SDE - MODULE 2 - PROBLEM SOLVING 8 - HANDSON  
  
1.Given the array nums, for each nums[i] find out how many numbers in the array are smaller than it. That is, for each nums[i] you have to count the number of valid j's such that j != i and nums[j] < nums[i]. Return the answer in an array.

Program : (SmallerThanCurrent)  
**package** com.handson;

**import** java.util.Scanner;

**public** **class** SmallerThanCurrent {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.print("Enter number of element: ");

**int** n = sc.nextInt();

System.***out***.println("Enter array :");

**int** a[]=**new** **int**[n];

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

a[i]=sc.nextInt();

}

**int** small[]=**new** **int**[n];

**int** x=0;

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

**int** p=*partition*(a,0,n-1);

small[i]=p;

x++;

**if**(x<n) {

**int** temp=a[0];

a[0]=a[x];

a[x]=temp;

}

}

System.***out***.print("output: ");

**for**(**int** val:small) {

System.***out***.print(val+" ");

}

System.***out***.println();

}

**public** **static** **int** partition(**int** a[], **int** s, **int** e) {

**int** x[]=a.clone();

**int** pivot = x[s];

**int** l = s + 1;

**int** r = e;

**while**(l <= r) {

**while**(l <= e && x[l] <= pivot) {

l++;}

**while**(r >= s && x[r] > pivot) {

r--;}

**if**(l < r) {

**int** temp = x[l];

x[l] = x[r];

x[r] = temp;

}

}

x[s] = x[r];

x[r] = pivot;

System.***out***.print("after partition: ");

**for**(**int** val:x) {

System.***out***.print(val+" ");

}

System.***out***.println();

**return** r;

}

}

Output :   
Enter number of element: 4

Enter array :

6 5 4 8

after partition: 4 5 6 8

after partition: 4 5 6 8

after partition: 4 6 5 8

after partition: 4 6 5 8

output: 2 1 0 3

2.Given an integer array nums that does not contain any zeros, find the largest positive integer k such that -k also exists in the array. Return the positive integer k. If there is no such integer, return -1.

Program : (PositiveInteger)  
**package** com.handson;

**import** java.util.Scanner;

**public** **class** PositiveInteger {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter n:");

**int** n = sc.nextInt();

**int** a[] = **new** **int**[n];

System.***out***.println("Enter array:");

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

a[i] = sc.nextInt();

}

**int** temp;

**boolean** flag = **false**;

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

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

**if** (a[j] > a[j + 1]) {

temp = a[j];

a[j] = a[j + 1];

a[j + 1] = temp;

}

}

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

**if**(a[n- i -1] + a[k] == 0) {

System.***out***.println(a[n- i -1]);

i = n;

flag = **true**;

}

}

}

**if**(flag == **false**) {

System.***out***.println("-1");

}

}

}

Output :   
Enter n:

4

Enter array:

-1 2 -3 3

3

3.Given a 0-indexed integer array nums of length n and an integer target, return the number of pairs (i, j) where 0 <= i < j < n and nums[i] + nums[j] < target.  
  
Program : (CountPairs)  
**package** com.handson;

**import** java.util.Scanner;

**public** **class** CountPairs {

**public** **static** **void** main(String [] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter number of element: ");

**int** n = sc.nextInt();

System.***out***.println("Enter array:");

**int** a[]=**new** **int**[n];

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

a[i]=sc.nextInt();

}

System.***out***.println("Enter the target:");

**int** t = sc.nextInt();

**int** temp,count = 0;

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

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

/\*if (a[j] > a[j + 1]) {

temp = a[j];

a[j] = a[j + 1];

a[j + 1] = temp;

}\*/

**if**(a[i] + a[j] < t) {

count++;

}

}

}

System.***out***.println(count);

}

}

Output :   
Enter number of element:

7

Enter array:

-6 2 5 -2 -7 -1 3

Enter the target:

-2

10

4.A sentence is a list of words that are separated by a single space with no leading or trailing spaces. Each word consists of lowercase and uppercase English letters. A sentence can be shuffled by appending the 1-indexed word position to each word then rearranging the words in the sentence.

