;
```

```
    }
```

```
    else {
```

```
        ansHead=head2;
```

```
        head2=head2.next;
```

```
    }
```

```
    temp=ansHead;
```

```
    while(head1!=null && head2!=null) {
```

```

        if(head1.data< head2.data) {
            temp.next=head1;
            temp=temp.next;
            head1=head1.next;
        }
        else {
            temp.next=head2;
            temp=temp.next;
            head2=head2.next;
        }
    }
    if(head1!=null) {
        temp.next=head1;
    }
    else if(head2!=null)
        temp.next=head2;

    return ansHead;
}
private static final Scanner scanner = new Scanner(System.in);

public static void main(String[] args) throws IOException {
    BufferedWriter bufferedWriter = new BufferedWriter(new
    FileWriter(System.getenv("OUTPUT_PATH")));

    int tests = scanner.nextInt();
    scanner.skip("(\\r\\n|[\\n\\r\\u2028\\u2029\\u0085])?");

    for (int testsItr = 0; testsItr < tests; testsItr++) {
        SinglyLinkedList llist1 = new SinglyLinkedList();

        int llist1Count = scanner.nextInt();
        scanner.skip("(\\r\\n|[\\n\\r\\u2028\\u2029\\u0085])?");

        for (int i = 0; i < llist1Count; i++) {
            int llist1Item = scanner.nextInt();
            scanner.skip("(\\r\\n|[\\n\\r\\u2028\\u2029\\u0085])?");

            llist1.insertNode(llist1Item);
        }

        SinglyLinkedList llist2 = new SinglyLinkedList();

```

```

int llist2Count = scanner.nextInt();
scanner.skip("(\\r\\n|[\\n\\r\\u2028\\u2029\\u0085])?");

for (int i = 0; i < llist2Count; i++) {
    int llist2Item = scanner.nextInt();
    scanner.skip("(\\r\\n|[\\n\\r\\u2028\\u2029\\u0085])?");

    llist2.insertNode(llist2Item);
}

SinglyLinkedListNode llist3 = mergeLists(llist1.head, llist2.head);

printSinglyLinkedList(llist3, " ", bufferedWriter);
bufferedWriter.newLine();
}

bufferedWriter.close();

scanner.close();
}
}

```

Input (stdin)

```

1
3
1
2
3
2
3
4

```

Your Output (stdout)

```

1 2 3 3 4

```

Expected Output

```

1 2 3 3 4

```

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

```
class myList{
    Node head=null;
    class Node{
        int data;
        Node next;

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

    void insertAtbeginning(int new_data){
        Node new_node= new Node(new_data);
        new_node.next=head;
        head=new_node;
    }

    void insertAt(int new_data,int pos){
        Node temp=head;
        Node new_node= new Node(new_data);
        int count=1;
        if(pos==1){
            new_node.next=head;
            head=new_node;
        }
        else{
            while(count!=pos-1){
                temp = temp.next;
                count++;
            }
            new_node.next=temp.next;
            temp.next=new_node;
        }
    }

    void insertAtEnd(int new_data){
        Node new_node= new Node(new_data);
        Node temp=head;
```

```

        if(temp==null){
            head=new_node;
            return;
        }
        else{
            while(temp.next!=null){
                temp=temp.next;
            }
            temp.next=new_node;
        }
    }

void delete(int key){
    Node temp=head;
    Node prev=null;
    if(temp!=null && temp.data==key){
        head=temp.next;
        return;
    }
    else{
        while(temp!=null && temp.data!=key){
            prev=temp;
            temp=temp.next;
        }
        if(temp==null){
            return;
        }
        else{
            prev.next=temp.next;
        }
    }
}

void deleteAtPosition(int pos){
    if(head==null)
        return;
    Node temp = head;
    if(pos==0){
        head=temp.next;
        return;
    }
}

```

```

        for(int i=0; temp!=null && i<pos-1 ;i++){
            temp= temp.next;
        }
        if(temp==null || temp.next==null)
            return;
        temp.next=temp.next.next;
    }

```

```

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

```

```

int count(){
    Node temp= head;
    int count=0;
    while(temp!=null){
        count++;
        temp=temp.next;
    }
    return count;
}

```

```

public boolean search(int x){
    Node temp = head;
    if(temp.data==x)
        return true;
    while(temp!=null){
        if(temp.data == x){
            return true;
        }
        temp=temp.next;
    }
}

```



```

    }
    return false;
}

```

```

void reverse(){
    Node temp=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;
}

```

```

void middleElement(){
    Node temp1=head;
    Node temp2=head;
    while(temp2!=null && temp2.next!=null){
        temp1=temp1.next;
        temp2=temp2.next.next;
    }
    System.out.println("Middle element is "+temp1.data);
}

```

