SELF PRACTICE

1. At least two greatest

Program :

import java.util.Scanner;  
import Sorting.MergeSort;  
public class AtleastTwo {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter the size of the array: ");  
 int size = sc.nextInt();  
 int[] arr = new int[size];  
 System.*out*.print("Enter the elements of the array: ");  
 for (int i = 0; i < size; i++) {  
 arr[i] = sc.nextInt();  
 }  
 MergeSort sort = new MergeSort();  
 sort.mergesort(arr, 0, arr.length - 1);  
 for (int i = 0; i < size-2; i++) {  
 System.*out*.print(arr[i] + " ");  
 }  
 }  
}

Output :

Enter the size of the array: 5

Enter the elements of the array: 2 8 7 1 5

1 2 5

1. Third Maximum Number

Program :

import java.util.Scanner;  
  
public class ThirdMax {  
 static int thirdMax(final int[] nums) {  
 long max1 = Long.*MIN\_VALUE*;  
 long max2 = Long.*MIN\_VALUE*;  
 long max3 = Long.*MIN\_VALUE*;  
  
 for(final int num : nums){  
 if(num > max1) {  
 max3 = max2;  
 max2 = max1;  
 max1 = num;  
 } else if(max1>num && num>max2) {  
 max3 = max2;  
 max2 = num;  
 } else if(max2>num && num>max3) {  
 max3=num;  
 }  
 }  
  
 return max3 != Long.*MIN\_VALUE* ? (int) max3 : (int) max1;  
 }  
  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 int n = sc.nextInt();  
 int[] nums = new int[n];  
 for(int i = 0; i < n; i++){  
 nums[i] = sc.nextInt();  
 }  
 System.*out*.println(*thirdMax*(nums));  
 }  
}

Output :

3

1 2 3

1

1. To check whether two arrays are equal

Program :

import java.util.Scanner;  
import Sorting.MergeSort;  
public class CheckEqual {  
 static int find(int[] arr1,int[] arr2)  
 {  
 if(arr1.length!=arr2.length)  
 {  
 return 0;  
 }  
 else  
 {  
 for (int i = 0; i < arr1.length; i++)  
 {  
 if (arr1[i]!=arr2[i])  
 return 0;  
 }  
 }  
 return 1;  
 }  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter the size of the array: ");  
 int size = sc.nextInt();  
 int[] arr = new int[size];  
 System.*out*.print("Enter the elements of the array: ");  
 for (int i = 0; i < size; i++) {  
 arr[i] = sc.nextInt();  
 }  
 MergeSort sort = new MergeSort();  
 System.*out*.print("Enter size of array2 : ");  
 int size2 = sc.nextInt();  
 int[] arr2 = new int[size2];  
 System.*out*.print("Enter the elements of the array2 : ");  
 for (int i = 0; i < size; i++) {  
 arr2[i] = sc.nextInt();  
 }  
 sort.mergesort(arr, 0, arr.length-1);  
 sort.mergesort(arr2, 0, arr2.length-1);  
 System.*out*.println(*find*(arr,arr2));  
 }  
}

Output :

Enter the size of the array: 3

Enter the elements of the array: 3 2 1

Enter size of array2 : 3

Enter the elements of the array2 : 3 2 1

1

1. Sort the People

Program :

import java.util.Arrays;  
import java.util.Scanner;  
import Sorting.MergeSort;  
public class Sort {  
 static void sort(int[] arr)  
 {  
 int n = arr.length;  
 boolean flag = false;  
 for (int i = 0; i < n; i++) {  
 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;  
 flag = true;  
 }  
 if(!flag)  
 break;  
 }  
 }  
 }  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 int n = sc.nextInt();  
 String[] arr = new String[n];  
 for (int i = 0; i < n; i++) {  
 arr[i] = sc.next();  
 }  
 int[] height = new int[n];  
 for (int i = 0; i < n; i++) {  
 height[i] = sc.nextInt();  
 }  
 *sort*(height);  
 for(int j = 0; j<n-1; j++){  
 for(int i = 0; i<n-1-j; i++){  
 if(height[i]<height[i+1]){  
 String t = arr[i+1];  
 arr[i+1] = arr[i];  
 arr[i] = t;  
 int tt = height[i+1];  
 height[i+1] = height[i];  
 height[i] = tt;  
 }  
 }  
 }  
 System.*out*.println(Arrays.*toString*(arr));  
 }  
}

