BITACORA ALGORITMOS

ANALISIS Y DISEÑO DE ALGORITMOS

LUIS FELIPE VELASCO TAO

UNIVERSIDAD DE SAN BUENAVENTURA

12 DE FEBRERO

2020

Contenido

ALGORITMO 1: Combustible Gastado

Juancito quiere calcular y mostrar la cantidad de litros de combustible gastado en un viaje, con un auto que hace 12 Km/L. Para eso, le gustaría que lo ayudes a través de un programa sencillo. Para realizar el cálculo, tienes que leer el tiempo (en horas) y la velocidad media (en Km/h) del viaje. De esta forma se puede obtener la distancia, y luego, calcular la cantidad de litros necesarios. Mostrar el valor con tres dígitos luego del punto decimal.

Entrada

La entrada contiene dos enteros. El primero es el tiempo que duró el viaje (en horas). El segundo es la velocidad media del viaje (en Km/h).

Salida

Imprimir cuantos litros de combustible fueron necesarios para hacer el viaje, con tres dígitos luego del punto decimal.

Ecuación

$$V = {}^{Km}/_{h}$$

$$t = h$$

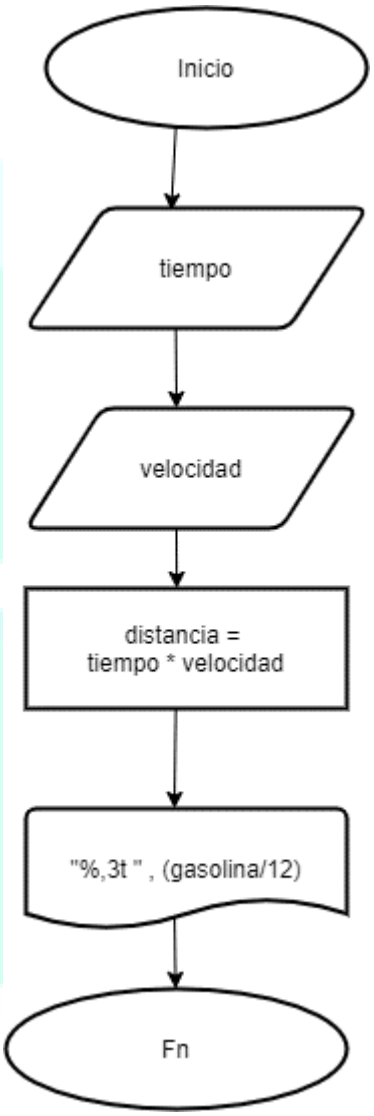
$$D = v * t = \frac{Km}{\cancel{h}} \; x \; \cancel{h} = Km$$

Si el auto gasta 1 litro de combustible cada 12 Km

$$Combustible = \frac{d}{12}$$

Pesudo código

1. Ingresar tiempo (h)
2. Ingresar velocidad (km / h)
3. Hallar distancia = tiempo * velocidad (~~h~~ * km / ~~h~~)
4. Hallar galones (l) = distancia / 12 (distancia en la que se gasta un litro del combustible)
5. Imprimir galones en %.3t (formato)



1 Diagrama de flujo de "Combustible consumido"

Código

```
import java.util.Scanner;

public class Principal {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int t = s.nextInt();
        int v = s.nextInt();
```

```
        double sol = (v*t);
        System.out.printf("%.3f \n",sol/12);
    }
}
```

Python

```
t = input()
v = input()
s = float (t*v)
print ("{0:.3f}".format(s/12))
```

ALGORITMO 2: Calculo simple

En este problema, la tarea consiste en leer un código de un producto 1, el número de unidades del producto 1, el precio de una unidad de producto 1, el código de un producto 2, el número de unidades del producto 2 y el precio de una unidad de producto 2. Después de esto, calcular y mostrar la cantidad a pagar.

Entrada

El archivo de entrada contiene dos líneas de datos. En cada línea habrá 3 valores: Dos enteros y un valor flotante con 2 dígitos después del punto decimal.

Salida

El archivo de salida debe ser un mensaje como en el siguiente ejemplo. Recuerde el espacio antes de ":" y antes del símbolo "R$". El valor debe ser presentado con 2 dígitos después del punto.

Ecuación

Entradas

```
A B C (valor flotante)

D E F (valor flotante)
```

Salida

```
VALOR A PAGAR: R$ (B x C) + (E x F)
```

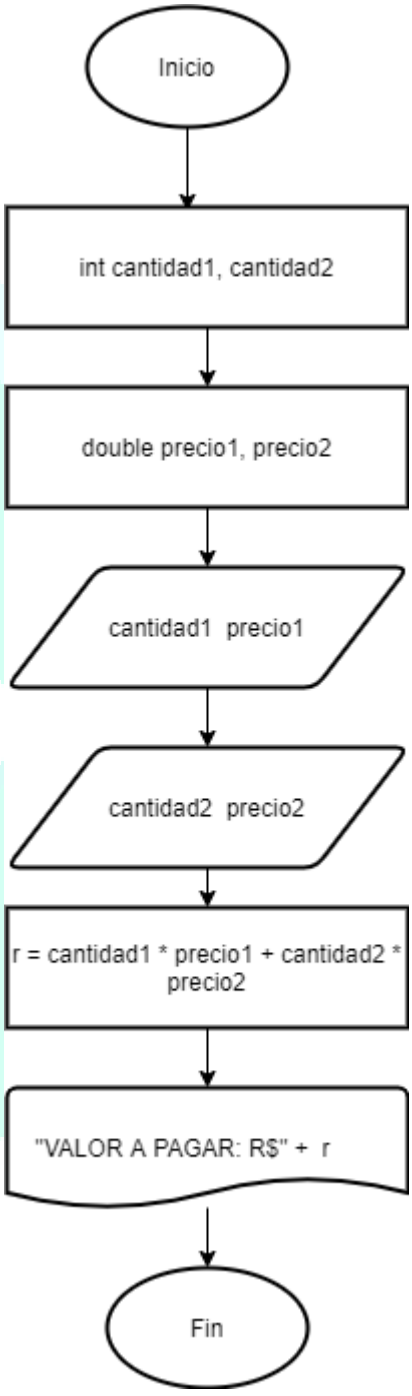
Pseudo código

1. Definir enteros b, e
2. Definir flotantes c, f
3. Ingresar línea 1 (a b c.)
4. Ingresar línea 2 (d e f.)
5. Calcular r = b x c + e x f

6. Imprimir “VALOR A PAGAR: R$”+r

Codigo

```
import java.util.Scanner;
public class Principal {
    public static double metodo(int a, int b,
double c, int d, int e, double f) {
        return (b * c) + (e * f);
    }
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.printf("VALOR A PAGAR: R$ %.2f
\n", metodo(
s.nextInt(), s.nextInt(), s.nextDouble(),
s.nextInt(), s.nextInt(), s.nextDouble())
);
    }
}
```



Python

```
a = input()
b = input()
c = float (input())
d = input()
e = input()
f = float (input())
print ("VALOR A PAGAR: R$ {0:.2f}".format(b*c+e*f))
```

ALGORITMO 3: Promedio 3

Leer cuatro números (N1, N2, N3, N4), con 1 dígito después del punto decimal, correspondiente a 4 resultados obtenidos por un estudiante. Calcular el promedio con pesos 2, 3, 4 e 1 respectivamente, para estos 4 resultados e imprimir el mensaje "Media: " (Promedio), seguido por el cálculo obtenido. Si el promedio es de 7.0 o más, imprimir el mensaje "Aluno aprovado." (Estudiante Aprobado). Si el promedio es menor que 5.0, imprimir el mensaje: "Aluno reprovado." (Estudiante Reprobado). Si el promedio es entre 5.0 and 6.9, incluyendo este, el programa deberá imprimir el mensaje "Aluno em exame." (Estudiante en examen).

En caso de examen, lea una puntuación más. Imprimir el mensaje "Nota do exame: " (Nota de examen) seguido por la puntuación mostrada. Vuelva a calcular el promedio (suma la puntuación del examen con el promedio calculado anteriormente y divida por 2) e imprima

el siguiente mensaje “Aluno aprovado.” (Estudiante Aprobado) en caso de que el promedio sea 5.0 o más o "Aluno reprovado." (Estudiante Reprobado) en caso de que el promedio sea 4.9 o menor. Para estos 2 casos (Aprobado o reprobado después del examen) imprimir el mensaje "Media final: " (Promedio Final) Seguido por el promedio final para este estudiante en la última línea.

Entrada

La entrada contiene cuatro números flotante que representan las calificaciones de los estudiantes.

Salida

Imprimir todas las respuestas con un dígito después del punto decimal.

Ecuación

Entradas

n1, n2, n3, n4 (cuatro números con punto flotante y un numero decimal después de este)

examen (número de punto flotante con un n) |se ingresa al momento de cumplirse la siguiente condición:

$$media = n1 * 0.2 + n2 * 0.3 + n3 * 0.4 + n4 * 0.1$$

$$media \leq 6{,}9 \;\&\&\; media \; > 5{,}0$$

Salidas

Media:

$$media = n1 * 0.2 + n2 * 0.3 + n3 * 0.4 + n4 * 0.1$$

En el caso en el que se cumpla media <= 6,9 y media >= 5,0

$$\boldsymbol{mediaf = \frac{media + eamen}{2}}$$

Pseudo código

1. Leer double nota 1, nota 2, nota 3, nota 4
2. Hallar media = (nota 1 + nota 2 + nota 3 + nota 4) / 4
3. Imprimir “Media: “+media
4. Si media >= 7.0
    a. Imprimir “Aluno aprovado.”

5. Si media < 5.0
    a. Imprimir “Aluno reprovado.”
6. Si media >= 5.0 y media <= 6.9
    a. Imprimir “Aluno em exame.”
    b. Leer double examen
    c. Si (media + examen) / 2 >= 5.0
        i. Imprimir “Aluno aprovado.”
    d. De lo contrario
        i. Imprimir “Aluno reprovado.”
    e. Imprimir “Media final: “+(media + examen) / 2

Código

```
import java.text.DecimalFormat;
import java.util.Scanner;

public class Main {

    private static void notas(double n1, double n2, double n3, double n4) {
        double media = ((n1 * 2) + (n2 * 3) + (n3 * 4) + (n4 * 1))/10;
        System.out.println("Media: " + impresion(media));
        if (media >= 7.0) {
            System.out.println("Aluno aprovado.");
        } else if (media >= 5.0 && media <= 6.9) {
            System.out.println("Aluno em exame.");
            Scanner s = new Scanner(System.in);
            examen(s.nextDouble(), media);
        } else {
            System.out.println("Aluno reprovado.");
        }
    }

    private static void examen(double examen, double media) {
        System.out.println("Nota do examen: " + impresion(examen));
        if (((media + examen) / 2) >= 5.0) {
            System.out.println("Aluno aprovado.");
        } else {
            System.out.println("Aluno reprovado.");
        }
        System.out.println("Media final: " + impresion((media + examen) / 2));
    }

    private static String impresion(double num) {
        DecimalFormat decimalFormat = new DecimalFormat("#.0");
        String format = decimalFormat.format(num);
        return format;
    }
```

```
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        notas(s.nextDouble(), s.nextDouble(), s.nextDouble(), s.nextDouble());
    }
}
```

ALGORITMO 4: Coordenadas de un punto

Escriba un algoritmo que lea dos valores flotantes (x e y), que deben representar las coordenadas de un punto en un plano. A continuación, determine qué cuadrante pertenece el punto o si está sobre uno de los ejes cartesianos o el origen (x = y = 0).

Si el punto está en el origen, escriba el mensaje "Origem".

Si el punto está sobre el eje X, escriba "Eixo X", de lo contrario si el punto está sobre el eje Y, escriba "Eixo Y".

Entrada

La entrada contiene las coordenadas de un punto.

Salida

La salida deberá mostrar en pantalla el cuadrante en el que se encuentra el punto.