```

public static void main(String args[]){
    myList l1= new myList();
    System.out.println("Inserting 11 in empty list");
    l1.insertAtbeginning(11);
    l1.display();
    System.out.println("Inserting 10 at the start of list");
    l1.insertAtbeginning(10);
    l1.display();
    System.out.println("Inserting 9 at the start of list");
    l1.insertAtbeginning(9);
    l1.display();
}

```

```

        System.out.println("Inserting 55 at 2nd position");
        l1.insertAt(55,2);
        l1.display();
        l1.insertAt(58,2);
        System.out.println("Inserting 58 at 2nd position");
        l1.display();
        //System.out.println("Deleting 10");
        //l1.delete(10);
        //l1.display();

        //System.out.println("Inserting 100 at end");
        //l1.insertAtEnd(100);
        //l1.display();

        //System.out.println("Deleting at node3");
        //l1.deleteAtPosition(3);
        //l1.display();

        //System.out.println("Number of nodes are " +l1.count());
        //System.out.println("Finding node with data as 55 "+l1.search(55));
        //l1.reverse();
        //System.out.println("Reversing a LL");
        //l1.display();
        l1.middleElement();
    }
}

```

```
D:\ARUSA CDAC\ADS\Day3>javac myList.java
```

```
D:\ARUSA CDAC\ADS\Day3>java myList
```

```
Inserting 11 in empty list
```

```
11----->
```

```
Inserting 10 at the start of list
```

```
10----->11----->
```

```
Inserting 9 at the start of list
```

```
9----->10----->11----->
```

```
Inserting 55 at 2nd position
```

```
9----->55----->10----->11----->
```

```
Inserting 58 at 2nd position
```

```
9----->58----->55----->10----->11----->
```

```
Middle element is 55
```

```
D:\ARUSA CDAC\ADS\Day3>
```

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

If there exists a link between all the nodes and the last nodes next pointer is pointing to first node then we can say there exists a loop in linked list.

```
if(tail.next==head){
```

```
System.out.println("There exists a loop");}
```

```
class myList{
```

```
    Node head=null;
```

```
    class Node{
```

```
        int data;
```

```
Node next;
```

```
Node(int data){
```

```
    this.data=data;
```

```
    next=null;
```

```
}
```

```
}
```

```
void insertAtbeginning(int new_data){
```

```
    Node new_node= new Node(new_data);
```

```
    new_node.next=head;
```

```
    head=new_node;
```

```
}
```

```
void insertAt(int new_data,int pos){
```

```
    Node temp=head;
```

```
    Node new_node= new Node(new_data);
```

```
    int count=1;
```

```
    if(pos==1){
```

```
        new_node.next=head;
```

```
        head=new_node;
```

```
    }
```

```
    else{
```

```
        while(count!=pos-1){
```

```
            temp = temp.next;
```

```
            count++;
```

```
        }  
        new_node.next=temp.next;  
        temp.next=new_node;  
    }  
}
```

```
void insertAtEnd(int new_data){  
    Node new_node= new Node(new_data);  
    Node temp=head;  
    if(temp==null){  
        head=new_node;  
        return;  
    }  
    else{  
        while(temp.next!=null){  
            temp=temp.next;  
        }  
        temp.next=new_node;  
    }  
}
```

```
void delete(int key){  
    Node temp=head;
```

```

Node prev=null;

if(temp!=null && temp.data==key){

    head=temp.next;

    return;

}

else{

    while(temp!=null && temp.data!=key){

        prev=temp;

        temp=temp.next;

    }

    if(temp==null){

        return;

    }

    else{

        prev.next=temp.next;

    }

}

}

```

```

void deleteAtPosition(int pos){

    if(head==null)

        return;

    Node temp = head;

    if(pos==0){

```

```

        head=temp.next;

        return;
    }

    for(int i=0; temp!=null && i<pos-1 ;i++){

        temp= temp.next;
    }

    if(temp==null || temp.next==null)

        return;

    temp.next=temp.next.next;
}

```

```

void display(){

    Node temp=head;

    if(head==null)

        return;

    else{

        while(temp!=null)

        {

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

            temp=temp.next;

        }

        System.out.println();}

}

```

```
int count(){  
    Node temp= head;  
    int count=0;  
    while(temp!=null){  
        count++;  
        temp=temp.next;  
    }  
    return count;  
}
```

```
public boolean search(int x){  
    Node temp = head;  
    if(temp.data==x)  
        return true;  
    while(temp!=null){  
        if(temp.data == x){  
            return true;  
        }  
        temp=temp.next;  
    }  
    return false;  
}
```

```
void reverse(){
```



```

Node temp=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;

}

void middleElement(){

    Node temp1=head;

    Node temp2=head;

    while(temp2!=null && temp2.next!=null){

        temp1=temp1.next;

        temp2=temp2.next.next;

