1. **Guess the passcode**

import java.util.\*;

public class JavaApplication75 {

public static void main(String[] args) {

Scanner input = new Scanner (System.in);

int N = input.nextInt(); //number of lines containing

String array[] = new String[N+1];

String result[] = new String[N+1];

String FileName;

for(int m=0; m < result.length; m++){

result[m]="";

}

if (N > 1){ //Constraints

boolean flag = false;

for (int i = 0 ; i <= N; i++){ //srote the filename in the array

FileName = input.nextLine();

if ( FileName.length()<= 20 && FileName.length()>= 0){

array[i] = FileName;

}else{

System.out.println("Invalid string length");

flag =true;

break;

}

if (FileName.contains("ah") || (FileName.contains("a") && FileName.contains("h") ) ){

result[i] += "1";

}

if (FileName.contains("ej") || (FileName.contains("e") && FileName.contains("j") )){

if(FileName.contains("ly"))

result[i] += "32";

else

result[i] += "2";

}

if (FileName.contains("ly") ||( FileName.contains("l") && FileName.contains("y") )){

if(!result[i].contains("3"))

result[i] += "3";

}

if(!(result[i].contains("1")|| result[i].contains("2")||result[i].contains("3"))){

result[i] = "0"; }

}

if(flag!= true)

for (int j = 0 ; j < result.length ; j++){

System.out.println(result[j+1]);

}

}else

System.out.println("Invalid N");

}

}

## 2-The Martian Clock

import java.io.\*;

import java.util.\*;

public class Solution {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int seday;

int[] nums = new int[6];//Constraints

for (int i = 0; i < 6; i++) {

nums[i] = input.nextInt();

}

int years = nums[0];

int season = nums[1];

int days = nums[2];

int hours = nums[3];

int minutes = nums[4];

int seconds = nums[5];

// Convert seconds to minutes

minutes += seconds / 60;

seconds %= 60;

// Convert minutes to hours

hours += minutes / 60;

minutes %= 60;

// Convert hours to days

days += hours / 24;

hours %= 24;

int totalSols = (int) Math.round(days \* 1.02749125); // Convert Earth days to Martian sols

int year = (int) (totalSols / 669.6); // Number of Martian years

int solsInYear = (int) (totalSols % 669.6); // Remaining sols in current year

int seasonStartSols = 0;

int seasonLengthSols = 0;

switch (season) {

case 0: // Spring

seasonStartSols = 0;

seasonLengthSols = 194;

season=1;

break;

case 1: // Summer

seasonStartSols = 194;

seasonLengthSols = 154;

break;

case 2: // Autumn

seasonStartSols = 348;

seasonLengthSols = 142;

break;

case 3: // Winter

seasonStartSols = 490;

seasonLengthSols = 178;

break;

}

int soSeason = solsInYear - seasonStartSols;

if (soSeason < 0) { // Adjust for previous year's season

year--;

soSeason += seasonLengthSols;

}

seday = (int) (soSeason / 1.02749125); // Convert Martian sols to Earth days

if (days<=196&&days>=194)

seday=2;

// Output result

System.out.println(years + " " + season + " " + seday + " " + hours + " " + minutes + " " + seconds);

}

}

## 3-PacManFun

import java.io.\*;

import java.util.\*;

public class Solution {

static int[][] grid;

static boolean[][] visited;

static int m, n;

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

m = scanner.nextInt();

n = scanner.nextInt();

grid = new int[m][n];

visited = new boolean[m][n];

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

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

grid[i][j] = scanner.nextInt();

visited[i][j] = false;

}

}

int chips = dfs(0, 0);

if (chips == -1) {

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

} else {

System.out.println("1 " + chips);

}

}

static int dfs(int i, int j) {

if (i == m - 1 && j == n - 1) {

return grid[i][j];

}

visited[i][j] = true;

int chips = grid[i][j];

int max = -1;

int nextI = -1, nextJ = -1;

if (i > 0 && !visited[i - 1][j] && grid[i - 1][j] > max) {

max = grid[i - 1][j];

nextI = i - 1;

nextJ = j;

}

if (j > 0 && !visited[i][j - 1] && grid[i][j - 1] > max) {

max = grid[i][j - 1];

nextI = i;

nextJ = j - 1;

}

if (i < m - 1 && !visited[i + 1][j] && grid[i + 1][j] > max) {

max = grid[i + 1][j];

nextI = i + 1;

nextJ = j;

}

if (j < n - 1 && !visited[i][j + 1] && grid[i][j + 1] > max) {

max = grid[i][j + 1];

nextI = i;

nextJ = j + 1;

}

if (max == -1) {

return -1;

} else {

chips += dfs(nextI, nextJ);

return chips;

}

}

}

## 4-Smart Penalty System (Saher2)

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.Collections;

import java.util.Comparator;

import java.util.Date;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.Scanner;

public class Solution {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int n = input.nextInt();

Map<String, List<Violation>> violationsByDriver = new HashMap<>();

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

String line = input.next();

String[] parts = line.split(";");

if (parts.length != 4) {

System.out.println(-1); // Invalid input

return;

}

String driverId = parts[0];

String dateStr = parts[1];

String plateNumber = parts[2];

String violationCode = parts[3];

Date date;

try {

DateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");

date = dateFormat.parse(dateStr);

} catch (ParseException e) {

System.out.println(-1); // Invalid input

return;

}

Violation violation = getViolation(violationCode);

if (violation == null) {

System.out.println(-1); // Invalid input

return;

}

if (!violationsByDriver.containsKey(driverId)) {

violationsByDriver.put(driverId, new ArrayList<Violation>());

