1.Implement singly linked list in java

```
class Q1{
  Node head;

  static class Node{
    int data;
    Node link;

    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[]){
  Q1 l1 = new Q1();
  l1.head = new Node(11); //head connected to first node
  Node second = new Node(22); // node created
  Node third = new Node(33);  // node created

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

  l1.display();
}
```

```
}
```

2.Implement Doubly linked list in java

```
class Q2{
  Node head;
  static class Node{
  int data;
  Node prev;
  Node next;

  Node(int d){
    data = d;
    next = null;
    prev = null;
  }
}
  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;
  }
  void display(Node n){
    Node p = null;
    while( n != null ){
      System.out.print( n.data+" "  );
    p = n;
    n = n.next;
    }
```

```
  }
  public static void main(String args []){
    Q2 d1 = new Q2();
    d1.insert(22);
    d1.insert(33);
    d1.insert(44);
    d1.insert(66);
    d1.display( d1.head );
  }
}
```

3.How to reverse a linked list in java

```
class Q3 {
  static Node head;

  static class Node {
      int data;
      Node next;

      Node(int d) {
          data = d;
          next = null;
      }
  }
  Node reverse(Node node) {
      Node prev = null;
      Node current = node;
      Node next = null;
      while (current != null) {
          next = current.next;
          current.next = prev;
          prev = current;
          current = next;
```

```
        }
        node = prev;
        return node;
    }
    void printList(Node node) {
        while (node != null) {
            System.out.print(node.data + " ");
            node = node.next;
        }
    }
    public static void main(String[] args) {
        Q3 list = new Q3();
        list.head = new Node(11);
        list.head.next = new Node(22);
        list.head.next.next = new Node(33);
        list.head.next.next.next = new Node(44);
    list.head.next.next.next.next = new Node(55);

        System.out.print("Reversed linked list :  ");
        list.printList(head);
    }
}
```

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

```
class merge {
    Node sortedMerge(Node headA, Node headB){
        Node dummyNode = new Node(0);
        Node tail = dummyNode;
        while (true) {
            if (headA == null) {
                tail.next = headB;
                break;
            }
```

```
            if (headB == null) {
                tail.next = headA;
                break;
            }
            if (headA.data <= headB.data) {
                tail.next = headA;
                headA = headA.next;
            }
            else {
                tail.next = headB;
                headB = headB.next;
            }
            tail = tail.next;
        }
        return dummyNode.next;
    }
}
class Q4 {
    Node head;
    public void addToTheLast(Node node){
        if (head == null) {
            head = node;
        }
        else {
            Node temp = head;
            while (temp.next != null)
                temp = temp.next;
            temp.next = node;
        }
    }
    void printList(){
        Node temp = head;
        while (temp != null) {
```

```
            System.out.print(temp.data + " ");
            temp = temp.next;
        }
        System.out.println();
    }
    public static void main(String args[]){
        Q4 llist1 = new Q4();
        Q4 llist2 = new Q4();
        llist1.addToTheLast(new Node(5));
        llist1.addToTheLast(new Node(25));
        llist1.addToTheLast(new Node(15));

        llist2.addToTheLast(new Node(17));
        llist2.addToTheLast(new Node(9));
        llist2.addToTheLast(new Node(86));
        llist1.head = new Gfg().sortedMerge(llist1.head, llist2.head);
        System.out.println("Merged : ");
        llist1.printList();
    }
}
```

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

```
class Q5 {
    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]));
    }
}
```

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

```
class Q6{
  static class Node {
      int data;
      Node next;
      int x;
      Node(int x){
          data = x;
```

```
        next = null;
          x = 0;
    }
}
static Node push(Node node, int new_data){
    Node new_node = new Node(new_data);
    new_node.next = node;
    node = new_node;
    return node;
}
static boolean detectLoop(Node root){
    while (root != null) {

        if (root.x == 1)
            return true;
        root.x = 1;

        root = root.next;
    }
    return false;
}
public static void main(String[] args){
    Node head = null;
    head = push(head, 20);
    head = push(head, 4);
    head = push(head, 15);
    head = push(head, 10);
    head.next.next.next.next = head;
    if (detectLoop(head))
        System.out.print("Loop Detected");
    else
        System.out.print("Loop Not Detected");
}
```

```
}
```

7.Find start node of loop in linkedlist

```
class Q7{
  static class Node {
    int key;
    Node next;
  }
  static Node newNode(int key){
    Node node = new Node();
    node.key = key;
    node.next = null;
    return node;
  }
  static void printList(Node head){
    while (head != null) {
    System.out.print(head.key + " ");
    head = head.next;
    }
    System.out.println();
  }
  static Node detectAndRemoveLoop(Node head){
    if (head == null || head.next == null)
    return null;
    Node slow = head, fast = head;
    slow = slow.next;
    fast = fast.next.next;
    while (fast != null &&
       fast.next != null) {
    if (slow == fast)
      break;
    slow = slow.next;
    fast = fast.next.next;
```

```
    }
    if (slow != fast)
    return null;
    slow = head;
    while (slow != fast){
    slow = slow.next;
    fast = fast.next;
    }
    return slow;
  }
  public static void main(String[] args){
    Node head = newNode(50);
    head.next = newNode(20);
    head.next.next = newNode(15);
    head.next.next.next = newNode(4);
    head.next.next.next.next = newNode(10);
    head.next.next.next.next.next = head.next.next;
    Node res = detectAndRemoveLoop(head);
    if (res == null)
    System.out.print("Null");
    else
    System.out.print("Starting Node : " +  res.key);
  }
}
```