    }

    System.out.println("Middle element is "+temp1.data);

}

void isLoop(){

```

```
Node temp=head;
while(temp.next!=null){
    temp=temp.next;
}
temp.next=head;
if(temp.next==head){
    System.out.println("Loop exists");
}
}
```

```
public static void main(String args[]){
    myList l1= new myList();
    System.out.println("Inserting 11 in empty list");
    l1.insertAtbeginning(11);
    l1.display();
    System.out.println("Inserting 10 at the start of list");
    l1.insertAtbeginning(10);
    l1.display();
    System.out.println("Inserting 9 at the start of list");
    l1.insertAtbeginning(9);
    l1.display();
    System.out.println("Inserting 55 at 2nd position");
    l1.insertAt(55,2);
    l1.display();
}
```

```
l1.insertAt(58,2);

System.out.println("Inserting 58 at 2nd position");

l1.display();

//System.out.println("Deleting 10");

//l1.delete(10);

//l1.display();


//System.out.println("Inserting 100 at end");

//l1.insertAtEnd(100);

//l1.display();


//System.out.println("Deleting at node3");

//l1.deleteAtPosition(3);

//l1.display();


//System.out.println("Number of nodes are " +l1.count());

//System.out.println("Finding node with data as 55 "+l1.search(55));

//l1.reverse();

//System.out.println("Reversing a LL");

//l1.display();

l1.middleElement();

l1.isLoop();

}

}
```

```
D:\ARUSA CDAC\ADS\Day3>javac myList.java
```

```
D:\ARUSA CDAC\ADS\Day3>java myList
```

```
Inserting 11 in empty list
```

```
11----->
```

```
Inserting 10 at the start of list
```

```
10----->11----->
```

```
Inserting 9 at the start of list
```

```
9----->10----->11----->
```

```
Inserting 55 at 2nd position
```

```
9----->55----->10----->11----->
```

```
Inserting 58 at 2nd position
```

```
9----->58----->55----->10----->11----->
```

```
Middle element is 55
```

```
Loop exists
```

7.Find start node of loop in linkedlist

We can find the start of LinkedList by looking at the head pointer since head pointer indicates the start of linked list.

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

```
class myList{
```

```
    Node head=null;
```

```
    class Node{
```

```
        int data;
```

```
        Node next;
```

```
        Node(int data){
```

```
            this.data=data;
```

```
            next=null;
```

```
}  
  
}
```

```
void insertAtbeginning(int new_data){  
  
    Node new_node= new Node(new_data);  
  
    new_node.next=head;  
  
    head=new_node;  
  
}
```

```
void insertAt(int new_data,int pos){  
  
    Node temp=head;  
  
    Node new_node= new Node(new_data);  
  
    int count=1;  
  
    if(pos==1){  
  
        new_node.next=head;  
  
        head=new_node;  
  
    }  
  
    else{  
  
        while(count!=pos-1){  
  
            temp = temp.next;  
  
            count++;  
  
        }  
  
        new_node.next=temp.next;  
  
        temp.next=new_node;  
  
    }  
  
}
```

```
}
```

```
void insertAtEnd(int new_data){  
  
    Node new_node= new Node(new_data);  
  
    Node temp=head;  
  
    if(temp==null){  
  
        head=new_node;  
  
        return;  
  
    }  
  
    else{  
  
        while(temp.next!=null){  
  
            temp=temp.next;  
  
        }  
  
        temp.next=new_node;  
  
    }  
  
}
```

```
void delete(int key){  
  
    Node temp=head;  
  
    Node prev=null;  
  
    if(temp!=null && temp.data==key){  
  
        head=temp.next;  
  
        return;  
  
    }  
  
}
```

```

    }
    else{
        while(temp!=null && temp.data!=key){
            prev=temp;
            temp=temp.next;
        }
        if(temp==null){
            return;
        }
        else{
            prev.next=temp.next;
        }
    }
}

```

```

void deleteAtPosition(int pos){
    if(head==null)
        return;
    Node temp = head;
    if(pos==0){
        head=temp.next;
        return;
    }
    for(int i=0; temp!=null && i<pos-1 ;i++){
        temp= temp.next;
    }
}

```

```
    }  
    if(temp==null || temp.next==null)  
        return;  
    temp.next=temp.next.next;  
}
```

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

```
int count(){  
    Node temp= head;  
    int count=0;  
    while(temp!=null){  
        count++;  
    }  
}
```



```
        temp=temp.next;
    }
    return count;
}
```

```
public boolean search(int x){
    Node temp = head;
    if(temp.data==x)
        return true;
    while(temp!=null){
        if(temp.data == x){
            return true;
        }
        temp=temp.next;
    }
    return false;
}
```

```
void reverse(){
    Node temp=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;