Output :

3

Mary John Emma

180 165 170

[Mary, Emma, John]

1. Find Target Indices After Sorting Array

Program :

import java.util.Scanner;  
import Sorting.MergeSort;  
public class Target {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 int n = sc.nextInt();  
 int[] arr = new int[n];  
 for (int i = 0; i < n; i++) {  
 arr[i] = sc.nextInt();  
 }  
 int target = sc.nextInt();  
 MergeSort sort = new MergeSort();  
 sort.mergesort(arr, 0, arr.length - 1);  
  
 for (int i = 0; i < n; i++) {  
 if (arr[i] == target) {  
 System.*out*.print(i+" ");  
 }  
 }  
 }  
}

Output :

5

1 2 5 2 3

2

1 2

1. Squares of a Sorted Array

Program :

import java.util.Arrays;  
import java.util.Scanner;  
import Sorting.MergeSort;  
public class SquareAndSortArray {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter the size of the array: ");  
 int n = sc.nextInt();  
 int[] arr = new int[n];  
 System.*out*.print("Enter the elements of the array: ");  
 for (int i = 0; i < n; i++) {  
 arr[i] = sc.nextInt();  
 arr[i] \*= arr[i];  
 }  
 MergeSort sort = new MergeSort();  
 sort.mergesort(arr, 0, arr.length - 1);  
 System.*out*.print("Sorted array: "+ Arrays.*toString*(arr));  
  
 }  
}

Output :

Enter the size of the array: 5

Enter the elements of the array: -4 -1 0 3 10

Sorted array: [0, 1, 9, 16, 100]

1. Intersection Of Two Sorted Arrays

Program :

import java.util.HashSet;  
import java.util.Scanner;  
  
public class IntersectionArray {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter the size of the array: ");  
 int n = sc.nextInt();  
 int[] arr = new int[n];  
 System.*out*.print("Enter the elements of the array: ");  
 for (int i = 0; i < n; i++) {  
 arr[i] = sc.nextInt();  
 }  
 System.*out*.print("Enter the size of the array2: ");  
 int n2 = sc.nextInt();  
 int[] arr2 = new int[n2];  
 System.*out*.print("Enter the elements of the array2: ");  
 for (int i = 0; i < n2; i++) {  
 arr2[i] = sc.nextInt();  
 }  
  
 HashSet<Integer> set = new HashSet<>();  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n2; j++) {  
 if (arr[i] == arr2[j]) {  
 set.add(arr[i]);  
 break;  
 }  
 }  
 }  
  
 System.*out*.print("Intersection elements: ");  
 for (int num : set) {  
 System.*out*.print(num + " ");  
 }  
 }  
}

Output :

Enter the size of the array: 6

Enter the elements of the array: 1 2 2 2 3 4

Enter the size of the array2: 4

Enter the elements of the array2: 2 2 3 3

Intersection elements: 2 2 3

1. Assign the Assignments

Program :

import java.util.Arrays;  
import java.util.Scanner;  
  
public class AssignTheAssignments {  
 public static int possTasks(int n, int a, int b, int[] c) {  
 if (b + a > n) {  
 return 0;  
 }  
 Arrays.*sort*(c);  
 return c[c.length - a] - c[b - 1];  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 int t = scanner.nextInt();  
  
 for (int i = 0; i < t; i++) {  
 int n = scanner.nextInt();  
 int a = scanner.nextInt();  
 int b = scanner.nextInt();  
 int[] c = new int[n];  
  
 for (int j = 0; j < n; j++) {  
 c[j] = scanner.nextInt();  
 }  
  
 int result = 0;  
 for (int x = 1; x <= n; x++) {  
 if (*possTasks*(n, a, b, c) == x) {  
 result++;  
 }  
 }  
  
 System.*out*.println(result);  
 }  
  
 scanner.close();  
 }  
}

Output :

