```java
package com.jinglin;

import java.util.ArrayList;
import java.util.Scanner;
import java.io.*;

public class Main {

    static int[] countC = new int[3];
    static int[] countI = new int[3];

    public static void main(String[] args) throws IOException{
        Scanner sc = new Scanner(new File("c:/sortFile.txt"));
        while(sc.hasNext()) {
            ArrayList<Integer> list = new ArrayList<Integer>();
            String heading = readData(sc, list);
            printOriginalData(heading, list);
            for(int i = 0; i < countC.length; i++) countC[i] = 0;
            for(int i = 0; i < countI.length; i++) countI[i] = 0;

            ArrayList<Integer> listCopy = new ArrayList<Integer>(list);
            String sortType = bubbleSort(listCopy);
            printSortedData(sortType, listCopy);

            listCopy = new ArrayList<Integer>(list);
            sortType = quickSort(listCopy);
            printSortedData(sortType, listCopy);

            listCopy = new ArrayList<Integer>(list);
            sortType = mergeSort(listCopy);
            printSortedData(sortType, listCopy);

            comparison();
            interchange();
            System.out.println();
        }

    }

    public static String readData(Scanner sc, ArrayList<Integer> list){
        String heading = sc.nextLine();
        int num = sc.nextInt();
        while(num != -999){
            list.add(num);
            num = sc.nextInt();
        }
        if(sc.hasNext()) sc.nextLine();
        return heading;
    }

    public static void printOriginalData(String heading, ArrayList<Integer> list){
        System.out.println(heading);
        for(int n: list) System.out.print(n + " ");
        System.out.println();
    }

    public static void printSortedData(String sortType, ArrayList<Integer> list){
        System.out.println(sortType);
        for(int n: list) System.out.print(n + " ");
        System.out.println();
    }

    public static void comparison(){
        String big = "", middle = "", small = "";
        int max = 0, midium = 0, min = 0;
        max = Math.max(countC[0], Math.max(countC[1], countC[2]));
        min = Math.min(countC[0], Math.min(countC[1], countC[2]));
        for(int i = 0; i < countC.length; i++){
            if(max == countC[i]){
                big = whichSort(i);
                countC[i] = -1;
            }else if(min == countC[i]){
                small = whichSort(i);
                countC[i] = -1;
            }
        }
        for(int i = 0; i < countC.length; i++){
            if(countC[i] != -1){
                middle = whichSort(i);
                midium = countC[i];
            }
        }
        System.out.println(big + " used the most comparison: " + max);
        System.out.println(middle + " is in the middle: " + midium);
        System.out.println(small + " used the least comparison: " + min);
    }

    public static void interchange(){
        String big = "", middle = "", small = "";
        int max = 0, midium = 0, min = 0;
        if(countI[0] == 0 && countI[1] == 0 && countI[2] == 0){
            System.out.println("All 3 sorts use 0 interchange");
            return;
        }
        max = Math.max(countI[0], Math.max(countI[1], countI[2]));
        min = Math.min(countI[0], Math.min(countI[1], countI[2]));
        for(int i = 0; i < countI.length; i++){
            if(max == countI[i]){
                big = whichSort(i);
                countI[i] = -1;
            }else if(min == countI[i]){
                small = whichSort(i);
                countI[i] = -1;
            }
        }
        for(int i = 0; i < countI.length; i++){
            if(countI[i] != -1){
                middle = whichSort(i);
                midium = countI[i];
            }
        }
        System.out.println(big + " used the most interchange: " + max);
        System.out.println(middle + " is in the middle: " + midium);
        System.out.println(small + " used the least interchange: " + min);
    }

    public static String whichSort(int i){
        String sort;
        if(i == 0) sort = "Bubble sort";
        else if(i == 1) sort = "Quick sort";
        else sort = "Merge sort";
        return sort;
    }

    public static String bubbleSort(ArrayList<Integer> list){
        int unsortedIndex;
        for(unsortedIndex = list.size(); unsortedIndex > 1; unsortedIndex--){
            for(int i = 0; i < unsortedIndex - 1; i++){
```

```
            if(list.get(i) > list.get(i + 1)){
                int temp = list.get(i);
                list.set(i, list.get(i + 1));
                list.set(i + 1, temp);
                countI[0]++;
            }
            countC[0]++;
        }
    }
    return "Bubble sort:";
}

public static String quickSort(ArrayList<Integer> list){
    quickSort2(list, 0, list.size());
    return "Quick sort:";
}

public static void quickSort2(ArrayList<Integer> list, int start, int end){
    if(end - start <2) return;
    int pivotIndex = partition(list, start, end);
    quickSort2(list, start, pivotIndex);
    quickSort2(list, pivotIndex + 1, end);
}

public static int partition(ArrayList<Integer> list, int start, int end){
    int pivot = list.get(start);
    int i = start;
    int j = end;
    while(i < j){
        while(i < j && list.get(--j) >= pivot) countC[1]++;
        if(i < j){
            list.set(i, list.get(j));
            countI[1]++;
        }
        while(i < j && list.get(++i) <= pivot) countC[1]++;
        if(i < j){
            list.set(j, list.get(i));
            countI[1]++;
        }
    }
    list.set(j, pivot);
    return j;
}

public static String mergeSort(ArrayList<Integer> list){
    mergeSort2(list, 0, list.size());
    return "Merge sort:";
}

public static void mergeSort2(ArrayList<Integer> input, int start, int end) {
    if (end - start < 2) return;
    int mid = (start + end) / 2;
    mergeSort2(input, start, mid);
    mergeSort2(input, mid, end);
    merge(input, start, mid, end);
}

public static void merge(ArrayList<Integer> input, int start, int mid, int end) {
    if (input.get(mid - 1) <= input.get(mid)) return;
    int i = start;
    int j = mid;
    int tempIndex = 0;
    int[] temp = new int[end - start];
    while (i < mid && j < end) {
            temp[tempIndex++] = input.get(i) <= input.get(j) ?
                    input.get(i++) : input.get(j++);
            countC[2]++;
        }
        for(int k = i, m = start + tempIndex; k < mid; k++, m++){
            input.set(m, input.get(k));
            countI[2]++;
        }
        for(int k = 0, m = start; k < tempIndex; k++, m++){
            input.set(m, temp[k]);
            countI[2]++;
        }
    }
}
```