Program : (SortingSentence)  
**package** com.handson;

**import** java.util.Scanner;

**public** **class** SortingSentence {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter n:");

**int** n = sc.nextInt();

sc.nextLine();

String a[] = **new** String[n];

System.***out***.println("Enter array:");

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

a[i] = sc.nextLine();

}

String temp;

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

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

**if** (a[j].charAt(a[j].length()-1) > (a[j + 1]).charAt(a[j+1].length()-1)) {

temp = a[j];

a[j] = a[j + 1];

a[j + 1] = temp;

}

}

}

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

System.***out***.print(a[i].substring(0,a[i].length()-1) + " ");

}

}

}

Output :   
Enter n:

4

Enter array:

myself2

me1

i4

and3

me myself and i

5.You are given two strings s and t. String t is generated by random shuffling string s and then add one more letter at a random position. Return the letter that was added to t.  
  
Program : (FindingDifference)  
**package** com.handson;

**import** java.util.\*;

**public** **class** FindingDifference {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter the string 1:");

String s1 = sc.nextLine();

System.***out***.println("Enter the string 2 : ");

String s2 = sc.nextLine();

**char** c1[] = s1.toCharArray();

**char** c2[] = s2.toCharArray();

**char** temp,temp2;

**for** (**int** i = 0; i < c1.length - 1; i++) {

**for** (**int** j = 0; j < c1.length - i - 1; j++) {

**if** (c1[j] > c1[j + 1]) {

temp = c1[j];

c1[j] = c1[j + 1];

c1[j + 1] = temp;

}

}

}

**for** (**int** i = 0; i < c2.length - 1; i++) {

**for** (**int** j = 0; j < c2.length - i - 1; j++) {

**if** (c2[j] > c2[j + 1]) {

temp2 = c2[j];

c2[j] = c2[j + 1];

c2[j + 1] = temp2;

}

}

}

**int** l = Math.*min*(c1.length, c2.length);

**boolean** flag = **false**;

**for**(**int** i = 0; i < l; i++) {

**if**(c1[i] != c2[i]) {

System.***out***.println(c2[i]);

flag = **true**;

**break**;

}

}

**if**(!flag) {

**if** (c1.length < c2.length) {

System.***out***.println(c2[c1.length]);

}

**else** **if** (c1.length > c2.length) {

System.***out***.println(c1[c2.length]);

}

}

}

}

Output :   
Enter the string 1:

cabd

Enter the string 2 :

acdbe

e

7.You have been given an unsorted array ‘ARR’. Your task is to sort the array in such a way that the array looks like a wave array.  
  
Program : (WaveForm)  
**package** com.handson;

**import** java.util.Scanner;

**public** **class** WaveForm {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter number of element: ");

**int** n = sc.nextInt();

System.***out***.println("Enter array:");

**int** a[]=**new** **int**[n];

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

a[i]=sc.nextInt();

}

**int** temp,temp2;

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

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

**if** (a[j] > a[j + 1]) {

temp = a[j];

a[j] = a[j + 1];

a[j + 1] = temp;

}

}

}

**for** (**int** i = 0; i < n - 1; i += 2) {

temp2 = a[i];

a[i] = a[i + 1];

a[i + 1] = temp2;

}

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

System.***out***.print(a[i] + " ");

}

}

}

Output :   
Enter number of element:

6

Enter array:

4 3 2 5 4 1

2 1 4 3 5 4

8. You are given an integer array arr. Sort the integers in the array in ascending order by the number of 1's in their binary representation and in case of two or more integers have the same number of 1's you have to sort them in ascending order. Return the array after sorting it.  
  
Program : (SortByOnes)  
**package** com.handson;

**import** java.util.Scanner;

**public** **class** SortByOnes {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter number of element: ");

**int** n = sc.nextInt();

System.***out***.println("Enter array:");

**int** a[]=**new** **int**[n];

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

a[i]=sc.nextInt();

}

**int** f[] = **new** **int**[n];

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

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

**if**(a[j] > a[j+1]) {

**int** temp = a[j];

a[j] = a[j+1];

a[j+1] = temp;

}

}

}

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

f[i] = Integer.*bitCount*(a[i]);

}

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

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

**if**(f[j] > f[j+1]) {

**int** temp = f[j];

f[j] = f[j+1];

f[j+1] = temp;

**int** temp2 = a[j];

a[j] = a[j+1];

a[j+1] = temp2;

}

}

}

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

System.***out***.print(a[i] + " ");

}

}

}

Output :   
Enter number of element:

9

Enter array:

0 1 2 3 4 5 6 7 8

0 1 2 4 8 3 5 6 7

9.Given an integer array nums, move all the even integers at the beginning of the array followed by all the odd integers. Return any array that satisfies this condition.  
  