1

5 2 6

1 2 3 4 5

0

1. Height Checker

Program :

import java.util.Scanner;  
import Sorting.MergeSort;  
public class HeightChecker {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter the size of the array : ");  
 int n = sc.nextInt();  
 int[] arr = new int[n];  
 int[] arr2 = new int[n];  
 System.*out*.print("Enter the elements of the array : ");  
 for (int i = 0; i < n; i++) {  
 arr[i] = sc.nextInt();  
 arr2[i] = arr[i];  
 }  
  
 MergeSort sort = new MergeSort();  
 sort.mergesort(arr, 0, arr.length - 1);  
 int count = 0;  
 for (int i = 0; i < n; i++) {  
 if(arr[i] != arr2[i])  
 count++;  
 }  
 System.*out*.println(count);  
 }  
}

Output :

Enter the size of the array : 6

Enter the elements of the array : 1 1 4 2 1 3

3

1. Mean of Array After Removing Some Elements

Program :

import java.util.Arrays;  
import java.util.Scanner;  
  
public class MeanOfArray {  
 public static double trimMean(int[] arr) {  
 Arrays.*sort*(arr);  
 int n = arr.length;  
 int toRemove = (int) (n \* 0.05);  
 int sum = 0;  
 for (int i = toRemove; i < n - toRemove; i++) {  
 sum += arr[i];  
 }  
 return (double) sum / (n - 2 \* toRemove);  
 }  
  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter the size of the array: ");  
 int n = sc.nextInt();  
 int[] arr = new int[n];  
 System.*out*.print("Enter the elements of the array: ");  
 for (int i = 0; i < n; i++) {  
 arr[i] = sc.nextInt();  
 }  
 double result = *trimMean*(arr);  
 System.*out*.println("Mean of the array : " + result);  
 }  
}

Ouptut :

Enter the size of the array: 20

Enter the elements of the array: 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3

Mean of the array : 2.0

1. Sort Diagonally

Program :

import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.Collections;  
import java.util.Scanner;  
public class DiagonalSort {  
  
 static int[][] diagonalSort(int[][] mat) {  
 int arr[][]=new int[mat.length][mat[0].length];  
  
 for(int i=0;i<mat[0].length;i++){  
 ArrayList<Integer> list=new ArrayList<>();  
 int ind=0;  
 int ind1=i;  
 while(ind<mat.length&&ind1<mat[0].length){  
 list.add(mat[ind][ind1]);  
 ind++;  
 ind1++;  
 }  
 ind =0;  
 ind1=i;  
 Collections.*sort*(list);  
 while(ind<mat.length&&ind1<mat[0].length){  
 arr[ind][ind1]=list.get(ind);  
 ind++;  
 ind1++;  
 }  
 }  
 for(int i=0;i<mat.length;i++){  
 ArrayList<Integer>list=new ArrayList<>();  
 int ind1=i;  
 int ind=0;  
 while(ind1<mat.length&&ind<mat[0].length){  
 list.add(mat[ind1][ind]);  
 ind1++;  
 ind++;  
 }  
 Collections.*sort*(list);  
 ind1=i;  
 ind=0;  
 while(ind1<mat.length&&ind<mat[0].length){  
 arr[ind1][ind]=list.get(ind);  
 ind1++;  
 ind++;  
 }  
 }  
 return arr;  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 System.*out*.print("Enter the number of rows : ");  
 int m = scanner.nextInt();  
  
 System.*out*.print("Enter the number of columns: ");  
 int n = scanner.nextInt();  
  
 int[][] mat = new int[m][n];  
  
 System.*out*.println("Enter the matrix elements : ");  
 for (int i = 0; i < m; i++) {  
 for (int j = 0; j < n; j++) {  
 mat[i][j] = scanner.nextInt();  
 }  
 }  
 System.*out*.println(Arrays.*deepToString*(*diagonalSort*(mat)));  
  
 }  
}

Output :

Enter the number of rows : 3

Enter the number of columns: 4

Enter the matrix elements :

3 3 1 1

2 2 1 2

1 1 1 2

[[1, 1, 1, 1], [1, 2, 2, 2], [1, 2, 3, 3]]