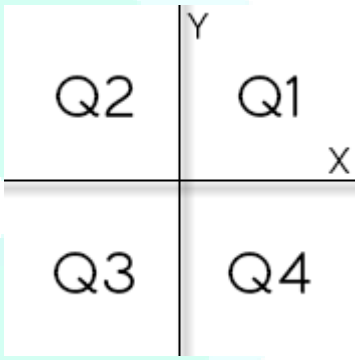
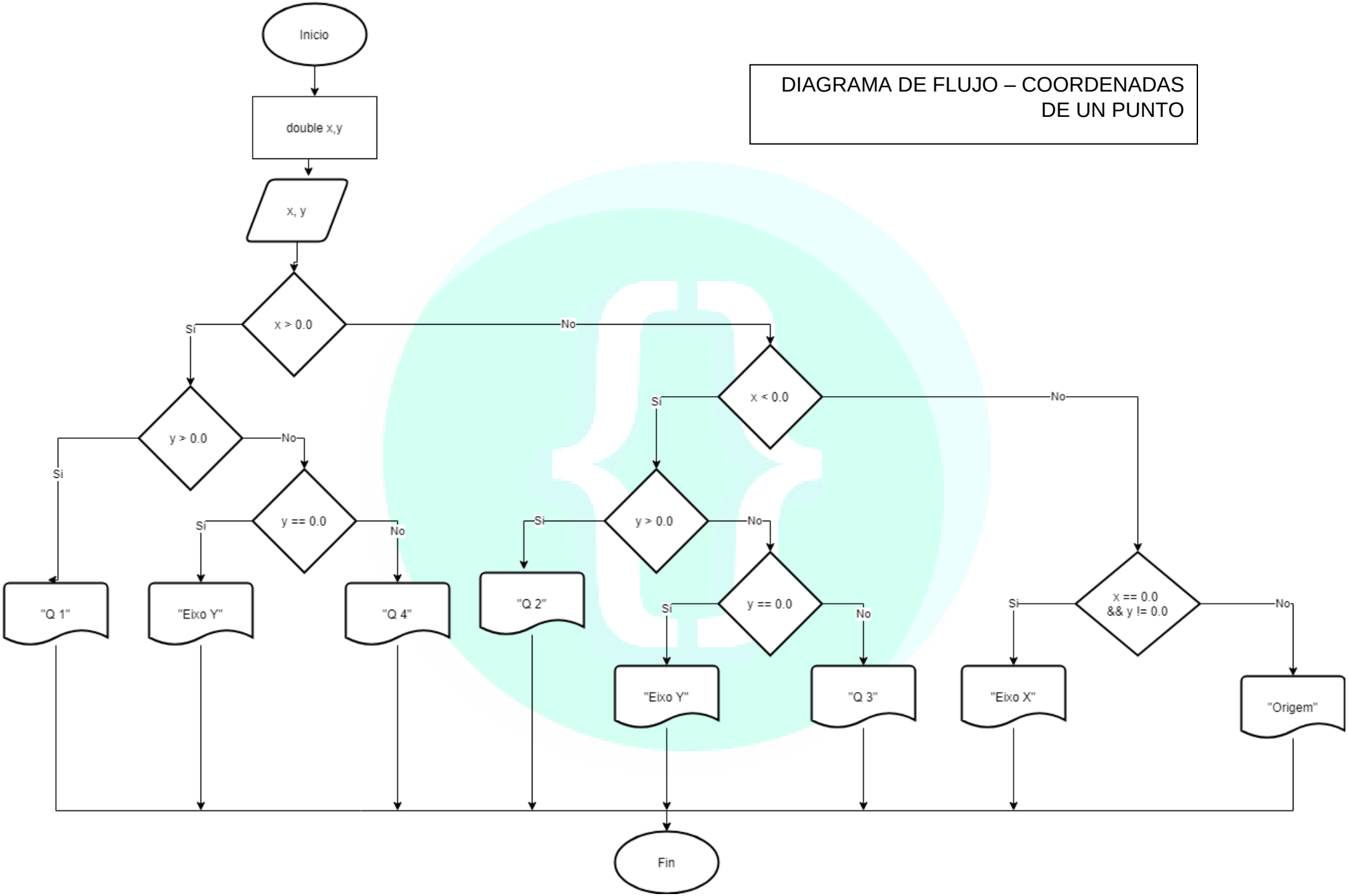


DIAGRAMA DE FLUJO – COORDENADAS DE UN PUNTO



Pseudo código

1. Ingresar x y
2. Si x > 0.0
    a. Si y > 0.0
        i. Imprimir “Q1”
    b. Si y == 0.0
        i. Imprimir “Eixo y”
    c. De lo contrario
        i. Imprimir “Q4”
3. Si x < 0.0
    a. Si y > 0.0
        i. Imprimir “Q2”
    b. Si y == 0.0
        i. Imprimir “Eixo y”
    c. De lo contrario
        i. Imprimir “Q3”
4. Si x == 0.0 & y != 0.0
    a. Imprimir “Eixo x”
5. Si x == 0.0 & y 0= 0.0
    a. Imprimir “Origem”

Código

Java

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.println(coordenadas(s.nextDouble(), s.nextDouble()));
    }
    private static String coordenadas(double x, double y) {
        if (x == 0.0 && y != 0.0) {
            return "Eixo Y";
        } else if (x != 0.0 && y == 0.0) {
            return "Eixo X";
        } else if (x > 0) {
            if (y > 0) {
                return "Q1";
```

```
            } else  {
                return "Q4";
            }
        } else if (x < 0) {
            if (y > 0) {
                return "Q2";
            } else  {
                return "Q3";
            }
        } else {
            return "Origem";
        }
    }
}
```

ALGORITMO 5: Honey Reservoir

Julius is the owner of a large apiry situated in Paraíba. Every year, every six months, Julius collect honey produced by bees of their property and stores it in a CYLINDRICAL container format that facilitates the transport of honey for establishments who order this natural product for commercialization . Once Julio realized due to an increase in honey production, over the previous six months, the container that owned the stand the volume of honey produced by his bees. Julius needs now that you make a program that informs the volume of honey in cm3 and the diameter of the inside of the container in cm, calculate and show:

- What should be the height (in cm) of Inside the container;

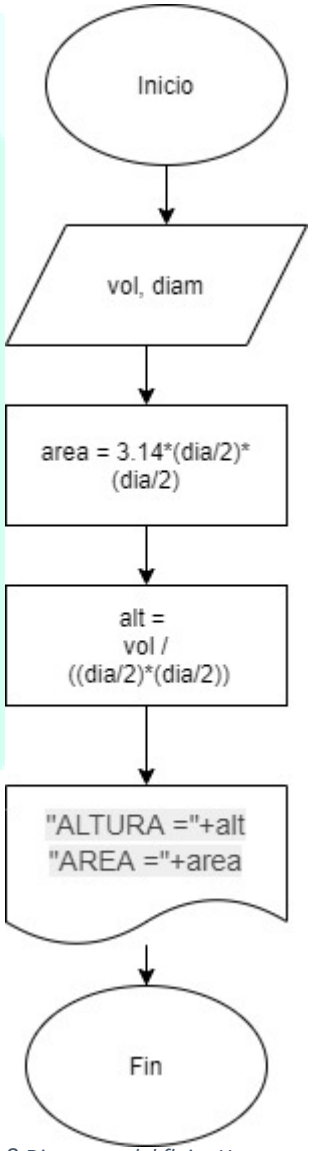
- The area (in cm2) of the mouth (entrance) of the container.

NB .: Consider π = 3.14

Input

The input contains several test cases and ends with EOF. Each test case consists of a line containing two floating-point values of double precision with two decimals after comma, one V (1.00 ≤ V ≤ 10000.00) and one D (1.00 ≤ D ≤ 100.00), respectively representing the volume and the container diameter.

Output



2 Diagrama del flujo: Honey Reservoir

For each test, the output contains the first line "ALTURA = " message, with a space after ALTURA and another after the symbol of equality, followed by the container height value with two decimals after comma and the second line message "AREA = ", also with a space after AREA and another after the symbol of equality, followed by the value of the area of the mouth (entrance) of the container with two decimals after comma.

- Do not forget the line break at the end of the exit, otherwise you will get "Presentation Error".v

Pseudo código

1. Leer volumen
2. Leer diámetro
3. Hallar área = 3.14 * (diámetro/2) ^2
4. Hallar altura = v / área
5. Imprimir

   ALTURA = altura

   AREA = área

Ecuación

$$V = \pi r^2 h \qquad r = \frac{d}{2} \qquad A = \pi r^2 \qquad h = \frac{V}{A}$$

Código

```
import java.util.Scanner;
public class Main {

    static Scanner s = new Scanner(System.in);
    static double r = 0.0;

    public static void main(String[] args) {
        System.out.printf("ALTURA = %.2f \nAREA = %.2f
\n",s.nextDouble()/((potencia(s.nextDouble()/2))),r);
    }
    private static double potencia(double x) {
        r = 3.14*(x*x);
```

```
return r;}}
```

ALGORITMO 6: Sunday Morning

Sunday is market day. Early in the morning many people move to the Parangaba square where happens a fair, known to be the largest in the city. At the fair the Parangaba you can find everything.

Every Sunday, Bino make purchases at the fair. He always mark with his friend Cino, they met at the bus terminal of Parangaba at 8 am to go together to buy at the fair. But often Bino wake up too late and is late for the meeting with his friend.

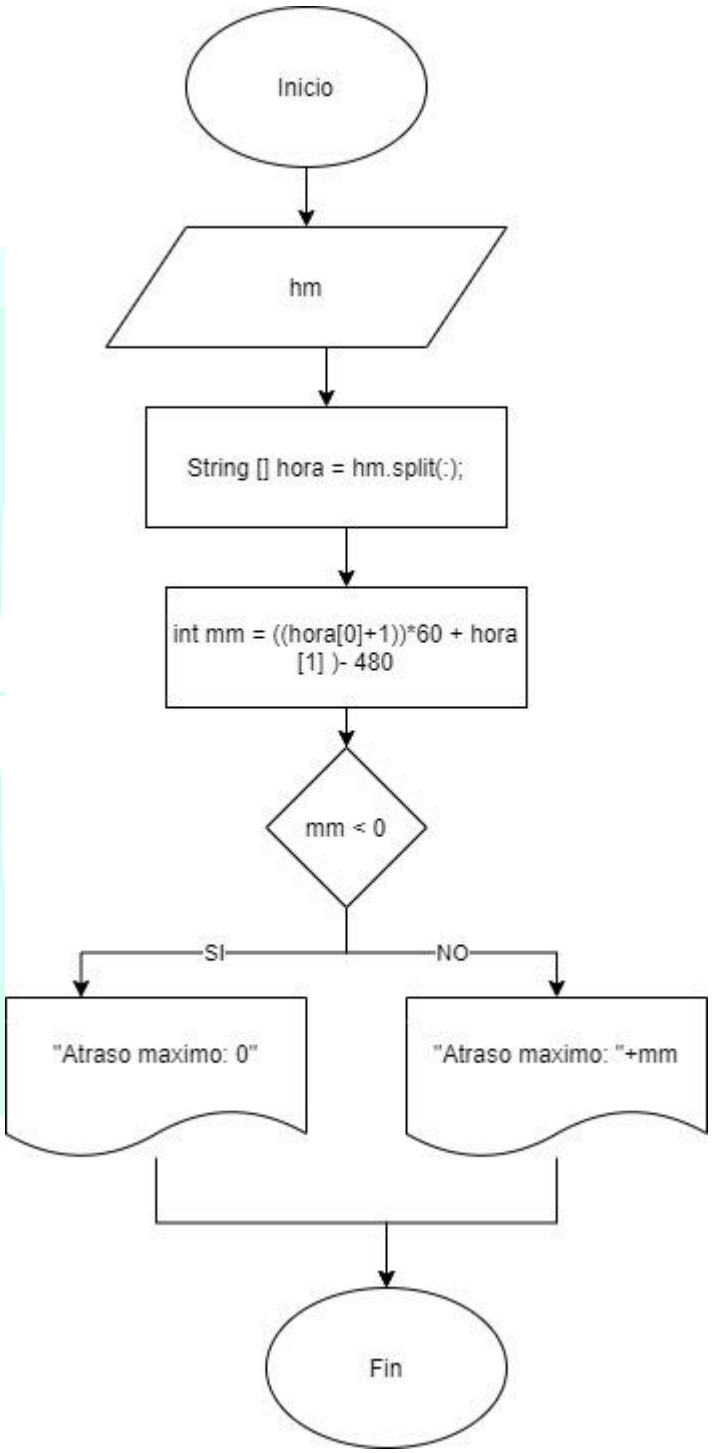
Knowing that Bino takes 30-60 minutes to reach the terminal. Tell the maximum delay Bino.

Input

The input consists of multiple test cases. Each case contains a single line containing a time **H** (5:00 ≤ **H** ≤ 9:00) that Bino woken up. The input ends with the final file (EOF).

Output

For each test case, print "Atraso maximo: **X**" (without quotes), **X** indicates the maximum delay (in minutes) of Bino in the encounter with Cino.



Input Sample	Output Sample
7:10 5:00	Atraso maximo: 10 Atraso maximo: 0

Pseudo código

1. Leer hora
2. Dividir cadena [0 = hh, 1 = mm],
3. Hallar retraso = ((hh+1)*60 + mm)-480
4. Evaluar si retraso menor a 0:
    a. Imprimir “Atraso máximo: 0”
5. Si no
    a. Imprimir “Atraso máximo: ”+retraso

Código