Program : (SortParity)  
**package** com.handson;

**import** java.util.Scanner;

**public** **class** SortParity {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter number of element: ");

**int** n = sc.nextInt();

System.***out***.println("Enter array:");

**int** a[]=**new** **int**[n];

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

a[i]=sc.nextInt();

}

**int** l = 0, r = n-1;

**while**(l < r) {

**if**(a[l] % 2 == 0) {

l++;

}

**else** {

**int** temp = a[l];

a[l] = a[r];

a[r] = temp;

r--;

}

}

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

System.***out***.print(a[i] + " ");

}

}

}

Output :   
Enter number of element:

4

Enter array:

5 4 3 8

8 4 3 5

10.Given two different arrays A and B of size N and M, the task is to merge the two arrays which are unsorted into another array which is sorted.  
  
Program : (MergeArray)  
**package** com.handson;

**import** java.util.Scanner;

**public** **class** MergeArray {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter n: ");

**int** n = sc.nextInt();

System.***out***.println("Enter m : ");

**int** m = sc.nextInt();

System.***out***.println("Enter array 1:");

**int** a[]=**new** **int**[n];

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

a[i]=sc.nextInt();

}

System.***out***.println("Enter array 2:");

**int** b[]=**new** **int**[m];

**for**(**int** i=0;i<m;i++) {

b[i]=sc.nextInt();

}

**int** c[] = **new** **int**[m+n];

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

c[i] = a[i];

}

**for** (**int** i = 0; i < m; i++) {

c[n + i] = b[i];

}

**int** temp;

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

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

**if** (c[j] > c[j + 1]) {

temp = c[j];

c[j] = c[j + 1];

c[j + 1] = temp;

}

}

}

**for**(**int** i = 0; i < m+n; i++) {

System.***out***.print(c[i] + " ");

}

}

}

Output :   
Enter n:

4

Enter m :

3

Enter array 1:

1 10 5 15

Enter array 2:

20 0 2

0 1 2 5 10 15 20

11.A prefix nums[0..i] is sequential if, for all 1 <= j <= i, nums[j] = nums[j - 1] + 1. In particular, the prefix consisting only of nums[0] is sequential. Return the smallest integer x missing from nums such that x is greater than or equal to the sum of the longest sequential prefix.  
  
Program : (MisingInteger)  
**package** com.handson;

**import** java.util.Scanner;

**public** **class** MissingInteger {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter n:");

**int** n = sc.nextInt();

**int** a[] = **new** **int**[n];

System.***out***.println("Enter array:");

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

a[i] = sc.nextInt();

}

**int** sum = a[0];

**for**(**int** i = 1; i < n; i++)

{

**if**(a[i] == a[i-1]+1)

sum = sum + a[i];

**else**

**break**;

}

System.***out***.println(sum);

}

}

Output :   
Enter n:

5

Enter array:

1 2 3 2 5

6

12.You are given two integer arrays of equal length target and arr. In one step, you can select any non-empty subarray of arr and reverse it. You are allowed to make any number of steps. Return true if you can make arr equal to target or false otherwise.  
  
Program : (ReverseArray)  
**package** com.handson;

**import** java.util.Scanner;

**public** **class** ReverseArray {

**public** **static** **void** sort(**int** a[], **int** s, **int** e) {

**if** (s < e) {

**int** p = *partition*(a, s, e);

*sort*(a, s, p - 1);

*sort*(a, p + 1, e);

}

}

**public** **static** **int** partition(**int** a[], **int** s, **int** e) {

**int** p = a[s];

**int** l = s + 1;

**int** r = e;

**while** (l <= r) {

**while** (l <= e && a[l] <= p) {

l++;

}

**while** (r >= s && a[r] > p) {

r--;

}

**if** (l < r) {

**int** temp = a[l];

a[l] = a[r];

a[r] = temp;

}

}

**int** temp = a[s];

a[s] = a[r];

a[r] = temp;

**return** r;

}

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter n1: ");

**int** n1 = sc.nextInt();

System.***out***.println("Enter array 1:");

**int** a1[] = **new** **int**[n1];

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

a1[i] = sc.nextInt();

}

System.***out***.println("Enter n2: ");

**int** n2 = sc.nextInt();

System.***out***.println("Enter array 2:");

**int** a2[] = **new** **int**[n2];

**for** (**int** i = 0; i < n2; i++) {

a2[i] = sc.nextInt();