1. Maximum area of square hole

Program :

import java.util.Arrays;  
import java.util.Scanner;  
  
public class MaxArea {  
 static int getMaxGap(int[] bars){  
 Arrays.*sort*(bars);  
 int count = 2, res = 2;  
 for(int i = 1; i < bars.length; ++i){  
 count = (bars[i-1] + 1 == bars[i])? count + 1: 2;  
 res = Math.*max*(res, count);  
 }  
 return res;  
 }  
 static int maxArea(int n, int m, int[] hBars, int[] vBars) {  
 int gap = Math.*min*(*getMaxGap*(hBars), *getMaxGap*(vBars));  
 return gap \* gap;  
 }  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 int n = sc.nextInt();  
 int m = sc.nextInt();  
 int[] h = new int[n];  
 int[] v = new int[n];  
 for (int i = 0; i < n; i++) {  
 h[i] = sc.nextInt();  
 }  
 for (int i = 0; i < m; i++) {  
 v[i] = sc.nextInt();  
 }  
 System.*out*.println(*maxArea*(n,m,h,v));  
 }  
}

Output :

2

1

2 3

2

4

1. Find minimum platforms needed to avoid delay in the train arrival

Program :

import java.util.Scanner;  
  
public class MinimumPlatform {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 int n = sc.nextInt();  
 float[] arr = new float[n];  
 float[] dep = new float[n];  
 for (int i = 0; i < n; i++) {  
 arr[i] = sc.nextFloat();  
 }  
 for (int i = 0; i < n; i++) {  
 dep[i] = sc.nextFloat();  
 }  
  
 int ans=1;  
 for(int i=0;i<=n-1;i++)  
 {  
 int count=1;  
 for(int j=i+1;j<=n-1;j++)  
 {  
 if((arr[i]>=arr[j] && arr[i]<=dep[j]) ||  
 (arr[j]>=arr[i] && arr[j]<=dep[i]))  
 {  
 count++;  
 }  
 }  
 ans=Math.*max*(ans,count);  
 }  
 System.*out*.println(ans);  
 }  
}

Output :

6

2.00 2.10 3.00 3.20 3.50 5.00

2.30 3.40 3.20 4.30 4.00 5.20

3

1. Sell Diminishing Bricks

Program :

import java.util.Scanner;  
  
public class MaxProfit {  
 static int maximumProfit(int[] arr, int l) {  
 int sum = 0;  
 while (l > 0) {  
 int maxIndex = 0;  
 int maxValue = Integer.*MIN\_VALUE*;  
 for (int i = 0; i < arr.length; i++) {  
 if (arr[i] > maxValue) {  
 maxValue = arr[i];  
 maxIndex = i;  
 }  
 }  
 sum += maxValue;  
 arr[maxIndex]--;  
 if (maxIndex > 0 && arr[maxIndex] < arr[maxIndex - 1]) {  
 int temp = arr[maxIndex];  
 arr[maxIndex] = arr[maxIndex - 1];  
 arr[maxIndex - 1] = temp;  
 }  
 l--;  
 }  
 return sum;  
 }  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 int n = sc.nextInt();  
 int l = sc.nextInt();  
 int[] arr = new int[n];  
 for (int i = 0; i < n; i++) {  
 arr[i] = sc.nextInt();  
 }  
 System.*out*.println(*maximumProfit*(arr,l));  
 }  
}

Output :

3 3

1 3 5

12

1. Connect Ropes

Program :

import java.util.PriorityQueue;  
import java.util.Scanner;  
import Sorting.MergeSort;  
public class ConnectRopes {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 int n = sc.nextInt();  
 int[] arr = new int[n];  
 for (int i = 0; i < n; i++) {  
 arr[i] = sc.nextInt();  
 }  
 PriorityQueue<Integer> list = new PriorityQueue<>();  
  
 for(int i=0; i<n; i++){  
 list.add(arr[i]);  
 }  
 long profit =0;  
  
 while(list.size()>=2){  
 int p1 = list.remove();  
 int p2 = list.remove();  
  
 list.add(p1+p2);  
 profit += p1+p2;  
 }  
 System.*out*.println(profit);  
  
 }  
}

Output :