```
import java.util.Scanner;
public class Main {

    static Scanner s = new Scanner(System.in);

    public static void main(String[] args) {
        System.out.println("Atraso maximo: "+sundayMonday(s.next()));
    }
    private static int sundayMonday(String hm) {
        String[] hora = hm.split(":");
        Int hh = (((Integer.parseInt(hora[0])+1)*60) +
(Integer.parseInt(hora[1]))) - 480;
        if (hh < 0) {
            return 0;
        } else {
            return hh;
        }
    }
```

ALGORITMO 7: Identifying Tea

Blind tea tasting is the skill of identifying a tea by using only your senses of smell and taste.

As part of the Ideal Challenge of Pure-Tea Consumers (ICPC), a local TV show is organized. During the show, a full teapot is prepared and five contestants are handed a cup of tea each. The participants must smell, taste and assess the sample so as to identify the tea type, which can be: (1) white tea; (2) green tea; (3) black tea; or (4) herbal tea. At the end, the

answers are checked to determine the number of correct guesses.

Given the actual tea type and the answers provided, determine the number of contestants who got the correct answer.

Input

The first line contains an integer **T** representing the tea type (1 ≤ **T** ≤ 4). The second line contains five integers **A, B, C, D** and **E**, indicating the answer given by each contestant (1 ≤ **A, B, C, D, E** ≤ 4).

Output

Output a line with an integer representing the number of contestants who got the correct answer.

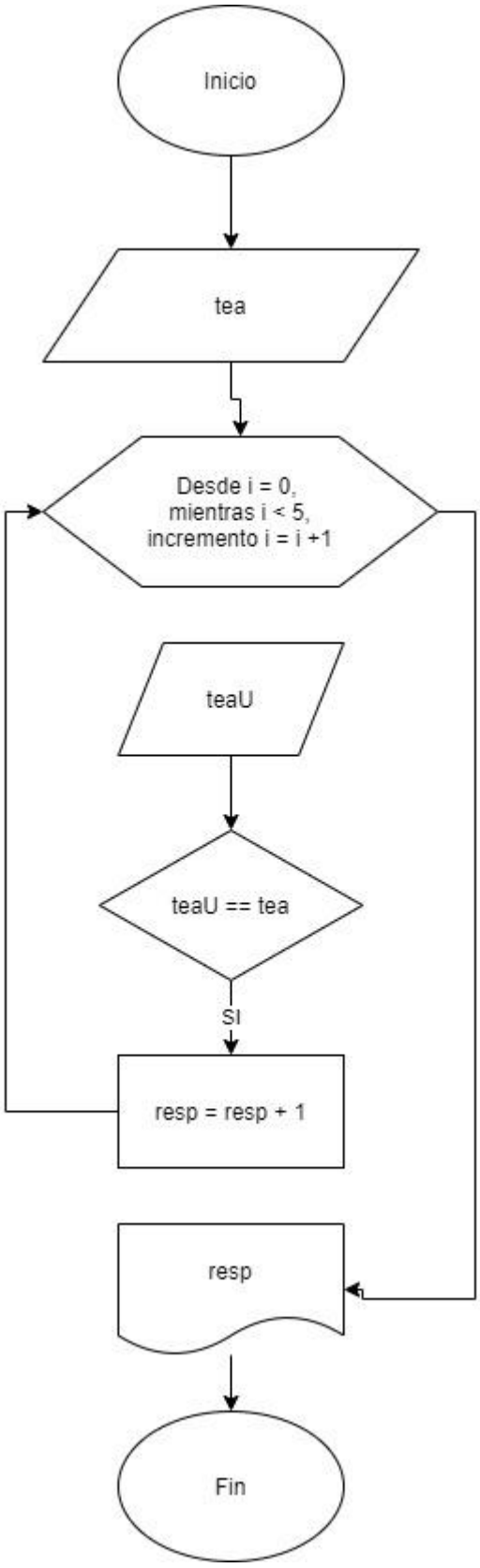
Input Samples	Output Samples
1 1 2 3 2 1	2
3 4 1 1 2 1	0

Pseudo código

1. Leer teaElegido
2. Con i = 0, mientras i sea menor a 5 y aumentado a i en 1:
    a. Leer teaUsuario
    b. Si teaUsuario == teaElegido
        i. Resuladop invrementa en 1.
3. Imprimir Resultado.

Código

```
import java.util.Scanner;
public class Main {
    static Scanner s = new
Scanner(System.in);
```



```
public static void main(String[] args) {
    System.out.println(identifyingTea(s.nextInt()));
}
private static int identifyingTea(int tea){
    int resultado = 0;
    for(int i = 0; i < 5; i++){
        if (s.nextInt() == tea){
            resultado++;
        }
    }
    return resultado;
}
}
```

ALGORITMO 8: El Desafío de Bino

Bino y Cino son amigos cercanos. A Bino le gusta crear desafíos matemáticos para que Cino los resuelva. Esta vez, Bino creó una lista de números y le preguntó a Cino: ¿Cuantos números son múltiplos de **2**, **3**, **4** y **5**?

Este desafío parece simple, pero si la lista contiene muchos números, Cino cometerá algunos errores de cálculo. Para ayudar a Cino, haga un programa que resuelva el Desafío de Bino.

Entrada

La primera línea de entrada consiste de un entero **N** ($1 \leq N \leq 1000$), que representa la cantidad de números en la lista de Bino.

La segunda línea contiene **N** enteros L_i ($1 \leq L_i \leq 100$), que representan los números en la lista de Bino.

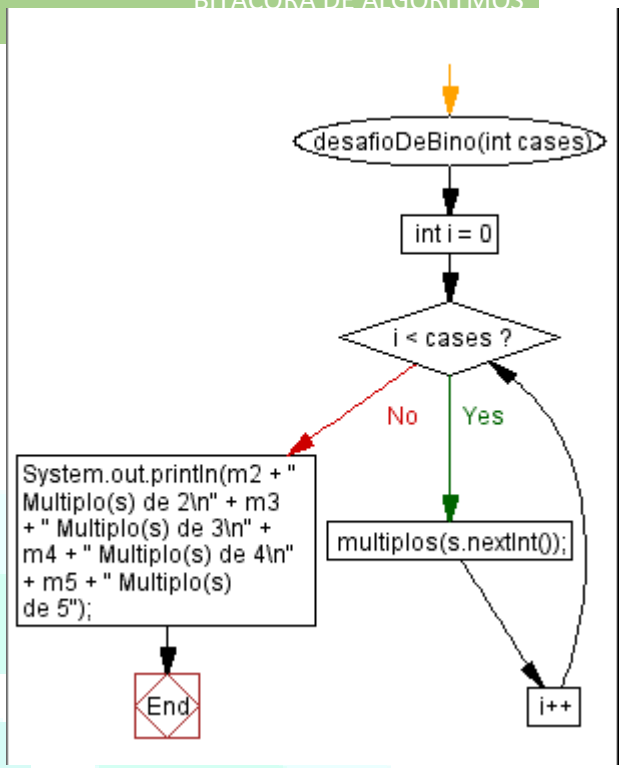
Salida

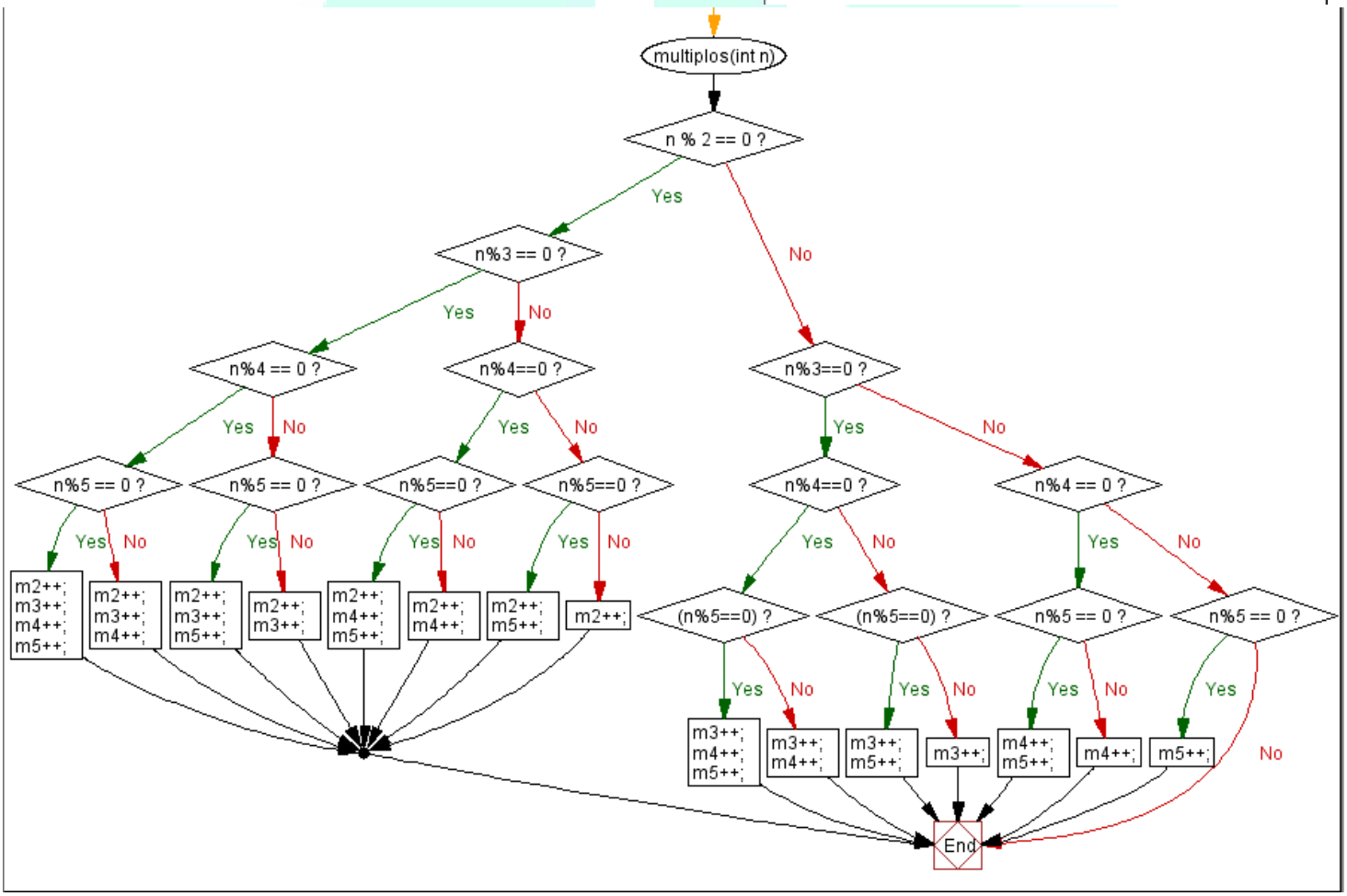
Mostrar la cantidad de múltiplos de **2**, **3**, **4** y **5** en la lista. Nótese que el formato de la salida mostrado en el ejemplo debe ser seguido estrictamente.

Ejemplos Entrada	Ejemplos Salida
5 2 5 4 20 10	4 Multiplo(s) de 2 0 Multiplo(s) de 3 2 Multiplo(s) de 4 3 Multiplo(s) de 5

DIAGRAMAS DE FLUJO (HECHOS CON VISTIN V8.07 DEMO)

1. Diagrama de flujo del metodo para hallar los multiplos de un numero
2. Diagrama de recoleccion de numeros e impresión de resultados (entrada el valor de casos del usuario)

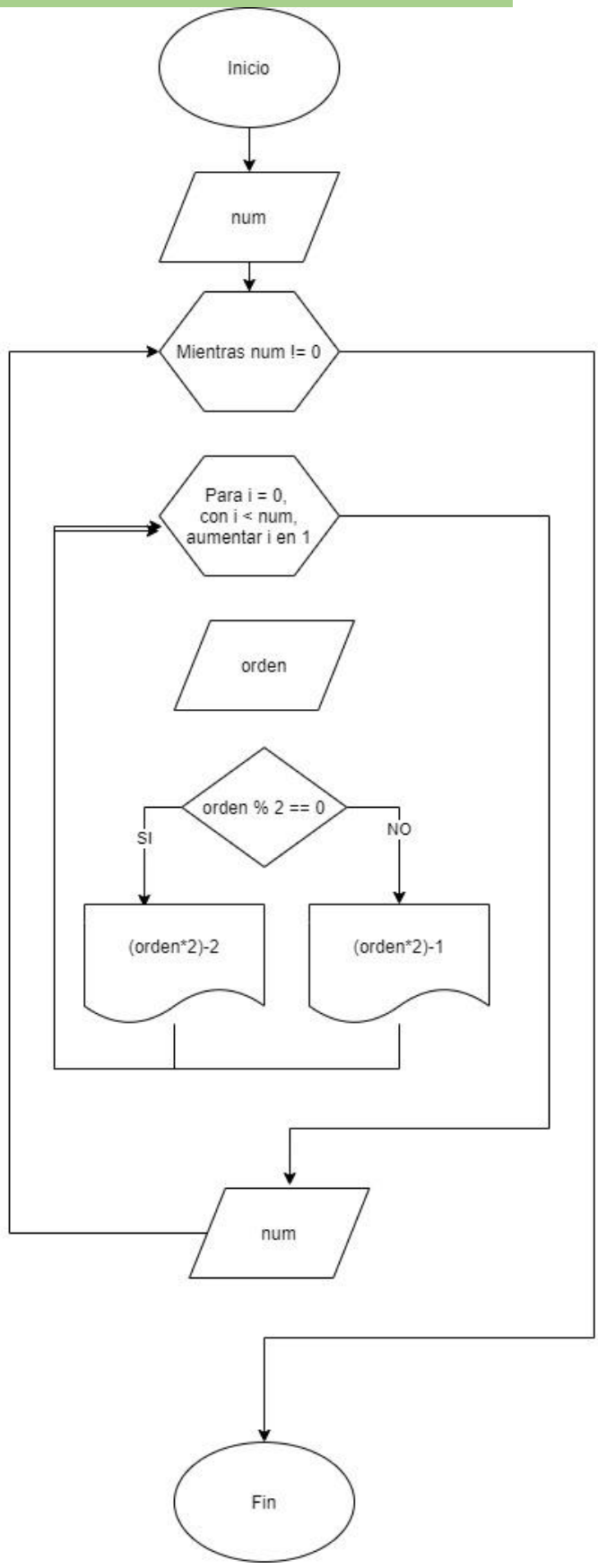




Código

