/////////////////////BUBBLE SORT////////////

public class Bubblesort {

    public static void main(String[] args) {

        int[] arr = {64, 34, 25, 12, 22, 11, 90};

        System.out.println("Original Array:");

        printArray(arr);

        bubbleSort(arr);

        System.out.println("\nSorted Array:");

        printArray(arr);

        System.out.println("\nTime Complexity:");

        System.out.println("Best Case: O(n)");

        System.out.println("Worst Case: O(n^2)");

        System.out.println("Average Case: O(n^2)");

    }

    public static void bubbleSort(int[] arr) {

        int n = arr.length;

        for (int i = 0; i < n - 1; i++) {

            boolean swapped = false;

            for (int j = 0; j < n - i - 1; j++) {

                if (arr[j] > arr[j + 1]) {

                    int temp = arr[j];

                    arr[j] = arr[j + 1];

                    arr[j + 1] = temp;

                    swapped = true;

                }

            }

            if (!swapped) break;

        }

    }

    public static void printArray(int[] arr) {

        for (int num : arr) {

            System.out.print(num + " ");

        }

        System.out.println();

    }

}

/////////////////////MERGE SORT////////////////

public class MergeSort {

    public static void main(String[] args) {

        int[] arr = {64, 34, 25, 12, 22, 11, 90};

        System.out.println("Original Array:");

        printArray(arr);

        mergeSort(arr, 0, arr.length - 1);

        System.out.println("\nSorted Array:");

        printArray(arr);

        System.out.println("\nTime Complexity:");

        System.out.println("Best Case: O(n log n)");

        System.out.println("Worst Case: O(n log n)");

        System.out.println("Average Case: O(n log n)");

    }

    public static void mergeSort(int[] arr, int left, int right) {

        if (left < right) {

            int mid = left + (right - left) / 2;

            mergeSort(arr, left, mid);

            mergeSort(arr, mid + 1, right);

            merge(arr, left, mid, right);

        }

    }

    public static void merge(int[] arr, int left, int mid, int right) {

        int n1 = mid - left + 1;

        int n2 = right - mid;

        int[] leftArr = new int[n1];

        int[] rightArr = new int[n2];

        for (int i = 0; i < n1; i++) {

            leftArr[i] = arr[left + i];

        }

        for (int j = 0; j < n2; j++) {

            rightArr[j] = arr[mid + 1 + j];

        }

        int i = 0, j = 0;

        int k = left;

        while (i < n1 && j < n2) {

            if (leftArr[i] <= rightArr[j]) {

                arr[k] = leftArr[i];

                i++;

            } else {

                arr[k] = rightArr[j];

                j++;

            }

            k++;

        }

        while (i < n1) {

            arr[k] = leftArr[i];

            i++;

            k++;

        }

        while (j < n2) {

            arr[k] = rightArr[j];

            j++;

            k++;

        }

    }

    public static void printArray(int[] arr) {

        for (int num : arr) {

            System.out.print(num + " ");

        }

        System.out.println();

    }

}

class Node {

    int data;

    Node next;

    Node(int data) {

        this.data = data;

        this.next = null;

    }

}

/////////////////////LINKEDLIST FOR DEL INSER/////////////

public class Linkedlist {

    private Node head;

    public void insert(int data) {

        Node newNode = new Node(data);

        if (head == null) {

            head = newNode;

        } else {

            Node current = head;

            while (current.next != null) {

                current = current.next;

            }

            current.next = newNode;

        }

    }

    public void delete(int value) {

        if (head == null) {

            System.out.println("List is empty.");

            return;

        }

        if (head.data == value) {

            head = head.next;

            return;

        }

        Node current = head;

        while (current.next != null && current.next.data != value) {

            current = current.next;

        }

        if (current.next == null) {

            System.out.println("Value not found in the list.");

        } else {

            current.next = current.next.next;

        }

    }

    public void printList() {

        if (head == null) {

            System.out.println("List is empty.");

            return;

        }

        Node current = head;

        while (current != null) {

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

            current = current.next;

        }

        System.out.println();

    }

    public static void main(String[] args) {

        Linkedlist list = new Linkedlist();

        list.insert(10);

        list.insert(20);

        list.insert(30);

        list.insert(40);

        System.out.println("Linked List after insertion:");

        list.printList();

        list.delete(20);

        System.out.println("Linked List after deleting 20:");

        list.printList();

        list.delete(50);

        System.out.println("Linked List after trying to delete 50:");

        list.printList();

        list.delete(10);

        System.out.println("Linked List after deleting the head (10):");

        list.printList();

    }

}