4

4 3 2 6

29

1. Board Cutting

Program :

import java.util.\*;  
import java.io.\*;  
  
public class BoardCutting {  
 static final int *MOD* = 1000000007;  
  
 public static int boardCutting(ArrayList<Integer> horizontal, ArrayList<Integer> vertical, int m, int n) {  
 Collections.*sort*(horizontal, Collections.*reverseOrder*());  
 Collections.*sort*(vertical, Collections.*reverseOrder*());  
  
 long vp = 1;  
 long hp = 1;  
 int v = 0;  
 int h = 0;  
 long cost = 0;  
  
 while (h < horizontal.size() && v < vertical.size()) {  
 if (horizontal.get(h) >= vertical.get(v)) {  
 cost = (cost + (vp \* horizontal.get(h)) % *MOD*) % *MOD*;  
 h++;  
 hp++;  
 } else {  
 cost = (cost + (hp \* vertical.get(v)) % *MOD*) % *MOD*;  
 v++;  
 vp++;  
 }  
 }  
  
 while (h < horizontal.size()) {  
 cost = (cost + (vp \* horizontal.get(h)) % *MOD*) % *MOD*;  
 h++;  
 hp++;  
 }  
  
 while (v < vertical.size()) {  
 cost = (cost + (hp \* vertical.get(v)) % *MOD*) % *MOD*;  
 v++;  
 vp++;  
 }  
  
 return (int) cost;  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 int m = scanner.nextInt();  
 int n = scanner.nextInt();  
 ArrayList<Integer> horizontal = new ArrayList<>();  
 ArrayList<Integer> vertical = new ArrayList<>();  
  
 for (int i = 0; i < m - 1; i++) {  
 horizontal.add(scanner.nextInt());  
 }  
  
 for (int i = 0; i < n - 1; i++) {  
 vertical.add(scanner.nextInt());  
 }  
  
 System.*out*.println(*boardCutting*(horizontal, vertical, m, n));  
 scanner.close();  
 }  
}

Output :

2 2

5

2

9

1. Pythagorean Triplets

Program :

import java.util.Scanner;  
  
public class PythagoreanTriplet {  
 static boolean Triplet(int[] arr)  
 {  
 int n = arr.length;  
 for (int i = 0; i < n; i++) {  
 for (int j = i + 1; j < n; j++) {  
 for (int k = j + 1; k < n; k++) {  
 int x = arr[i] \* arr[i], y = arr[j] \* arr[j], z = arr[k] \* arr[k];  
  
 if (x == y + z || y == x + z || z == x + y)  
 return true;  
 }  
 }  
 }  
 return false;  
 }  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 int n = sc.nextInt();  
 int[] arr = new int[n];  
 for (int i = 0; i < n; i++) {  
 arr[i] = sc.nextInt();  
 }  
 System.*out*.println(*Triplet*(arr)?"YES":"NO");  
 }  
}

Output :

5

1 4 3 2 5

YES

1. Water Jug Problem

Program :

import java.util.Arrays;  
import java.util.Scanner;  
import Sorting.MergeSort;  
public class WaterJugProblem {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 int n = sc.nextInt();  
 int[] arr = new int[n];  
 int[] arr2 = new int[n];  
 for (int i = 0; i < n; i++) {  
 arr[i] = sc.nextInt();  
 }  
 for (int i = 0; i < n; i++) {  
 arr2[i] = sc.nextInt();  
 }  
 MergeSort sort = new MergeSort();  
 sort.mergesort(arr, 0, arr.length - 1);  
 sort.mergesort(arr2, 0, arr2.length - 1);  
  
 System.*out*.println(Arrays.*toString*(arr));  
 System.*out*.println(Arrays.*toString*(arr2));  
  
 }  
}

Output :

5

6 3 2 8 7

8 6 7 2 3

[2, 3, 6, 7, 8]

[2, 3, 6, 7, 8]