```
import java.util.Scanner;
public class Main {

    static Scanner s = new Scanner(System.in);
    static int m2 = 0, m3 = 0, m4 = 0, m5 = 0;

    public static void main(String[] args) {
        desafioDeBino(s.nextInt());
    }
    private static void desafioDeBino(int cases) {

        for (int i = 0; i < cases; i++) {
            multiplos(s.nextInt());
        }
        System.out.println(m2 + " Multiplo(s) de 2\n"
                + m3 + " Multiplo(s) de 3\n"
                + m4 + " Multiplo(s) de 4\n"
                + m5 + " Multiplo(s) de 5");
    }

    private static void multiplos(int n) {
        if (n % 2 == 0) {
            if (n%3 == 0){
                if (n%4 == 0){
                    if (n%5 == 0){
                        m2++;m3++;m4++;m5++;
                    } else{
                        m2++;m3++;m4++;
                    }
                } else if (n%5 == 0){
                    m2++;m3++;m5++;
                } else{
                    m2++;m3++;
                }
            } else if(n%4==0){
                if(n%5==0){
                   m2++;m4++;m5++;
                } else {
                    m2++;m4++;
                }
            } else if (n%5==0){
                m2++;m5++;
            } else {
                m2++;
            }
        } else if(n%3==0){
            if (n%4==0){
                if ((n%5==0)){
                    m3++;m4++;m5++;
                } else{
                    m3++;m4++;
                }
```

```
            } else if((n%5==0)){
                m3++;m5++;
            } else{
                m3++;
            }
        } else if(n%4 == 0){
            if(n%5 == 0){
                m4++;m5++;
            }else {
                m4++;
            }
        } else if(n%5 == 0){
            m5++;
        }
    }
}
```



ALGORITMO 9: The Return of Radar

Every year after the contest that takes place in Taxiland, contestants and coaches go to the famous and renowned restaurant Radar. However, the waiters (always very kind and polite) get overwhelmed due to the amount of people, and therefore, end up taking a while to serve them.

Contestants or coaches who sit at the ends are the lucky ones, because they order only once and are served right away, but the others must always order twice, because the waiters (always very kind and polite) are careless and easily forget the orders. Besides, there's a superstition going on among the contestants and coaches that if there's not an even amount of people who don't sit at the ends, none of the university's teams will win the next contest.

So your task is to determine the sum of the number of of orders of each one to know if it's worth going to Radar. But whatever the result is, remember: it's always worth going to Radar!

Input

The input consists of the integer **T** ($1 \leq \mathbf{T} \leq 100$) indicating the number of test cases and then, **T** integers **N** ($3 \leq \mathbf{N} \leq 10^4$) indicating the number of people. The table is rectangular and there will be at least and at most one person at one end, i.e., if an end is empty, the other must be

occupied, otherwise, the two ends must be occupied, but the number of people that are not at the ends must always be even. Read input until T = 0.

Output

Print the sum of the number of orders of each person. There's no newline between the test cases.

Input Sample	Output Sample
5 13 37 49 11 9 2 5 17 0	25 73 97 21 17 9 33

Código

```
import java.util.Scanner;
public class Main {
    static Scanner s = new Scanner(System.in);

    public static void main(String[] args) {
        int num = s.nextInt(), ord = 0;
        while (num != 0) {
            for (int i = 0; i < num; i++) {
                radarRule(s.nextInt());
            }
            num = s.nextInt();
        }
    }


    private static void radarRule(int ord) {
        if (ord % 2 == 0) {
            System.out.println((ord * 2) - 2);
        } else {
            System.out.println((ord * 2) - 1);
        }
    }
}
```

ALGORITMO 10: LEDS

John quiere configurar un panel que contenga diferentes números con LEDs. No posee muchos LEDs y no está seguro de si él será capaz de montar el número deseado. Considerando la configuración de los

LEDs de los siguientes números, realice un algoritmo que le ayude a descubrir el número de LEDs necesarios para establecer el valor.

Entrada

La entrada contiene un entero **N**, **(1 ≤ N ≤ 2000)** correspondiente al número de casos de prueba, seguido por **N** líneas, cada línea que contiene un número **(1 ≤ V ≤ 10^{100})** correspondiente al valor que John quiere establecer con el Leds.

Salida

Para cada caso de prueba, imprima una línea que contenga el número de LEDs que John necesita para establecer el valor deseado, seguido por la palabra "leds".

Ejemplos de Entrada	Ejemplos de Salida
3 115380 2819311 23456	27 leds 29 leds 25 leds

Código

```
import java.util.Scanner;

public class Main {

    static int[] led = {6, 2, 5, 5, 4, 5, 6, 3, 7, 6};
    static Scanner s = new Scanner(System.in);

    public static void main(String[] args) {
        leds(s.nextInt());
    }

    public static void leds(int cases) {
        String c = "";
        while (cases > 0) {
            int numLed = 0;
            String x = s.next();
```

```
            for (int i = 0; i < x.length(); i++) {
                for (int j = 0; j < led.length; j++) {
                    if (x.charAt(i) == String.valueOf(j).charAt(0)) {
                        numLed += led[j];
                    }
                }
            }
            c += numLed + " leds\n";
            cases--;
        }
        System.out.println(c);
    }
}
```

ALGORITMO 11: Compare Substring

Find the longest common substring between the two informed Strings. The substring can be any part of the String, including the entire String. If there is no common substring, return 0. The search is *case sensitive* ('x' != 'X').

Input

The input contains several test cases. Each test case is composed by two lines that contains a string each. Both input Strings will contain between 1 and 50, inclusive, letters (a-z, A-Z), and/or spaces.

Output

The length of the longest common substring between the two Strings.

Sample Input	Sample Output
abcdef cdofhij TWO FOUR abracadabra open Hey This java is hot Java is a new paradigm	2 1 0 7

Código

```
import java.util.Scanner;
public class Main {
    static Scanner s = new Scanner(System.in);

    public static void main(String[] args) {
        compareStrings(s.nextLine().toLowerCase(), s.nextLine().toLowerCase());
    }
    public static void compareStrings(String cadena1, String cadena2) {
        String aux = "";
```

```
        int longitud = 0;
        for (int i = 0; i < cadena1.length(); i++) {
            aux += cadena1.charAt(i);
            if (cadena2.contains(aux)) {
                if (aux.length() > longitud) {
                    longitud = aux.length();
                }
                for (int j = i + 1; j < cadena1.length(); j++) {
                    aux += cadena1.charAt(j);
                    if (cadena2.contains(aux)) {
                        if (aux.length() > longitud) {
                            longitud = aux.length();
                        }
                    } else {
                        break;
                    } }
            } else {
                aux = "";
            } }
        System.out.println(longitud);
    }}
```

ALGORITMO 12: Internship

Googlbook is a famous IT company that opened an office in your town this year! Also, *Googlbook* has just offered interviews to an internship position in the company!

To be interviewed, you need to send some personal information to the company, that will be used to decide who will earn the position. You sent all information they need except one: your API (Academic Performance Index). To get things worse, *Student's Portal*, the system that provide your API, is not working!

Fortunately, you remember all the grades you got in all **M** subjects you coursed, as well their workloads. You also remember how the API is calculated:

$$\frac{\sum_{i=1}^{M}(N_i \times C_i)}{(\sum_{i=1}^{M} C_i) \times 100}$$

, where **N1**, **N2**, ..., **NM** are your grades in each subject, and **C1**, **C2**, ..., **CM** are the workload of the respective subjects.

Given the grades you got and the workload of each subject, determine your API, so you can send it to Googlbook as soon as possible!

Input

The input contains several test cases. The first line of each test case contains integer $\mathbf{M}$ ($1 \leq \mathbf{M} \leq 40$), the number of subjects you coursed. Each of the next $\mathbf{M}$ lines describe a subject. Each line contains two integers $\mathbf{Ni}$ and $\mathbf{Ci}$ ($0 \leq \mathbf{Ni} \leq 100$, $30 \leq \mathbf{Ci} \leq 120$), indicating the grade you got in that subject and its workload, respectively.

The input ends with end-of-file (EOF).

Output

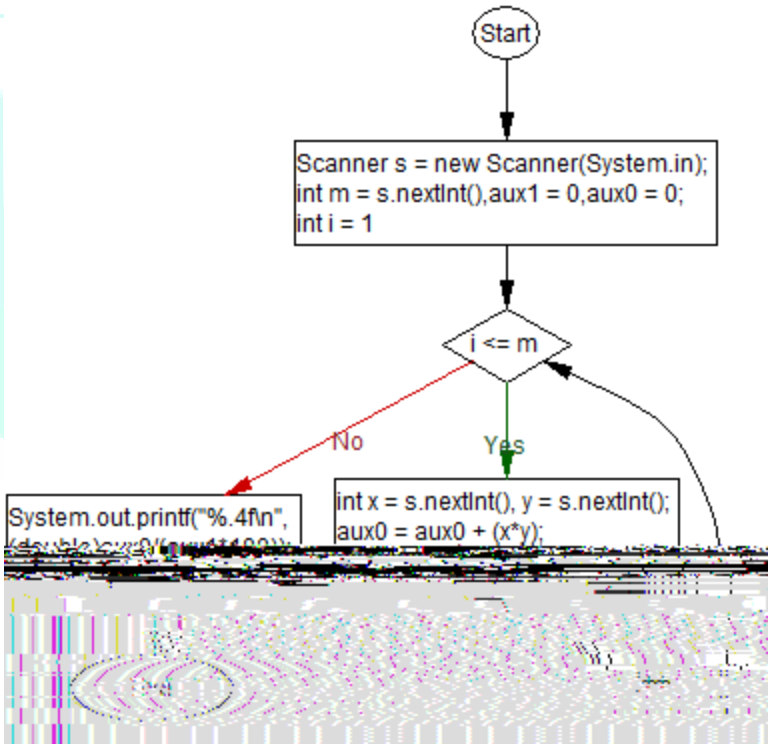
For each test case, print a line containing your API. Round and print it with exactly 4 decimal places.

Input Sample	Output Sample
3 70 60 90 60 80 120	0.8000

CODIGO

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int m = s.nextInt(),aux1 = 0,aux0 = 0;
        for (int i = 1; i <= m;i++ ) {
            int x = s.nextInt(), y = s.nextInt();
            aux0 = aux0 + (x*y);
            aux1 = aux1 + y;
        }
    System.out.printf("%.4f\n
",(double)aux0/(aux1*100));
    }
}
```