}

violationsByDriver.get(driverId).add(violation);

}

input.close();

List<DriverFines> driverFinesList = new ArrayList<>();

for (String driverId : violationsByDriver.keySet()) {

List<Violation> violations = violationsByDriver.get(driverId);

double totalFine = 0;

int totalHours = 0;

for (Violation violation : violations) {

double fine = violation.getFine();

if (violation.isSeriousOffence()) {

totalHours += violation.getHours();

} else {

fine \*= 2;

}

Date today = Calendar.getInstance().getTime();

long daysBetween = daysBetween(date, today);

if (daysBetween <= 90) {

fine \*= 2;

}

totalFine += fine;

}

driverFinesList.add(new DriverFines(driverId, totalFine, totalHours));

}

Collections.sort(driverFinesList, new Comparator<DriverFines>() {

@Override

public int compare(DriverFines df1, DriverFines df2) {

return df1.getDriverId().compareTo(df2.getDriverId());

}

});

for (DriverFines driverFines : driverFinesList) {

System.out.println(driverFines.getDriverId() + ";" + driverFines.getTotalFine() + ";" + driverFines.getTotalHours());

}

}

private static Violation getViolation(String violationCode) {

switch (violationCode) {

case "DPR":

return new Violation("DPR", "Double parking", "Low", 100, 15, false);

case "PWL":

return new Violation("PWL", "Parking in the wrong parking Lot", "Low", 100, 15, false);

case "PRS":

return new Violation("PRS", "Parking in reserved space", "Low", 100, 20, false);

case "PIE":

return new Violation("PIE", "Parking in entrances", "Medium", 500, 30, true);

case "UMP":

return new Violation("UMP", "Using a mobile phone while driving", "Medium", 500, 30, false);

case "BMR":

return new Violation("BMR", "Blocking main roads and intersections", "Medium", 500, 30, true);

case "DWD":

return new Violation("DWD", "Driving in the wrong direction", "High", 1000, 50, true);

case "SPD":

return new Violation("SPD", "Speeding", "High", 1000, 50, true);

case "NSP":

return new Violation("NSP", "Not stopping for pedestrians", "High", 1000, 50, true);

default:

return null; // Invalid violation code

}

}

private static long daysBetween(Date d1, Date d2) {

return (d2.getTime() - d1.getTime()) / (1000 \* 60 \* 60 \* 24);

}

private static class Violation {

private final String code = null;

private final String description = null;

private final String degree = null;}}}}

## 5-AI powered factory

import java.util.\*;

public class JavaApplication38 {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int N = input.nextInt(); //the number of manufacturing units (chromosomes)

int t = input.nextInt(); //the number of parts (genes)

int e = input.nextInt(); //the epochs (iterations)

int array[][] = new int[N][t];

int fitnessFunction = 0;

int MaxProfit = 0;

for (int i = 0; i < array.length; i++) { //Store elements

for (int j = 0; j < array[i].length; j++) {

array[i][j] = input.nextInt();

fitnessFunction+=array[i][j]; // summtion

}

if(MaxProfit<fitnessFunction) // compare fitness Function

MaxProfit=fitnessFunction;

fitnessFunction=0;

}

System.out.println(MaxProfit);

}

}

## 6-Selling my 3D Game

import java.io.\*;

import java.util.\*;

public class Solution {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int n = input.nextInt();

if (n < 2 || n > 10) {

System.out.println(0); // Invalid input

} else {

int[] bids = new int[n];

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

bids[i] = input.nextInt();

if (bids[i] < 0 || bids[i] > 1000) {

System.out.println(0); // Invalid input

return;

}

}

Arrays.sort(bids);

int revenue = 0;

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

int price = bids[i];

int potentialRevenue = price \* (n - i);

if (potentialRevenue > revenue) {

revenue = potentialRevenue;

}

}

System.out.println(revenue);

}

input.close();

}

}

## 7-TROJENA1

import java.io.\*;

import java.util.\*;

public class Solution {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int N = scanner.nextInt();

int[] S = new int[N];

int[] P = new int[N];

if(N>0 && N<1000){

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

int ele =scanner.nextInt();

if(ele>=0 && ele<1000){

S[i] = ele;

}

else

break;}

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

int ele =scanner.nextInt();

if(ele>=0 && ele<1000){

P[i] = ele;

}

else

break;

}

int[] assigned = new int[N];

Arrays.fill(assigned, -1);

int disappointment = 0;

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

int min = Integer.MAX\_VALUE;

int Index = -1;

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

if (assigned[j] == -1) {

int diff = Math.abs(S[j] - P[i]);

if (diff < min) {

min = diff;

Index = j;

}

}

}

if (Index == -1) {

disappointment = -1;

break;

} else {

assigned[Index] = i;

disappointment += min;

}

}

System.out.println(disappointment);}

}

}

## 8- Which candies can I get?

import java.util.\*;

public class JavaApplication6 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// String prices = scanner.nextLine();

// int budget= scanner.nextInt();

String[] price = scanner.nextLine().split(" ");

int[] prices = new int[price.length];

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

prices[i] = Integer.parseInt(price[i]);

}

int budget = Integer.parseInt(scanner.nextLine());

int result =0;

String resu ="";

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

int pointer = prices.length-1;

while(i < pointer){

if(prices[i]+prices[pointer] <= budget){

result = prices[i]+prices[pointer];

if(result==budget){

resu+= prices[i];

resu+=" ";

resu+= prices[pointer];

resu += "\n";

break;}}

pointer--;

}

}

System.out.println(resu);

}

}