1. The Latest Time to Catch a Bus

Program :

import java.util.Arrays;  
import java.util.HashSet;  
import java.util.Scanner;  
  
public class Bus {  
 static int CatchTheBus(int[] buses, int[] passengers, int capacity) {  
 Arrays.*sort*(buses);  
 Arrays.*sort*(passengers);  
  
 if(passengers[0] > buses[buses.length - 1]) return buses[buses.length - 1];  
 int ans = passengers[0] - 1, j = 0;  
  
 for(int i = 0; i < buses.length; i++) {  
 int space = capacity;  
 while(j < passengers.length && space > 0 && passengers[j] <= buses[i]) {  
 if(j > 0 && passengers[j] - passengers[j-1] != 1) ans = passengers[j] - 1;  
 j++;  
 space--;  
 }  
  
 if(j > 0 && space > 0 && buses[i] != passengers[j-1]) ans = buses[i];  
 }  
  
 return ans;  
 }  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 int n = sc.nextInt();  
 int pass = sc.nextInt();  
 int capacity = sc.nextInt();  
 int[] bus = new int[n];  
 for (int i = 0; i < n; i++) {  
 bus[i] = sc.nextInt();  
 }  
 int[] passenger = new int[pass];  
 for (int i = 0; i < pass; i++) {  
 passenger[i] = sc.nextInt();  
 }  
 System.*out*.println(*CatchTheBus*(bus, passenger, capacity));  
 }  
}

Output :

2

4

2

10 20

2 17 18 19

16

1. Apartments

Program :

import java.util.\*;  
  
public class Apartments {  
 public static int maxApartmentsSold(int[] apartmentSize, int[] desiredSize, int k, int m, int n) {  
 Arrays.*sort*(apartmentSize);  
 Arrays.*sort*(desiredSize);  
 int i = 0, j = 0, count = 0;  
 while (i < m && j < n) {  
 if (Math.*abs*(apartmentSize[i] - desiredSize[j]) <= k) {  
 count++;  
 i++;  
 j++;  
 } else if (apartmentSize[i] < desiredSize[j]) {  
 i++;  
 } else {  
 j++;  
 }  
 }  
 return count;  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 int m = scanner.nextInt();  
 int[] apartmentSize = new int[m];  
 for (int i = 0; i < m; i++) {  
 apartmentSize[i] = scanner.nextInt();  
 }  
 int n = scanner.nextInt();  
 int[] desiredSize = new int[n];  
 for (int i = 0; i < n; i++) {  
 desiredSize[i] = scanner.nextInt();  
 }  
 int k = scanner.nextInt();  
 System.*out*.println(*maxApartmentsSold*(apartmentSize, desiredSize, k, m, n));  
 scanner.close();  
 }  
}

Output :

5

2 40 2 50 2

3

35 50 30

10

2

1. Sort an Array

Program :

import java.util.Arrays;  
import java.util.Scanner;  
import Sorting.MergeSort;  
public class SortAnArray {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 int n = sc.nextInt();  
 int[] arr = new int[n];  
 for (int i = 0; i < n; i++) {  
 arr[i] = sc.nextInt();  
 }  
 MergeSort sort = new MergeSort();  
 sort.mergesort(arr, 0, arr.length - 1);  
 System.*out*.println(Arrays.*toString*(arr));  
 }  
}

Output :

5

5 4 3 2 1

[1, 2, 3, 4, 5]

1. The k Strongest Values in an Array

Program :

import Sorting.MergeSort;  
  
import java.util.Arrays;  
import java.util.Scanner;  
  
public class Strongest {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 int n = sc.nextInt();  
 int[] arr = new int[n];  
 for (int i = 0; i < n; i++) {  
 arr[i] = sc.nextInt();  
 }  
 int k = sc.nextInt();  
 int[] strongest = new int[k];  
 MergeSort sort = new MergeSort();  
 int left = 0, right = arr.length -1;  
 int median = arr[(arr.length - 1) / 2], insertIndex = 0;  
 while (insertIndex < k) {  
 if (arr[right] - median >= median - arr[left]) {  
 strongest[insertIndex++] = arr[right--];  
 } else {  
 strongest[insertIndex++] = arr[left++];  
 }  
 }  
 System.*out*.println(Arrays.*toString*(strongest));  
 }  
}

Output :

5

1 2 3 4 5

2

[5, 1]

1. Rearrange Words in a Sentence

Program :

import java.util.Arrays;  
import java.util.Scanner;  
  
public class RearrangeWords {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 int n = sc.nextInt();  
 String[] arr = new String[n];  
 for (int i = 0; i < n; i++) {  
 arr[i] = sc.next();  
 }  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n-i-1; j++) {  
 if(arr[j].compareTo(arr[j + 1]) < 0)  
 {  
 String temp = arr[j];  
 arr[j] = arr[j+1];  
 arr[j+1] = temp;  
 }  
 }  
 }  
 System.*out*.println(Arrays.*toString*(arr));  
 }  
}