}

**if** (n1 != n2) {

System.***out***.println("false");

**return**;

}

**int** s = 0, e = n2 - 1;

*sort*(a2, s, e);

**boolean** flag = **true**;

**for** (**int** i = 0; i < n2; i++) {

**if** (a1[i] != a2[i]) {

System.***out***.println("false");

flag = **false**;

**break**;

}

}

**if** (flag) {

System.***out***.println("true");

}

}

}

Output :   
Enter n1:

3

Enter array 1:

3 7 9

Enter n2:

3

Enter array 2:

3 7 11

false

13.You have been given two arrays/list ‘ARR1’ and ‘ARR2’ consisting of ‘N’ and ‘M’ integers respectively. Your task is to return the number of elements common to ‘ARR1’ and ‘ARR2’ and the number of elements in the union of ‘ARR1’ and ‘ARR2’.   
  
Program : (Similarities)  
**package** com.handson;

**import** java.util.Scanner;

**public** **class** Similarities {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter n1: ");

**int** n1 = sc.nextInt();

System.***out***.println("Enter array 1:");

**int** a1[] = **new** **int**[n1];

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

a1[i] = sc.nextInt();

}

System.***out***.println("Enter n2: ");

**int** n2 = sc.nextInt();

System.***out***.println("Enter array 2:");

**int** a2[] = **new** **int**[n2];

**for** (**int** i = 0; i < n2; i++) {

a2[i] = sc.nextInt();

}

**int** inter = 0,union = 0;

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

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

**if** (a1[i] == a2[j]) {

inter++;

**break**;

}

}

}

union = n1+n2-inter;

System.***out***.println(inter+ " " +union);

}

}

Output :   
Enter n1:

5

Enter array 1:

1 2 3 4 5

Enter n2:

4

Enter array 2:

2 4 6 8

2 7

14.Given two integer arrays nums1 and nums2, return an array of their intersection. Each element in the result must appear as many times as it shows in both arrays and you may return the result in any order.   
  
Program : (Intersection)  
package com.handson;

import java.util.\*;

public class Intersection {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter number of element: ");

int n = sc.nextInt();

int a[]=new int[n];

System.out.print("Enter of element of array 1: ");

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

a[i]=sc.nextInt();

}

System.out.print("Enter number of element: ");

int m = sc.nextInt();

System.out.print("Enter of element of array 2: ");

int b[]=new int[m];

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

b[i]=sc.nextInt();

}

Map<Integer, Integer> counts = new HashMap<>();

for (int num : a) {

counts.put(num, counts.getOrDefault(num, 0) + 1);

}

List<Integer> result = new ArrayList<>();

for (int num : b) {

if (counts.containsKey(num) && counts.get(num) > 0) {

result.add(num);

counts.put(num, counts.get(num) - 1);

}

}

int[] resultArray = new int[result.size()];

for (int i = 0; i < result.size(); i++) {

resultArray[i] = result.get(i);

}

for(int i:result) {

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

}

}

}

Output :   
Enter number of element: 3

Enter of element of array 1: 4 9 5

Enter number of element: 5

Enter of element of array 2: 9 4 9 8 4

9 4

15.Given an array 'arr' of size 'N', find the minimum number of swaps required to be made between any two elements to sort the array in ascending order. The array does not contain duplicates i.e all the elements of the array are distinct.  
  
Program : (MinimumSwaps)  
**package** com.handson;

**import** java.util.Scanner;

**public** **class** MinimumSwaps {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter n: ");

**int** n = sc.nextInt();

System.***out***.println("Enter array :");

**int** a[] = **new** **int**[n];

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

a[i] = sc.nextInt();

}

**int** c=0;

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

**int** minIndex = i;

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

**if** (a[j] < a[minIndex]) {

minIndex = j;

}

}

**if** (minIndex != i) {

**int** temp = a[minIndex];

a[minIndex] = a[i];

a[i] = temp;

c++;

}

}

System.***out***.println(c);

}

}

Output :

Enter n:

5

Enter array :

2 4 5 1 3

3

16. A sequence of numbers is called an arithmetic progression if the difference between any two consecutive elements is the same.Given an array of numbers arr, return true if the array can be rearranged to form an arithmetic progression. Otherwise, return false.  
  