DIAGRAMA DE FLUJO --→



ALGORITMO 13: Web Browser

Lucas is a pretty radical guy when it comes to software licenses. Since beginning his undergraduate degree in computer engineering, he seeks to develop all the tools he needs. All this started after bad experiences

using proprietary software. Now he believes that a real programmer must be self-sufficient, that is, he must build all the programs he needs, from a simple calculator to his own operating system.

This semester, Lucas is studying the web systems development course. To continue his philosophy of life, using only software he built himself, Lucas is already programming his own web browser. Much of the work has been completed, but some functionality still needs to be finished.

Lucas' browser has a search field where the user can enter a keyword, and clicking a confirmation button it will redirect to another page with the results of his search. When some string is entered in the search field, Lucas wants his program to display, below, some options to auto complete this string according to the searches already performed by the user.

For example, if the words "algoritmos" and "algas" have already been searched, when typing the string "alg", the program should suggest the words "algorithms" and "algas". Therefore, for each string entered, the program should suggest previously searched words prefixed with this string. If any word is equal to the typed string, it should also be suggested.

Lucas is concerned about the amount of words his program can suggest, and the maximum size they can reach. So he asked you to help him by writing a program where given a few words already searched and a series of queries composed of a string, indicate how many words the browser should suggest to the user, and the length of the largest of these words.

Input

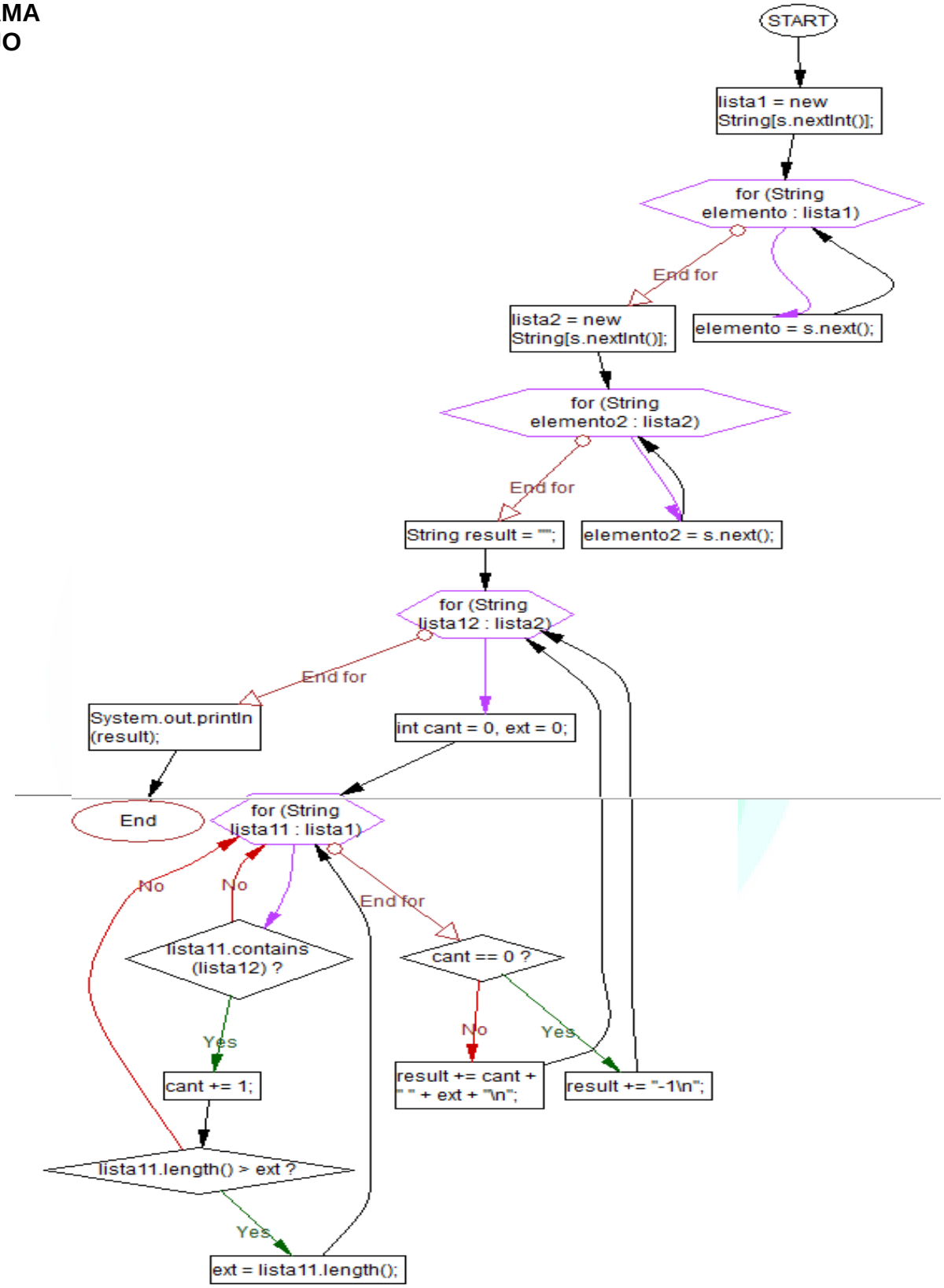
The input is composed of several test cases. The first line of a test case has an integer **N** (1 ≤ **N** ≤ 10^4) indicating the number of words that have already been searched by the Lucas' program. Each of the next **N** lines contains a nonempty string of a maximum of 100 lowercase letters [a - z]. For each test case, **N** words are different. Then there will be an integer **Q** (1 ≤ **Q** ≤ 100) indicating the number of queries. Each of the next **Q** lines describes a query with a non-empty string of a maximum of 100 lowercase letters [a - z], representing a string entered in the search field.

Output

For each test case print **Q** lines, where the i-th row is the answer to the i-th query. The response of each query should be composed of two integers separated by space, representing, respectively, the number of words suggested by the program when entering the string indicated by the ith query, and the length of the largest word contained in that subset. If no words are suggested, print -1. Print a blank line at the end of each test case.

Input Sample	Output Sample
5 maratonaicpc maraton programacao progress inputs 3 marat programacao outputs	2 12 1 11 -1

DIAGRAMA DE FLUJO



CODIGO

```
public class Main {

    static Scanner s = new Scanner(System.in);
    static String[] lista1;
    static String[] lista2;
    public static void main(String[] args) {
        lista1 = new String [s.nextInt()];
        llenarLista(lista1);
        lista2 = new String [s.nextInt()];
        llenarLista(lista2);
        System.out.println(resultado());
    }
    private static void llenarLista(String[] lista) {
        for (int i = 0; i < lista.length;i++){
            lista[i] = s.next();
        }
    }
    private static String resultado() {
        String result = "";
        for (String lista12 : lista2) {
            int cant = 0, ext = 0;
            for (String lista11 : lista1) {
                if (lista11.contains(lista12)) {
                    cant +=1;
                    if (lista11.length() > ext){
                        ext = lista11.length();
                    }
                }
            }
            if (cant == 0){
                result += "-1\n";
            } else {
                result += cant+" "+ext+"\n";
            }
        }
        return result;
    }}
```

ALGORITMO 14: Magic and Sword

In the Magic and Sword Tower defense, the player can cast area spells to defeat the enemy units. The spells are elemental: fire, water, air and earth, and the affected region is determined by a circle whose radius depends on the level of the spell.

The table below lists each spell, damage and its radius per level:

Magia	Descrição	Dano	Level 1	Level 2	Level 3
Fire	Um meteoro é conjurado para eliminar as unidades inimigas com o calor do fogo	200	20	30	50
Water	Uma *tsunami* arrasta os inimigos com a força da água	300	10	25	40
Earth	Um terremoto abala as unidades inimigas, causando dano massivo	400	25	55	70
Air	Um tornado mina as forças inimigas com a velocidade do ar	100	18	38	60

The enemy units are delimited by a rectangle of width **w** and height **h**, with the lower left corner positioned at the point (x0, y0). The enemy will suffer damage if their bounding rectangle has any intercession with the area defined by the spell circle.

Given the position and the bounding rectangle of the enemy unit, the center of the explosion, the identifier and level of the spell, determine the damage to the unit. If the unit is out of the spell range, the damage is equal to zero.

Input

The input consists of **T** (1 ≤ **T** ≤ 1000) test cases, where the value of **T** is reported in the first line of the input. Each test case consists of two lines. The first contains four integers representing the dimensions **w** and **h** (1 ≤ **w**, **h** ≤ 1000) of the rectangle and the coordinates **x0** and **y0** (0 ≤ **x0**, **y0** ≤ 1000) from the lower left corner. The second line of the test case contains a string with the spell identifier (fire, water, earth and air), the level **N** of this spell (1 ≤ **N** ≤ 3) and the coordinates **cx** e **cy** (0 ≤ **cx**, **cy** ≤ 1000) from the center of the explosion area.

Output

For each test case, the output must be the value of the damage received by the unit, followed by a line break.