}

```

```

void middleElement(){

    Node temp1=head;

    Node temp2=head;

    while(temp2!=null && temp2.next!=null){

        temp1=temp1.next;

        temp2=temp2.next.next;

    }

    System.out.println("Middle element is "+temp1.data);

}

```

```

/*void isLoop(){

    Node temp=head;

    while(temp.next!=null){

        temp=temp.next;

    }

    temp.next=head;
}

```

```

        if(temp.next==head){

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

        }

    }*/

```

```

int nthElementFromEnd(int pos){

    reverse();

    int count=1;

    Node temp= head;

    while(temp!=null){

        if(pos==count)

            return temp.data;

        else{

            count++;

            temp=temp.next;

        }

    }

    return -1;

}

```

```

public static void main(String args[]){

    myList l1= new myList();

    System.out.println("Inserting 11 in empty list");

    l1.insertAtbeginning(11);

    l1.display();

    System.out.println("Inserting 10 at the start of list");
}

```

```
l1.insertAtbeginning(10);

l1.display();

System.out.println("Inserting 9 at the start of list");

l1.insertAtbeginning(9);

l1.display();

System.out.println("Inserting 55 at 2nd position");

l1.insertAt(55,2);

l1.display();

l1.insertAt(58,2);

System.out.println("Inserting 58 at 2nd position");

l1.display();

//System.out.println("Deleting 10");

//l1.delete(10);

//l1.display();


//System.out.println("Inserting 100 at end");

//l1.insertAtEnd(100);

//l1.display();


//System.out.println("Deleting at node3");

//l1.deleteAtPosition(3);

//l1.display();


//System.out.println("Number of nodes are " +l1.count());

//System.out.println("Finding node with data as 55 "+l1.search(55));
```

```

        //l1.reverse();

        //System.out.println("Reversing a LL");

        //l1.display();

        l1.middleElement();

        //l1.isLoop();

        System.out.println("Printing element at 2nd position from end of linkedlist :
"+l1.nthElementFromEnd(2));

    }

}

```

```

D:\ARUSA CDAC\ADS\Day3>java myList
Inserting 11 in empty list
11----->
Inserting 10 at the start of list
10----->11----->
Inserting 9 at the start of list
9----->10----->11----->
Inserting 55 at 2nd position
9----->55----->10----->11----->
Inserting 58 at 2nd position
9----->58----->55----->10----->11----->
Middle element is 55
Printing element at 2nd position from end of linkedlist : 10

```

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

```

import java.util.*;

class linkedListPalindrome {
    public static void main(String args[])
    {
        Node one = new Node(1);
        Node two = new Node(2);
        Node three = new Node(3);
    }
}

```

```

Node four = new Node(4);
Node five = new Node(3);
Node six = new Node(2);
Node seven = new Node(1);
one.next = two;
two.next = three;
three.next = four;
four.next = five;
five.next = six;
six.next = seven;
boolean condition = isPalindrome(one);
System.out.println("isPalidrome :" + condition);
}
static boolean isPalindrome(Node head)
{

    Node slow = head;
    boolean ispalin = true;
    Stack<Integer> stack = new Stack<Integer>();

    while (slow != null) {
        stack.push(slow.data);
        slow = slow.next;
    }

    while (head != null) {

        int i = stack.pop();
        if (head.data == i) {
            ispalin = true;
        }
        else {
            ispalin = false;
            break;
        }
        head = head.next;
    }
    return ispalin;
}
}

```

```

class Node {

```

```

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

```

```

D:\ARUSA CDAC\ADS\Day3>java LinkedListPalindrome
Error: Could not find or load main class LinkedListPalindrome
Caused by: java.lang.NoClassDefFoundError: linkedListPalindrome (wrong name: LinkedListPalindrome)

D:\ARUSA CDAC\ADS\Day3>java linkedListPalindrome
isPalidrome :true

D:\ARUSA CDAC\ADS\Day3>

```

10.Add two numbers represented by linked list in java

```

import java.util.*;

class linkedListPalindrome {
    public static void main(String args[])
    {
        Node one = new Node(1);
        Node two = new Node(2);
        Node three = new Node(3);
        Node four = new Node(4);
        Node five = new Node(3);
        Node six = new Node(2);
        Node seven = new Node(1);
        one.next = two;
        two.next = three;
        three.next = four;
        four.next = five;
        five.next = six;
        six.next = seven;
        int sum=one.data+two.data;
        System.out.println("Sum of node 1 and node 2 is"+sum);
    }
}

```

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

```
D:\ARUSA CDAC\ADS\Day3>javac LinkedListPalindrome.java
```

```
D:\ARUSA CDAC\ADS\Day3>java linkedListPalindrome  
Sum of node 1 and node 2 is3
```

```
D:\ARUSA CDAC\ADS\Day3>
```