Program : (ArithmeticProgression)  
**package** com.handson;

**import** java.util.Scanner;

**public** **class** ArithmeticProgression {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter n: ");

**int** n = sc.nextInt();

System.***out***.println("Enter array :");

**int** a[] = **new** **int**[n];

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

a[i] = sc.nextInt();

}

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

**int** minIndex = i;

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

**if** (a[j] < a[minIndex]) {

minIndex = j;

}

}

**if** (minIndex != i) {

**int** temp = a[minIndex];

a[minIndex] = a[i];

a[i] = temp;

}

}

**boolean** flag = **true**;

**int** diff = a[1] - a[0];

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

**if** (a[i] - a[i - 1] != diff) {

flag = **false**;

**break**;

}

}

**if** (flag) {

System.***out***.println("True");

} **else** {

System.***out***.println("false");

}

}

}

Output :   
Enter n:

3

Enter array :

4 1 2

false

17.Given an array of meeting time intervals where intervals[i] = [starti, endi], determine if a person could attend all meetings.  
  
Program : (MeetingRoom)  
**package** com.handson;

**import** java.util.Scanner;

**public** **class** MeetingRoom {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter row: ");

**int** n = sc.nextInt();

System.***out***.println("Enter array :");

**int** a[][] = **new** **int**[n][2];

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

**for**(**int** j =0; j< 2;j++) {

a[i][j] = sc.nextInt();

}

}

**int** temp;

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

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

**if** (a[j][0] > a[j+1][1]) {

temp = a[j][0];

a[j][0] = a[j + 1][1];

a[j + 1][0] = temp;

}

}

}

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

**for**(**int** j =0; j< 2;j++) {

System.***out***.print(a[i][j] + " ");

}

}

**boolean** flag = **true**;

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

**if**(a[i][0] < a[i - 1][1]) {

flag = **false**;

**break**;

}

}

**if** (flag) {

System.***out***.println("True");

} **else** {

System.***out***.println("false");

}

}

}

Output :   
  
18.You have been given the start and end times of 'N' intervals. Write a function to check if any two intervals overlap with each other.  
  
Program : (OverLapping)  
package com.handson;

import java.util.\*;

public class Overlaping {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter number of intervals: ");

int n = sc.nextInt();

int arr[][] = new int[n][2];

System.out.println("Enter the intervals ");

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

arr[i][0] = sc.nextInt();

arr[i][1] = sc.nextInt();

}

Arrays.sort(arr, new Comparator<int[]>() {

@Override

public int compare(int[] a, int[] b) {

return Integer.compare(a[0], b[0]);

}

});

boolean canAttendAll = true;

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

if (arr[i][1] > arr[i + 1][0]) {

canAttendAll = false;

break;

}

}

System.out.println(canAttendAll ? "false" : "true");

sc.close();

}

}

Output :   
Enter number of intervals: 3

Enter the intervals

5 7

1 3

9 13

false

19.You are given the list of costs of pants in an array “pants”, shirts in an array “shirts”, shoes in an array “shoes”, and skirts in an array “skirts”. You are also given a budget amount ‘X’ to spend. You want to buy exactly 1 item from each list. Your task is to determine the total number of possible combinations that you can buy, given that the total amount of your purchase does not exceed your budget amount.   
  
Program : (Shopping)  
**package** com.handson;

**import** java.util.\*;

**public** **class** Shopping {

**public** **static** **void** main(String [] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter n for pants:");

**int** n1 = sc.nextInt();

**int** pants[] = **new** **int**[n1];

System.***out***.println("Enter pants array:");

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

pants[i] = sc.nextInt();

}

System.***out***.println("Enter n for shirts:");

**int** n2 = sc.nextInt();

**int** shirts[] = **new** **int**[n2];

System.***out***.println("Enter array:");

**for** (**int** i = 0; i < n2; i++){

shirts[i] = sc.nextInt();

}

System.***out***.println("Enter n for shoe:");

**int** n3 = sc.nextInt();

**int** shoe[] = **new** **int**[n3];

System.***out***.println("Enter array:");

**for** (**int** i = 0; i < n3; i++){

shoe[i] = sc.nextInt();

}

System.***out***.println("Enter n for skirt:");

**int** n4 = sc.nextInt();

**int** skirt[] = **new** **int**[n4];

System.***out***.println("Enter array:");

**for** (**int** i = 0; i < n4; i++){

skirt[i] = sc.nextInt();

}

System.***out***.println("Enter the amount :");

**int** budget = sc.nextInt();

**int** answer = 0;

**int**[]ans = **new** **int**[n1 \* n2];

**int** x = 0;

**for** (**int** pant : pants) {

**for** (**int** i : shirts) {

ans[x++] = pant + i;