Input Sample	Output Sample
4 10 10 50 50 fire 1 85 55 10 10 50 50 fire 2 85 55 10 10 50 100 water 3 100 100 10 10 50 100 air 3 100 100	0 200 300 100

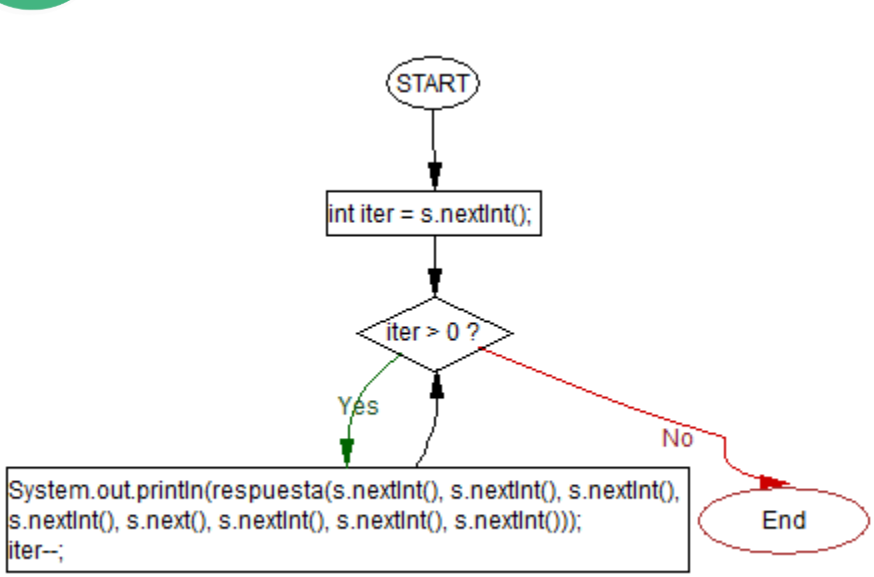
CODIGO

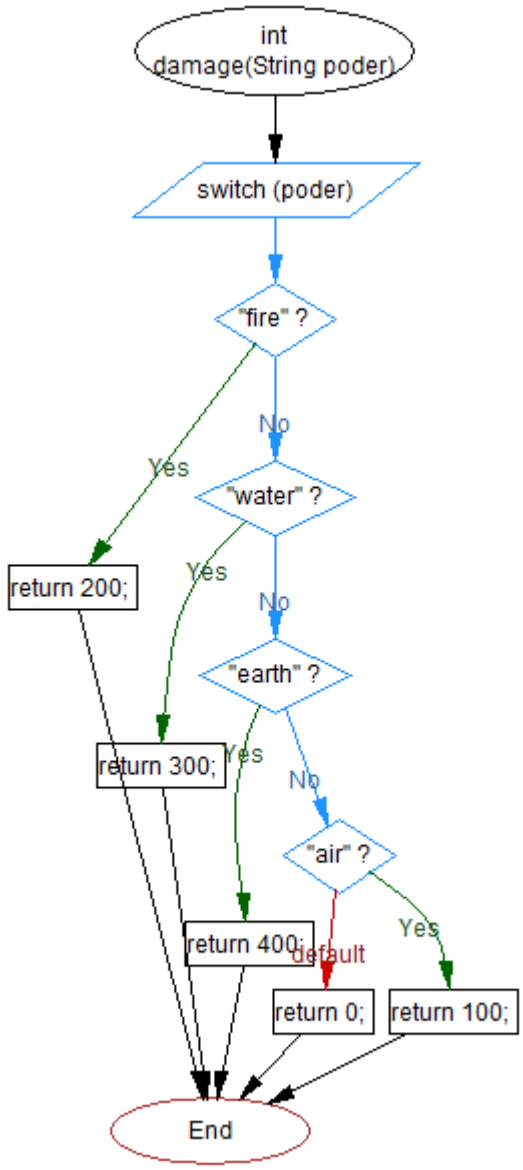
```
import java.util.Scanner;

public class Main {

    static Scanner s = new Scanner(System.in);
    static int[] rec = new int[4];
    static int[] cir = new int[4];

    public static void main(String[] args) {
        int iter = s.nextInt();
        while (iter > 0) {
            System.out.println(respuesta(s.nextInt(), s.nextInt(), s.nextInt(),
s.nextInt(), s.next(), s.nextInt(), s.nextInt(), s.nextInt()));
            iter--;
        }
    }

    private static void dibujarcuadrado(int w, int h, int x, int y) {
        rec[0] = x + w;
        rec[1] = x;
        rec[2] = y;
        rec[3] = y + h;

    }

    private static void dibujarCirculo(String poder, int level, int x, int y) {
        int r = radio(poder, level);
        cir[0] = x - r;
        cir[1] = x + r;
        cir[3] = y + r;
        cir[2] = y - r;
    }

    private static int radio(String poder, int level) {
        switch (poder) {
            case "fire":
                switch (level) {
                    case 1:
                        return 20;
                    case 2:
                        return 30;
                    case 3:
                        return 50;
                }
                break;
            case "water":
                switch (level) {
                    case 1:
                        return 10;
                    case 2:
```

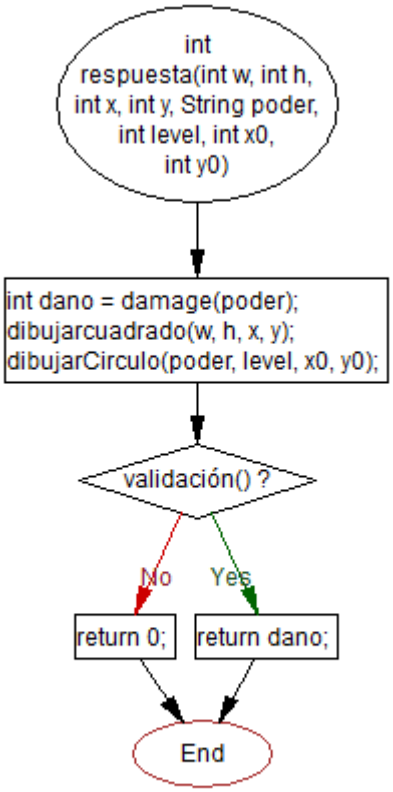
```
                    return 25;
                case 3:
                    return 40;
            }
            break;
        case "earth":
            switch (level) {
                case 1:
                    return 25;
                case 2:
                    return 55;
                case 3:
                    return 70;
            }
            break;
        case "air":
            switch (level) {
                case 1:
                    return 18;
                case 2:
                    return 38;
                case 3:
                    return 60;
            }
            break;
    }
    return 0;
}

    private static int respuesta(int w, int h, int x, int y, String poder, int level, int
x0, int y0) {
        int dano = damage(poder);
        dibujarcuadrado(w, h, x, y);
        dibujarCirculo(poder, level, x0, y0);
        if (validación()) {
            return dano;
        }
        return 0;
    }

    private static int damage(String poder) {
        switch (poder) {
            case "fire":
                return 200;
            case "water":
                return 300;
            case "earth":
                return 400;
            case "air":
                return 100;
```

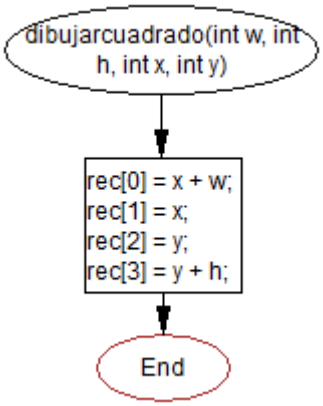
```
        }
        return 0;
    }

    private static boolean validación() {
        if ((rec[0] >= cir[0] && rec[0] <= cir[1]) || (rec[1] <= cir[1] && rec[1] >=
cir[0])) {
            if ((rec[2] >= cir[2] && rec[2] <= cir[3]) || (rec[3] <= cir[2] && rec[3] >=
cir[3])) {
                return true;
            }
        }
        return false;
    }
}
```

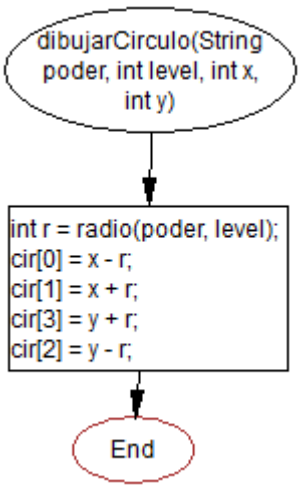
DIAGRAMAS DE FLUJO

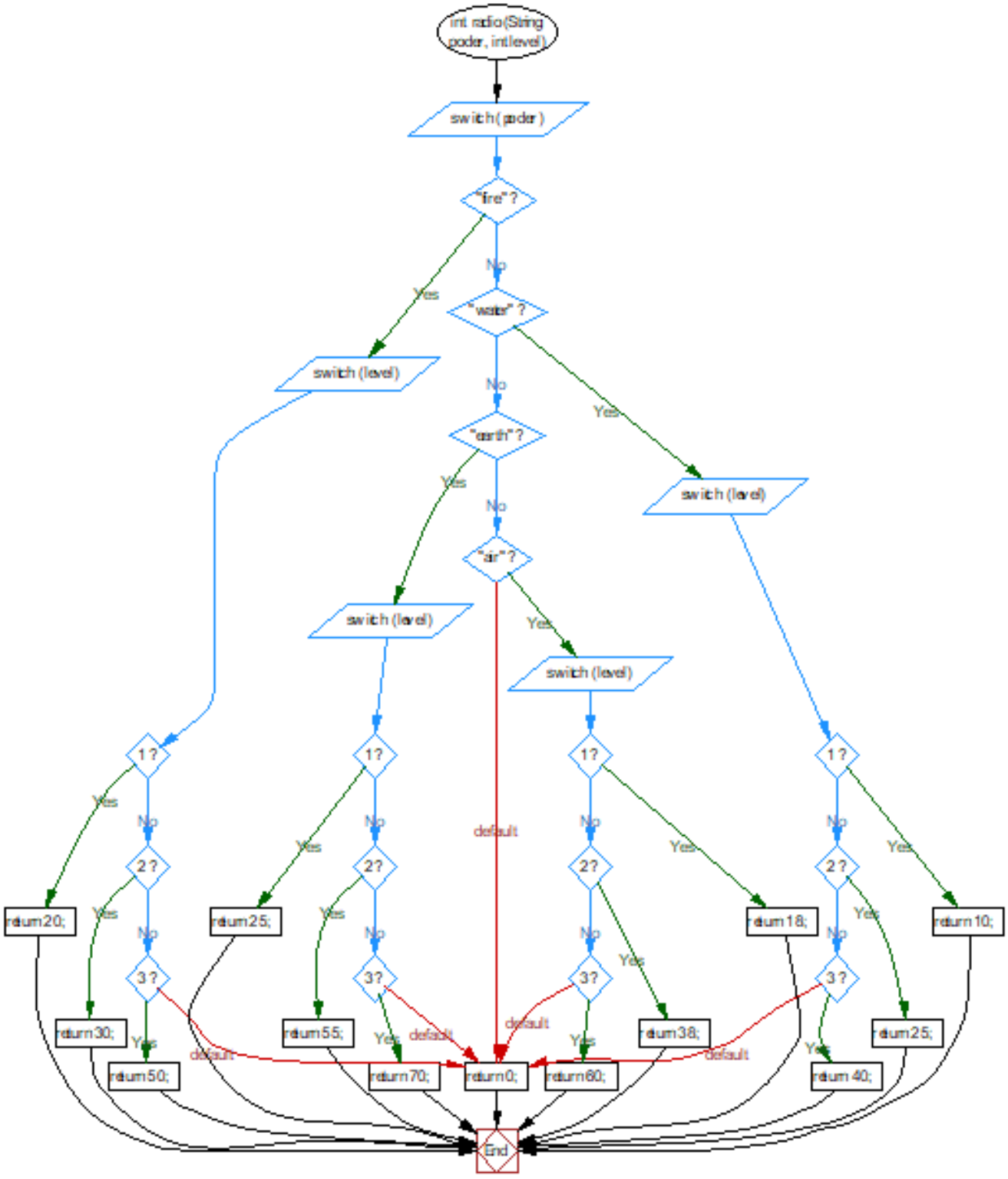

START
int iter = s.nextInt();
iter > 0 ?
Yes
No
System.out.println(respuesta(s.nextInt(), s.nextInt(), s.nextInt(), s.nextInt(), s.next(), s.nextInt(), s.nextInt(), s.nextInt()));
iter--;
End