Output :

3

Leetcode is cool

[is, cool, Leetcode]

1. Largest Number formed from an Array

Program :

import java.util.Arrays;  
import java.util.Scanner;  
  
public class Largest {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 int n = sc.nextInt();  
 String[] arr = new String[n];  
 for (int i = 0; i < n; i++) {  
 arr[i] = sc.next();  
 }  
 Arrays.*sort*(arr,(a, b) -> (b+a).compareTo(a+b));  
 StringBuilder str = new StringBuilder();  
 for(int i=0; i<n; i++) {  
 str.append(arr[i]);  
 }  
 System.*out*.println(str);  
 }  
}

Output :

5

3 30 34 5 9

9534330

1. Arithmetic Subarrays

Program :

import java.util.\*;  
  
public class ArithmeticSubarrays {  
 static List<Boolean> check(int[] nums, int[] l, int[] r) {  
 List<Boolean> result = new ArrayList<>();  
 for (int i = 0; i < l.length; i++) {  
 int[] subarray = new int[r[i] - l[i] + 1];  
 for (int j = l[i]; j <= r[i]; j++) {  
 subarray[j - l[i]] = nums[j];  
 }  
 Arrays.*sort*(subarray);  
 boolean isArithmetic = true;  
 int diff = subarray[1] - subarray[0];  
 for (int j = 2; j < subarray.length; j++) {  
 if (subarray[j] - subarray[j - 1] != diff) {  
 isArithmetic = false;  
 break;  
 }  
 }  
 result.add(isArithmetic);  
 }  
 return result;  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 int n = scanner.nextInt();  
 int[] nums = new int[n];  
 for (int i = 0; i < n; i++) {  
 nums[i] = scanner.nextInt();  
 }  
 int q = scanner.nextInt();  
 int[] l = new int[q];  
 int[] r = new int[q];  
 for (int i = 0; i < q; i++) {  
 l[i] = scanner.nextInt();  
 }  
 for (int i = 0; i < q; i++) {  
 r[i] = scanner.nextInt();  
 }  
 List<Boolean> result = *check*(nums, l, r);  
 for (boolean b : result) {  
 System.*out*.print(b+" ");  
 }  
 }  
}

Output :

6

4 6 5 9 3 7

3

0 0 2

2 3 5

true false true

1. 3Sum

Program :

import java.util.Scanner;  
  
public class ThreeSum {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 int n = sc.nextInt();  
 int[] arr = new int[n];  
 for (int i = 0; i < n; i++) {  
 arr[i] = sc.nextInt();  
 }  
 for (int i = 0; i < n; i++) {  
 for (int j = i + 1; j < n; j++) {  
 for (int k = j + 1; k < n; k++) {  
 if(i != j && j != k && k != i){  
 if(arr[i] + arr[j] + arr[k] == 0)  
 System.*out*.println(arr[i] + " " + arr[j] + " " + arr[k]);  
 }  
 }  
 }  
 }  
 }  
}

Output :

3

0 0 0

0 0 0

1. Count Ways to Group Overlapping Ranges

Program :

import java.util.\*;  
  
public class OverlappingRanges {  
 public static int countGroups(int[][] ranges) {  
 Arrays.*sort*(ranges, (a, b) -> a[0] - b[0]);  
 int n = ranges.length;  
 int[] dp = new int[n];  
 Arrays.*fill*(dp, 1);  
 int count = 1;  
 for (int i = 1; i < n; i++) {  
 for (int j = 0; j < i; j++) {  
 if (ranges[i][0] >= ranges[j][0] && ranges[i][1] <= ranges[j][1]) {  
 dp[i] = Math.*max*(dp[i], dp[j] + 1);  
 } else if (ranges[i][0] >= ranges[j][0] && ranges[i][0] <= ranges[j][1]) {  
 dp[i] = Math.*max*(dp[i], dp[j] + 1);  
 }  
 }  
 if (dp[i] > count) {  
 count = dp[i];  
 }  
 }  
 return count;  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 int n = scanner.nextInt();  
 int[][] ranges = new int[n][2];  
 for (int i = 0; i < n; i++) {  
 ranges[i][0] = scanner.nextInt();  
 ranges[i][1] = scanner.nextInt();  
 }  
 System.*out*.println(*countGroups*(ranges));  
 scanner.close();  
 }  
}