}

}

Arrays.*sort*(ans);

**for** (**int** i : shoe) {

**for** (**int** j : skirt) {

**int** r = (budget - i) - j;

answer += *countValidCombinations*(ans, r);

}

}

System.***out***.println(answer);

}

**public** **static** **int** countValidCombinations(**int**[] array, **int** value) {

**int** left = 0;

**int** right = array.length;

**while** (left < right) {

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

**if** (array[mid] <= value) {

left = mid + 1;

} **else** {

right = mid;

}

}

**return** left;

}

}

Output :   
Enter n for pants:

2

Enter pants array:

3 4

Enter n for shirts:

2

Enter array:

2 4

Enter n for shoe:

1

Enter array:

2

Enter n for skirt:

1

Enter array:

1

Enter the amount :

10

3

20.You must buy exactly two chocolates in such a way that you still have some non-negative leftover money. You would like to minimize the sum of the prices of the two chocolates you buy. Return the amount of money you will have left over after buying the two chocolates. If there is no way for you to buy two chocolates without ending up in debt, return the money. Note that the leftover must be non-negative.   
  
Program : (Chocolate)  
**package** com.handson;

**import** java.util.Scanner;

**public** **class** Chocolate {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("Enter n:");

**int** n = sc.nextInt();

**int** a[] = **new** **int**[n];

System.***out***.println("Enter array:");

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

a[i] = sc.nextInt();

}

System.***out***.println("enter the money :");

**int** m = sc.nextInt();

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

**int** pos = i;

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

**if** (a[j] < a[pos]) {

pos = j;

}

}

**int** temp = a[pos];

a[pos] = a[i];

a[i] = temp;

}

**if**(a[0] + a[1] <= m)

System.***out***.println(m-(a[0] + a[1]));

**else**

System.***out***.println(m);

}

}

Output :   
Enter n:

3

Enter array:

1 2 2

enter the money :

3

0

21.Given n points on a 2D plane where points[i] = [xi, yi], Return the widest vertical area between two points such that no points are inside the area. A vertical area is an area of fixed width extending infinitely along the yaxis (i.e., infinite height). The widest vertical area is the one with the maximum.

Program : (WidestVertical)  
package com.handson;

import java.util.Scanner;

public class WidestVertical {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter number of rouws : ");

int r = sc.nextInt();

int arr[][] = new int[r][2];

System.out.println("Enter the elements: ");

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

arr[i][0] = sc.nextInt();

arr[i][1] = sc.nextInt();

}

int a[]=new int[r];

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

a[i]=arr[i][0];

}

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

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

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

int temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

int max=Integer.MIN\_VALUE;

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

max=Math.max(max, Math.abs(a[i]-a[i+1]));

}

System.out.print(max);

sc.close();

}

}

Output :   
Enter number of rows : 6

Enter the elements:

3 1

9 0

1 0

1 4

5 3

8 8

3

22. Alice and Bob have a different total number of candies. You are given two integer arrays aliceSizes and bobSizes where aliceSizes[i] is the number of candies of the ith box of candy that Alice has and bobSizes[j] is the number of candies of the jth box of candy that Bob has. Since they are friends, they would like to exchange one candy box each so that after the exchange, they both have the same total amount of candy.   
  
Program : (FairCandy)  
package com.handson;

import java.util.Scanner;

public class FairCandy {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter number of element: ");

int m = sc.nextInt();

int arr[]=new int[m];

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

arr[i]=sc.nextInt();

}

System.out.print("Enter number of element: ");

int n = sc.nextInt();

int brr[]=new int[n];

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

brr[i]=sc.nextInt();

}

int aliceTotal = 0;

int bobTotal = 0;

for (int candies : arr) aliceTotal += candies;

for (int candies : brr) bobTotal += candies;

outer:

for (int i = 0; i < m; i++)

for (int j = 0; j < n; j++)

if ((aliceTotal - arr[i] + brr[j]) == (bobTotal - brr[j] + arr[i])) {

System.out.print(arr[i]+" "+brr[j]);

break outer;

}

}}

Output :   
Enter number of element: 2

1 2

Enter number of element: 2

2 3

1 2

23. Dhoni is the security administrator of the company you are working and you are junior to him. Dhoni’s job is to come up with new and innovative encryption algorithms to encrypt the data while sending it to the receiver. He came up with a new encryption algorithm as follows: Let’s suppose there is a string ‘STR’ of length ‘N’ which consists of lowercase English letters and numbers from ‘0’ to ‘9’. The above algorithm will first find the total of all the digits occurring in the string ‘STR’ and remove them from the string, then will sort the whole string in alphabetical order and push all digits sum at the end of the above-generated string. In case there is no digit present in the given string, we don't add anything at the end.