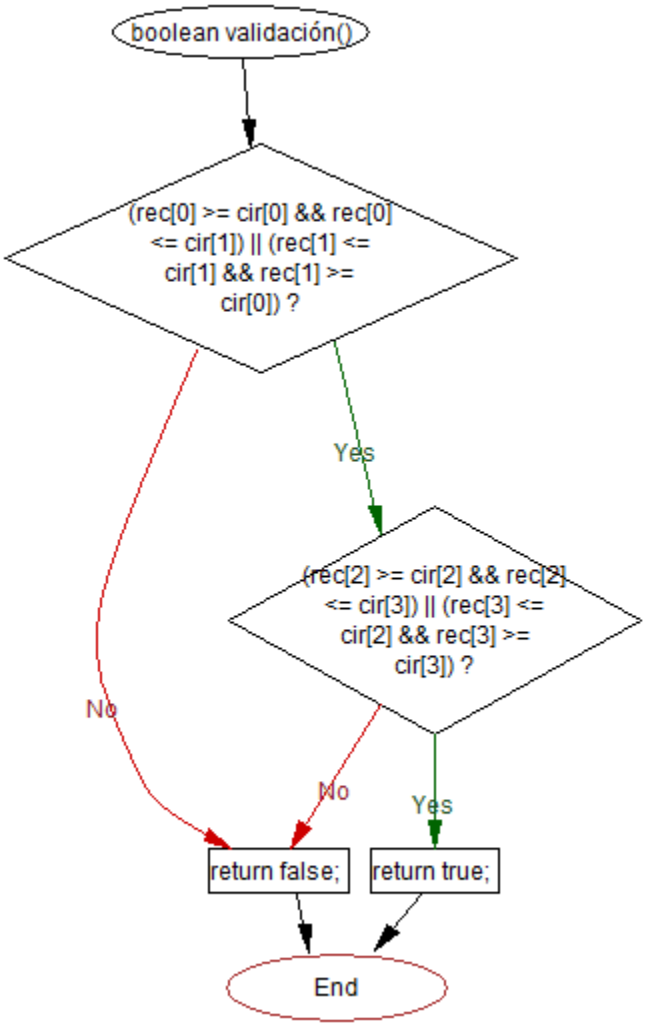

int damage(String poder)
switch (poder)
"fire" ?
Yes
No
return 200;
"water" ?
Yes
No
return 300;
"earth" ?
Yes
No
return 400;
"air" ?
default
Yes
return 0;
return 100;
End


int respuesta(int w, int h, int x, int y, String poder, int level, int x0, int y0)
int dano = damage(poder);
dibujarcuadrado(w, h, x, y);
dibujarCirculo(poder, level, x0, y0);
validación() ?
No
Yes
return 0;
return dano;
End


dibujarcuadrado(int w, int h, int x, int y)
rec[0] = x + w;
rec[1] = x;
rec[2] = y;
rec[3] = y + h;
End


dibujarCirculo(String poder, int level, int x, int y)
int r = radio(poder, level);
cir[0] = x - r;
cir[1] = x + r;
cir[3] = y + r;
cir[2] = y - r;
End


int radio(String poder, int level)
switch(poder)
"fire" ?
Yes
No
"water" ?
switch (level)
No
"earth" ?
Yes
No
Yes
switch (level)
"air" ?
switch (level)
Yes
switch (level)
1 ?
1 ?
1 ?
1 ?
Yes
No
Yes
No
default
No
Yes
No
Yes
2 ?
2 ?
2 ?
2 ?
Yes
No
Yes
No
Yes
No
Yes
No
return 20;
return 25;
return 18;
return 10;
3 ?
3 ?
3 ?
3 ?
Yes
No
Yes
No
Yes
No
Yes
No
return 30;
return 55;
return 38;
return 25;
default
default
default
default
Yes
Yes
Yes
Yes
return 50;
return 70;
return 0;
return 60;
return 40;
End



ALGORITMO 15: Domination Bacteriana

The Federal Institute of the South of Minas (IF), Muzambinho campus opened vacancies for the new superior course: Veterinary Medicine. The coordinator of the course already anticipated some problems with the studied bacteria, once that is inevitable that a bowl of bacteria falls to the ground. It would be difficult to control them if this situation happened because, for the application of the antidote it is necessary a proportionality, in addition, it must be thrown around the bacteria, killing from the outside to inside.

Beyond to an exact amount, the antidote has to be thrown around the bacteria, so the area of the bacteria needs to be calculated to the coordinator can apply the right dose. The area of a bacterium is given in a grid(x, y) and in it, the coordinator marks the edge of the bacterium represented by 1. Through this, the area of the bacterium is given by (perimeter (edge (1)) + content the inner side of the border or is the number of zeros surrounded by 1) divided by two, a simple average.

Because of your good reputation, she asked you to develop an application that solves this problem. Worth some extra points, develop the algorithm capable of calculating the area dominated by bacteria, which has very bizarre formats, outlined by the coordinator so that the application of the antidote has resulted.

Input

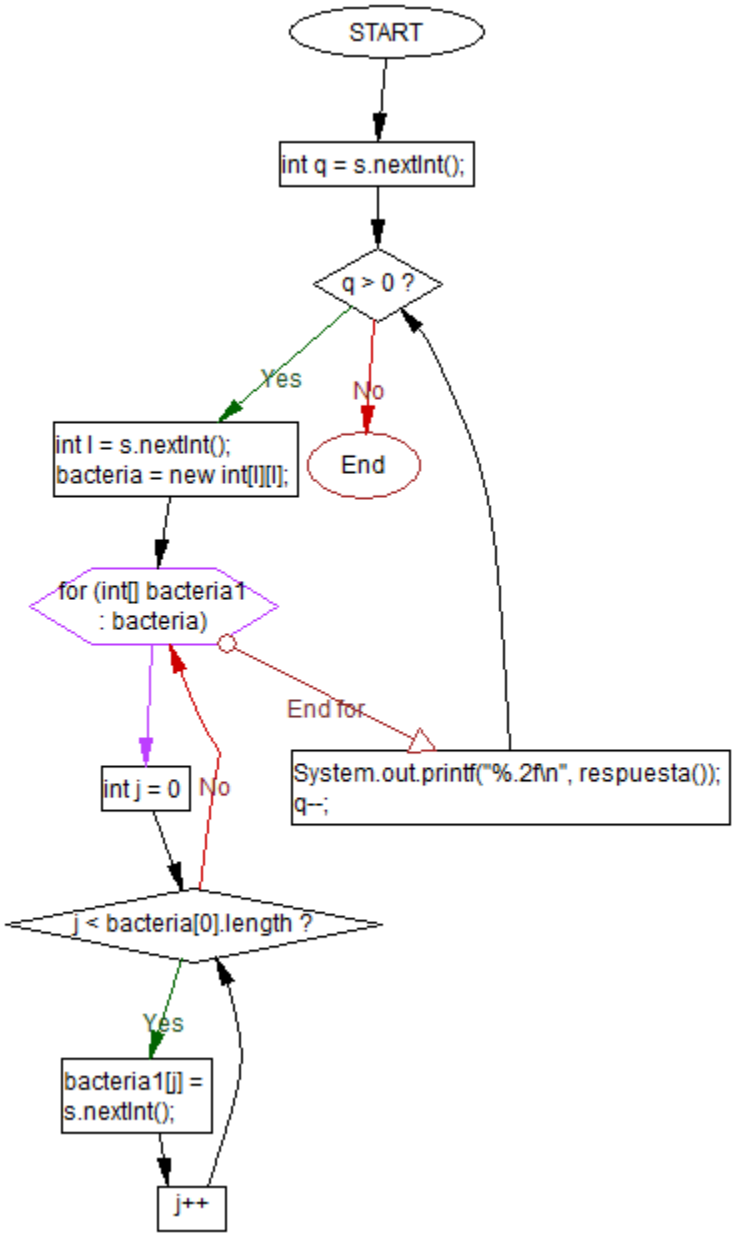
The input has Q (0 < Q <100) amounts of exposed bacteria, and then a small description of the area, composed of an integer L (0 < L <15) indicating the width and height of the grid and finally the grid LxL) with the edge drawing of the bacterium.

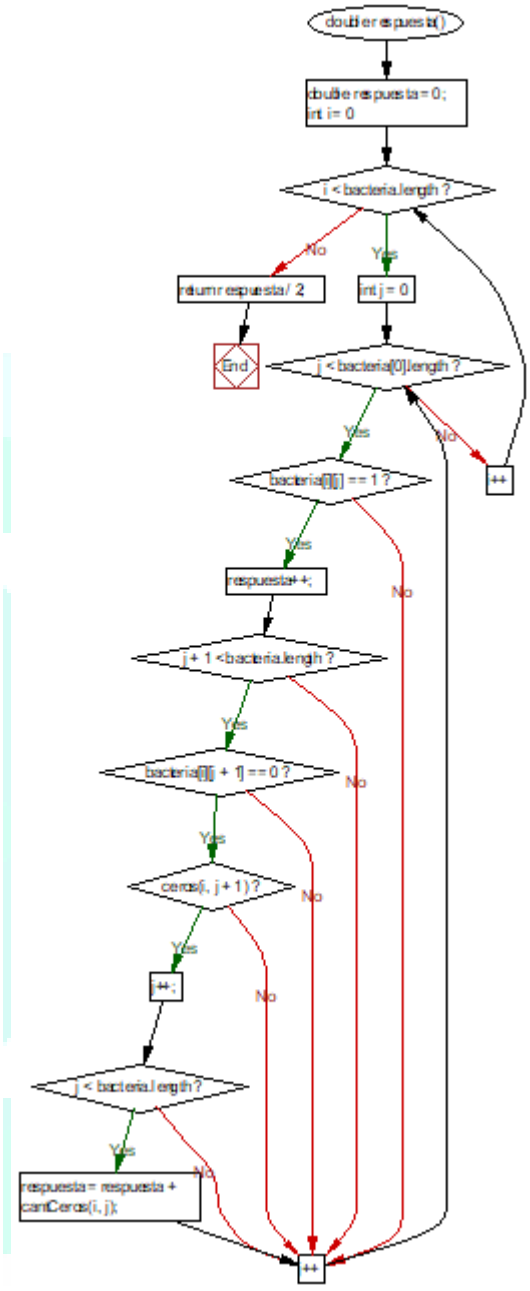
Output

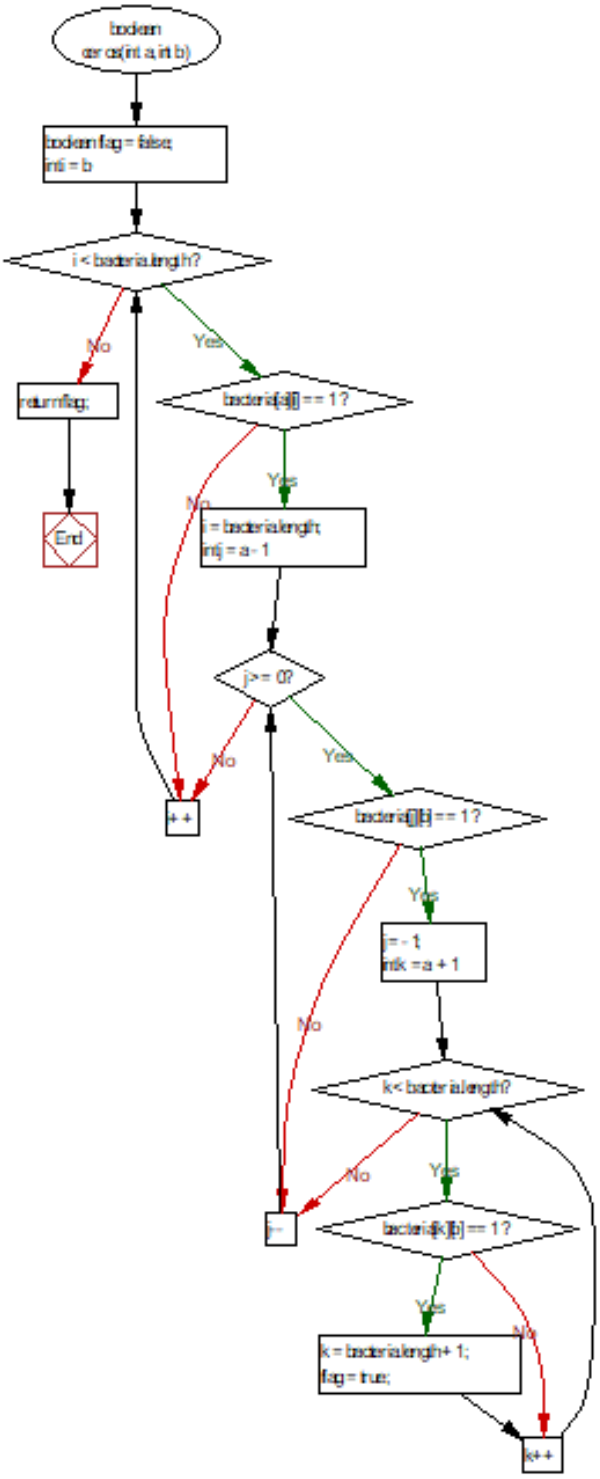
As informed, display the dominated area with two decimal places.