Output :

2

6 10

5 15

2

1. Sell Diminishing-Valued Colored Balls

Program :

import java.util.\*;  
  
public class ColoredBalls {  
 private static final int *MOD* = 1000000007;  
  
 public static int maxProfit(int[] inventory, int orders) {  
 Arrays.*sort*(inventory);  
 int n = inventory.length;  
 long profit = 0;  
 int i = n - 1;  
 while (orders > 0) {  
 if (i >= 0 && inventory[i] > (i > 0 ? inventory[i - 1] : 0)) {  
 int diff = inventory[i] - (i > 0 ? inventory[i - 1] : 0);  
 int count = Math.*min*(diff \* (i + 1), orders);  
 profit = (profit + (long) (inventory[i] + inventory[i] - count + 1) \* count / 2) % *MOD*;  
 orders -= count;  
 }  
 i--;  
 }  
 return (int) profit;  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 int n = scanner.nextInt();  
 int[] inventory = new int[n];  
 for (int i = 0; i < n; i++) {  
 inventory[i] = scanner.nextInt();  
 }  
 int orders = scanner.nextInt();  
 System.*out*.println(*maxProfit*(inventory, orders));  
 scanner.close();  
 }  
}

Output :

2

3 5

6

19

1. Attack On titans

Program :

import java.util.\*;  
  
public class AttackOnTitans {  
 public static int penalty(int length, int breadth, int n, int[][] coordinates) {  
 List<Integer> x = new ArrayList<>();  
 List<Integer> y = new ArrayList<>();  
 for (int[] coordinate : coordinates) {  
 x.add(coordinate[0]);  
 y.add(coordinate[1]);  
 }  
 x.add(0);  
 y.add(0);  
 x.add(length + 1);  
 y.add(breadth + 1);  
 Collections.*sort*(x);  
 Collections.*sort*(y);  
 int maxX = 0;  
 int maxY = 0;  
 for (int i = 1; i < x.size(); i++) {  
 maxX = Math.*max*(x.get(i) - x.get(i - 1), maxX);  
 maxY = Math.*max*(y.get(i) - y.get(i - 1), maxY);  
 }  
 return (maxX - 1) \* (maxY - 1);  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 int length = scanner.nextInt();  
 int breadth = scanner.nextInt();  
 int n = scanner.nextInt();  
 int[][] coordinates = new int[n][2];  
 for (int i = 0; i < n; i++) {  
 coordinates[i][0] = scanner.nextInt();  
 coordinates[i][1] = scanner.nextInt();  
 }  
 System.*out*.println(*penalty*(length, breadth, n, coordinates));  
 scanner.close();  
 }  
}

Output :

6

2

2 5

5 2

4

1. Restaurant Customers

Program :

import java.util.Scanner;  
  
public class RestaurantCustomers {  
 static int findMaxCustomers(int[] arrival, int[] leaving) {  
 int maxCustomers = 0;  
 int currentCustomers = 0;  
 int i = 0, j = 0;  
 while (i < arrival.length && j < leaving.length) {  
 if (arrival[i] < leaving[j]) {  
 currentCustomers++;  
 i++;  
 } else {  
 currentCustomers--;  
 j++;  
 }  
 maxCustomers = Math.*max*(maxCustomers, currentCustomers);  
 }  
 return maxCustomers;  
 }  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 int N = scanner.nextInt();  
 int[] arrival = new int[N];  
 int[] leaving = new int[N];  
 for (int i = 0; i < N; i++) {  
 arrival[i] = scanner.nextInt();  
 leaving[i] = scanner.nextInt();  
 }  
 int maxCustomers = *findMaxCustomers*(arrival, leaving);  
  
 System.*out*.println(maxCustomers);  
 }  
  
  
}

Output :

2

1 3

4 5

1