Program : (StringSum)  
package com.handson;

import java.util.Scanner;

public class StringSum {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter string : ");

String s = sc.nextLine();

StringBuilder letters = new StringBuilder();

int digitSum = 0;

for (char ch : s.toCharArray()) {

if (Character.isDigit(ch)) {

digitSum += Character.getNumericValue(ch);

} else if (Character.isLetter(ch)) {

letters.append(ch);

}

}

char[] letterArray = letters.toString().toCharArray();

bubbleSort(letterArray);

StringBuilder sortedLetters = new StringBuilder();

for (char ch : letterArray) {

sortedLetters.append(ch);

}

if (digitSum > 0) {

sortedLetters.append(digitSum);

}

System.out.print(sortedLetters.toString());

}

public static void bubbleSort(char[] array) {

int n = array.length;

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

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

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

char temp = array[j];

array[j] = array[j + 1];

array[j + 1] = temp;

}

}

}

}

}

Output :   
Enter string : x3a8

ax11

24.You are given two strings order and s. All the characters of order are unique and were sorted in some custom order previously. Permute the characters of s so that they match the order that order was sorted. More specifically, if a character x occurs before a character y in order, then x should occur before y in the permuted string. Return any permutation of s that satisfies this property.   
  
Program : ()  
package com.handson;

import java.util.\*;

public class CustomSort {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter string : ");

String order = sc.nextLine();

System.out.print("Enter string : ");

String s = sc.nextLine();

Map<Character, Integer> map = new HashMap<>();

for (char c : s.toCharArray()) {

map.put(c, map.getOrDefault(c, 0) + 1);

}

StringBuilder ans = new StringBuilder();

for (char c : order.toCharArray()) {

if (map.containsKey(c)) {

int count = map.get(c);

ans.append((c+"").repeat(count));

map.remove(c);

}

}

for(char ch : map.keySet()){

int count = map.get(ch);

ans.append((ch+"").repeat(count));

}

System.out.print(ans.toString());

sc.close();

}

}

Output :   
Enter string : cba

Enter string : abcd

Cbad

25.You are given an array of strings of the same length words. In one move, you can swap any two even indexed characters or any two odd indexed characters of a string words[i]. Two strings words[i] and words[j] are special-equivalent if after any number of moves, words[i] == words[j].

Program : (SpecialEvents)  
package com.handson;

import java.util.\*;

public class SpecialEvents {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter number of string : ");

int n=sc.nextInt();

System.out.print("Enter strings : ");

String s[]=new String[n];

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

s[i] = sc.nextLine();

}

HashSet<String> h=new HashSet<>();

for(String str:s){

int[] odd=new int[26];

int [] even=new int[26];

for(int i=0;i<str.length();i++){

if(i%2==0){

even[str.charAt(i)-'a']++;

}else{

odd[str.charAt(i)-'a']++;

}}

String key=Arrays.toString(odd)+Arrays.toString(even);

h.add(key);

}

System.out.print(h.size());

}

}

Output :   
Enter number of string : 6

Enter strings : abc

acb

bac

bca

cab

4

27.Given an array of integers arr, sort the array by performing a series of pancake flips. In one pancake flip we do the following steps: Choose an integer k where 1 <= k <= arr.length. Reverse the sub-array arr[0...k-1] (0-indexed).  
  
Program : (PancakeFlip)  
package com.handson;

import java.util.Arrays;

import java.util.Scanner;

public class PancakeFlip {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the number of elements (n):");

int n = sc.nextInt();

System.out.println("Enter the array elements:");

int arr[] = new int[n];

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

arr[i] = sc.nextInt();

}

int[] flips = pancakeSort(arr);

System.out.println("Pancake flips: " + Arrays.toString(flips));

sc.close();

}

public static int[] pancakeSort(int[] A) {

int n = A.length;

int[] flips = new int[2 \* n];

int flipCount = 0;

for (int x = n; x > 0; --x) {

int i;

for (i = 0; A[i] != x; ++i);

if (i != x - 1) {

if (i != 0) {

flip(A, i + 1);

flips[flipCount++] = i + 1;