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

```
import java.util.Scanner;
class Q8 {
    Node head;
  class Node {
    int data;
    Node next;
    Node(int d) {
```

```
        data = d;
        next = null;
    }
}
    public int GetNth(int pos){
        Node current = head;
        int count = 0;
        while (current != null)
        {
            if (count == pos)
                return current.data;
            count++;
            current = current.next;
        }
        assert (false);
        return 0;
    }
    public void insert(int new_data) {
        Node new_Node = new Node(new_data);
        new_Node.next = head;
        head = new_Node;
    }
    public static void main(String[] args) {
  Scanner sc = new Scanner(System.in);
        D3Que8 list = new D3Que8();
        list.insert(11);
        list.insert(22);
        list.insert(33);
        list.insert(44);
        list.insert(55);

    System.out.print(" Enter the position to be search : ");
    int n = sc.nextInt();
```

```
        System.out.println("Element at Position "+ n +" : " + list.GetNth(n));
    }
}
```

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

```
class Q9 {
  Node head;
  Node slow_ptr, fast_ptr, mid;
  class Node {
      char data;
      Node next;
      Node(char d){
          data = d;
          next = null;
      }
  }
  boolean isPalindrome(Node head) {
      slow_ptr = head;
      fast_ptr = head;
      Node prev_of_slow_ptr = head;
      Node midnode = null;
      boolean res = true;

      if (head != null && head.next != null) {
          while (fast_ptr != null
                 && fast_ptr.next != null) {
              fast_ptr = fast_ptr.next.next;

              prev_of_slow_ptr = slow_ptr;
              slow_ptr = slow_ptr.next;
          }
          if (fast_ptr != null) {
              midnode = slow_ptr;
```

```
            slow_ptr = slow_ptr.next;
        }
        mid = slow_ptr;
        prev_of_slow_ptr.next
            = null;
        reverse();
        res = compareLists(head, mid);
        reverse();
        if (midnode != null) {
            prev_of_slow_ptr.next = midnode;
            midnode.next = mid;
        }
        else
            prev_of_slow_ptr.next = mid;
    }
    return res;
}
void reverse(){
    Node prev = null;
    Node current = mid;
    Node next;
    while (current != null) {
        next = current.next;
        current.next = prev;
        prev = current;
        current = next;
    }
    mid = prev;
}
boolean compareLists(Node head1, Node head2){
    Node temp1 = head1;
    Node temp2 = head2;
    while (temp1 != null && temp2 != null) {
```

```
        if (temp1.data == temp2.data) {
            temp1 = temp1.next;
            temp2 = temp2.next;
        }
        else
            return false;
    }
    if (temp1 == null && temp2 == null)
        return true;
    return false;
}
public void push(char new_data){
    Node new_node = new Node(new_data);
    new_node.next = head;
    head = new_node;
}
void printList(Node ptr){
    while (ptr != null) {
        System.out.print(ptr.data + "->");
        ptr = ptr.next;
    }
    System.out.println("NULL");
}
public static void main(String[] args){
    D3Que9 llist = new D3Que9();
    char str[] = { 'a', 'b', 'c', 'e', 'c', 'b', 'a' };
    String string = new String(str);
    for (int i = 0; i < 7; i++) {
        llist.push(str[i]);
    }
    if (llist.isPalindrome(llist.head) != false) {
        System.out.println("Palindrome");
    }
```

```
      else {
          System.out.println("Not Palindrome");
      }
  }
}
```

10.Add two numbers represented by linked list in java

```
public class Q10{
  Node head;
  static class Node{
    int data;
    Node link;

    Node(int d){
      data = d;
      link = null;
  }
}
  public void insertEnd( int new_data ){
    Node new_node = new Node(new_data);
    if(head == null){
      head = new Node(new_data);
      return;
    }
    new_node.link = null;
    Node last = head;
    while( last.link != null )
      last = last.link;
      last.link = new_node;
      return;
  }
    public void display(){
  Node n = head;
```

```
  while( n != null ){
      System.out.print(n.data + " -> ");
          n = n.link;
  }
}
public static void main(String args[]){
  Q10 l1 = new Q10();
  l1.head = new Node(11);   //head connected to first node
  Node second = new Node(22);   // node created
  Node third = new Node(33);    // node created
  l1.head.link = second;    //link connection
  second.link = third;  // link connection
  l1.insertEnd( 44 );
  l1.insertEnd( 55 );
  l1.display();
}
}
```