}

flip(A, x);

flips[flipCount++] = x;

}

}

return Arrays.copyOf(flips, flipCount);

}

private static void flip(int[] A, int k) {

for (int left = 0, right = k - 1; left < right; ++left, --right) {

int tmp = A[left];

A[left] = A[right];

A[right] = tmp;

}

}

}

Output :   
Enter the number of elements (n):

4

Enter the array elements:

3 2 4 1

Pancake flips: [ 4, 2, 3, 2]

28.Given an array, find the total number of inversions of it. If (i < j) and (A[i] > A[j]), then pair (i, j) is called an inversion of an array A. We need to count all such pairs in the array.  
  
Program : (InversionArray)  
package com.handson;

import java.util.Scanner;

public class InversionArray {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter number of elements: ");

int n = sc.nextInt();

System.out.println("enter the array elements");

int a[]=new int[n];

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

a[i]=sc.nextInt();

}

int c=0;

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

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

if(a[i]>a[j] && i<j) {

c++;

System.out.println("pairs:("+ a[i] + " " +a[j]+")");

}

}

}

sc.close();

System.out.print(c);

}

}

Output :   
Enter number of elements:

5

enter the array elements

1 9 6 4 5

pairs:(9 6)

pairs:(9 4)

pairs:(9 5)

pairs:(6 4)

pairs:(6 5)

5

29.Given an integer array nums, reorder it such that nums[0] < nums[1] > nums[2] < nums[3].... You may assume the input array always has a valid answer.  
Program : (WiggleSort2)  
package com.handson;

import java.util.Arrays;

import java.util.Scanner;

public class WiggleSort2 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the n:");

int n = sc.nextInt();

System.out.println("Enter the a:");

int[] nums = new int[n];

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

nums[i] = sc.nextInt();

}

wiggleSort(nums);

System.out.println("Reordered array: " + Arrays.toString(nums));

sc.close();

}

public static void wiggleSort(int[] nums) {

for (int i = 0; i < nums.length - 1; i++) {

if ((i % 2 == 0 && nums[i] > nums[i + 1]) || (i % 2 == 1 && nums[i] < nums[i + 1])) {

int temp = nums[i];

nums[i] = nums[i + 1];

nums[i + 1] = temp;

}

}

}

}

Output :   
Enter the n:

6

Enter the a:

1 3 2 2 3 1

Reordered array: [1, 3, 2, 3, 1, 2]

30. Given an array nums of n integers, return an array of all the unique quadruplets [nums[a], nums[b], nums[c], nums[d]] such that: 0 <= a, b, c, d < n a, b, c, and d are distinct. nums[a] + nums[b] + nums[c] + nums[d] == target You may return the answer in any order.   
  
Program : (FourSum)  
package com.handson;

import java.util.\*;

public class FourSum {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the n:");

int n = sc.nextInt();

System.out.println("Enter the array :");

int[] a = new int[n];

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

a[i] = sc.nextInt();

}

System.out.println("Enter the target:");

int t = sc.nextInt();

List<List<Integer>> quadruplets = fourSum(a, t);

System.out.println("Quadruplets: " + quadruplets);

sc.close();

}

public static List<List<Integer>> fourSum(int[] nums, int t) {

int n = nums.length;

List<List<Integer>> ans = new ArrayList<>();

if (n < 4) {

return ans;

}

Arrays.sort(nums);

for (int i = 0; i < n - 3; ++i) {

if (i > 0 && nums[i] == nums[i - 1]) {

continue;

}

for (int j = i + 1; j < n - 2; ++j) {

if (j > i + 1 && nums[j] == nums[j - 1]) {

continue;

}

int k = j + 1, l = n - 1;

while (k < l) {

long x = (long) nums[i] + nums[j] + nums[k] + nums[l];

if (x < t) {

++k;

} else if (x > t) {

--l;

} else {

ans.add(List.of(nums[i], nums[j], nums[k++], nums[l--]));

while (k < l && nums[k] == nums[k - 1]) {

++k;

}

while (k < l && nums[l] == nums[l + 1]) {

--l;

}

}

}

}

}

return ans;

}

}

Output :   
Enter the n:

6

Enter the array :

1 0 -1 0 -2 2

Enter the target:

0

Quadruplets: [[-2, -1, 1, 2], [-2, 0, 0, 2], [-1, 0, 0, 1]]