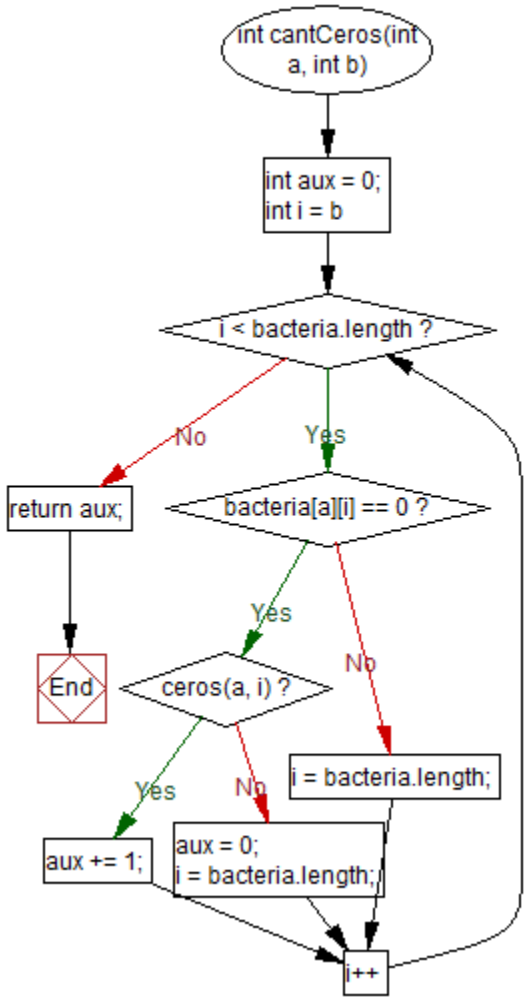
Input Sample	Output Sample
2 8 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 1 1 1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 1 1 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 9 1 0 0 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 0 0 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0 0 1 1 0 0 0 1 1 0 0 1 0 0 0 0 0 1 0 0 1 1 1 1 1 1 1 0 1 0 0 1 0 1 0 0 1	13.00 25.00

Diagramas de flujo






boolean
cer os(int a,int b)
boolean flag = false;
int i = b
i < bacteria.length?
No
Yes
return flag;
bacteria[a][i] == 1?
End
No
Yes
i = bacteria.length;
int j = a - 1
j >= 0?
No
Yes
+ +
bacteria[j][i] == 1?
Yes
j = - 1
int k = a + 1
No
k< bacteria.length?
No
Yes
-
bacteria[k][i] == 1?
Yes
No
k = bacteria.length+ 1;
flag = true;
k++


int cantCeros(int
a, int b)
int aux = 0;
int i = b
i < bacteria.length ?
No
Yes
return aux;
bacteria[a][i] == 0 ?
Yes
No
End
ceros(a, i) ?
Yes
No
i = bacteria.length;
aux += 1;
aux = 0;
i = bacteria.length;
i++

Código

```
import java.util.Scanner;

public class Main {

    static Scanner s = new Scanner(System.in);
    static int[][] bacteria;

    public static void main(String[] args) {
        int q = s.nextInt();
        while (q > 0) {
            int l = s.nextInt();
            bacteria = new int[l][l];
            for (int[] bacteria1 : bacteria) {
                for (int j = 0; j < bacteria[0].length; j++) {
                    bacteria1[j] = s.nextInt();
                }
            }
            System.out.printf("%.2f\n", respuesta());
            q--;
        }

    }

    private static double respuesta() {
        double respuesta = 0;
        for (int i = 0; i < bacteria.length; i++) {
            for (int j = 0; j < bacteria[0].length; j++) {
                if (bacteria[i][j] == 1) {
                    respuesta++;
                    if (j + 1 < bacteria.length) {
                        if (bacteria[i][j + 1] == 0) {
                            if (ceros(i, j + 1)) {
                                j++;
                                if (j < bacteria.length) {
                                    respuesta = respuesta + cantCeros(i, j);
                                }
                            }
                        }
                    }
                }
            }
        }
        return respuesta / 2;
    }

    private static boolean ceros(int a, int b) {
        boolean flag = false;
        for (int i = b; i < bacteria.length; i++) {
            if (bacteria[a][i] == 1) {
                i = bacteria.length;
                for (int j = a - 1; j >= 0; j--) {
                    if (bacteria[j][b] == 1) {
```

```
                        j = -1;
                        for (int k = a + 1; k < bacteria.length; k++) {
                            if (bacteria[k][b] == 1) {
                                k = bacteria.length + 1;
                                flag = true;
                            }
                        }
                    }
                }
            }
        }
        return flag;
    }

    private static int cantCeros(int a, int b) {
        int aux = 0;
        for (int i = b; i < bacteria.length; i++) {
            if (bacteria[a][i] == 0) {
                if (ceros(a, i)) {
                    aux += 1;
                } else {
                    aux = 0;
                    i = bacteria.length;
                }
            } else {
                i = bacteria.length;
            }
        }
        return aux;
    }
}
```

ALGORITMO 16: Dijkstra

In the game The Witcher, Sigismund Dijkstra is the leader of the Redanian Secret Service, because of this he is one of the most important people in the world.

In addition Dijkstra has a large treasure, which has several types of jewelry.

Dijkstra is very curious to know how many different types of jewelry his treasure has.

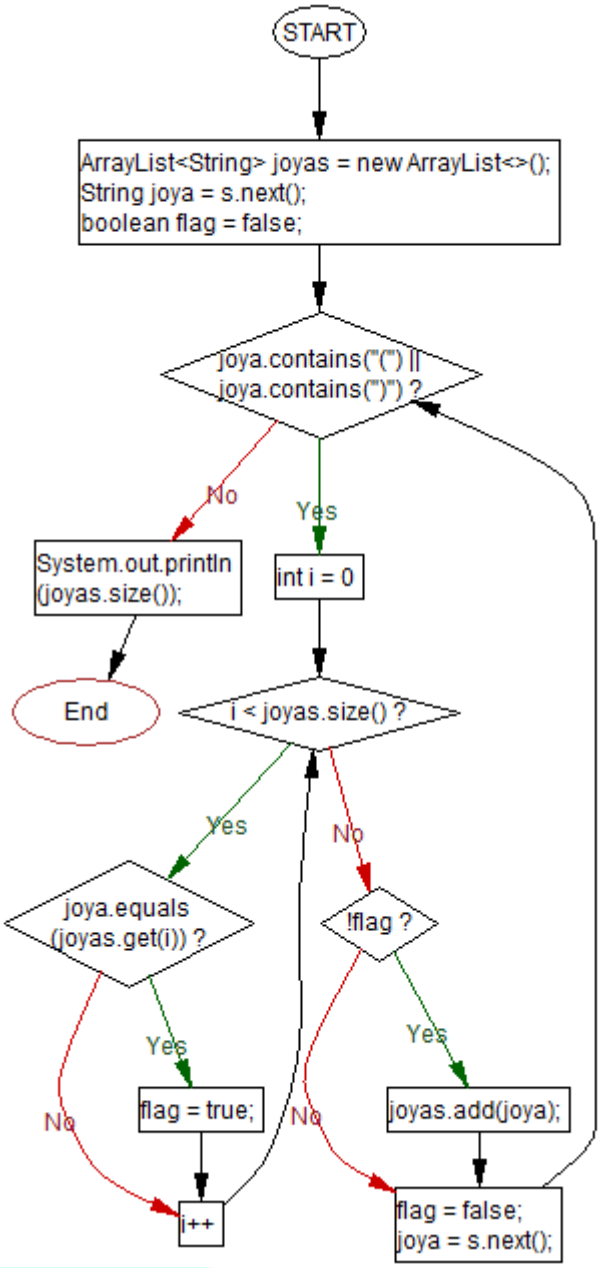
Knowing that you are the best programmer on the continent Dijkstra hired you to check how many different types of jewelry he has in his treasure.

Input

The entry consists of several lines and each contains a string describing one of Dijkstra's jewels. This string is composed only of the characters '(' and ')', the sum of the length of all the string does not exceed 10^6.

Output

Print how many different kinds of jewelry Dijkstra has.



Input Sample	Output Sample
(( )) (( )) (	3

CODIGO

```
import java.util.ArrayList;
import java.util.Scanner;

public class Main {

    static Scanner s = new Scanner(System.in);
    static ArrayList<String> joyas = new ArrayList<String>();

    public static void main(String[] args) {
        String joya = s.next();
        while (joya.contains("(") || joya.contains(")")) {
            addJoya(joya);
            joya = s.next();
        }
        System.out.println(joyas.size());
    }

    private static void addJoya(String joya) {
        if (!existsJoya(joya)) {
            joyas.add(joya);
        }
    }

    private static boolean existsJoya(String joya) {
        for (int i = 0; i < joyas.size(); i++) {
            if (joya.equals(joyas.get(i))) {
                return true;
            }
        }
        return false;
    }

}
```

ALGORITMO 17: LU DI OH!

Iu-di-oh! is a card game really popular among kids! Every *Iu-di-oh!* player has his own deck containing many cards. Each card contains **N** attributes (such as power, speed, smartness, etc.). Attributes are numbered from 1 to **N** and are given as positive integers.

A match of *Iu-di-oh!* is always played by two players. At the beginning of the match, each player chooses exactly one card from his deck. Then, an attribute is randomly chosen. The player whose the chosen attribute is greater in the card he choose wins the match. If the such attribute is equal in both cards, there is a tie.

Marcos and Leonardo are in the big final of the Brazilian *Iu-di-oh!* championship. The great prize is a Dainavision (that is almost as good as a Plaisteition 2!). Given the deck of both players, the card each one chooses and the chosen attribute, determine the winner!

Input

The input contains several test cases. The first line of each test case contains an integer **N** ($1 \leq \mathbf{N} \leq 100$), the number of attributes each card contains. The second line contains two integers **M** and **L** ($1 \leq \mathbf{M}, \mathbf{L} \leq 100$), the number of cards in Marcos' and Leonardo's deck, respectively.

Next **M** lines describe Marcos' deck. His cards are numbered from 1 to **M**, and *i*-th line describes the *i*-th card. Each line contains **N** integers $\mathbf{a}_{i,1}, \mathbf{a}_{i,2}, \ldots, \mathbf{a}_{i,N}$ ($1 \leq \mathbf{a}_{i,j} \leq 10^9$). Integer $\mathbf{a}_{i,j}$ indicates the *j*-th attribute of the *i*-th card.

Next **L** lines describe Leonardo's deck. His cards are numbered from 1 to **L** and are described in the same way as Marcos' deck.

Next line contains two integers $\mathbf{C_M}$ and $\mathbf{C_L}$ ($1 \leq \mathbf{C_M} \leq \mathbf{M}$, $1 \leq \mathbf{C_L} \leq \mathbf{L}$), the cards chosen by Marcos and Leonardo, respectively. Finally, the last line contains an integer **A** ($1 \leq \mathbf{A} \leq \mathbf{N}$) indicating the chosen attribute.

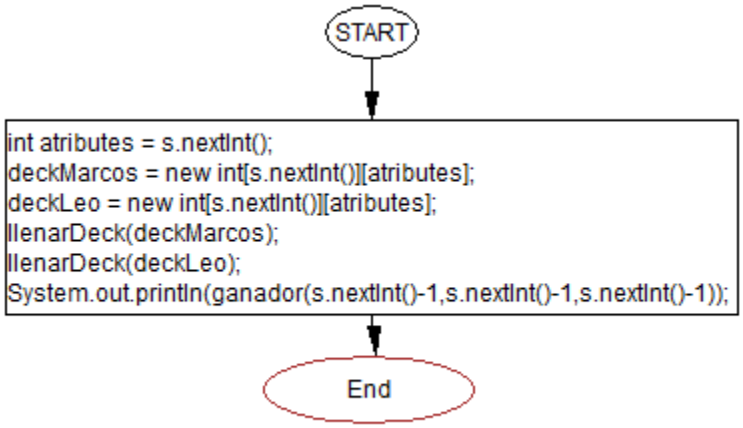
The input ends with end-of-file (EOF).

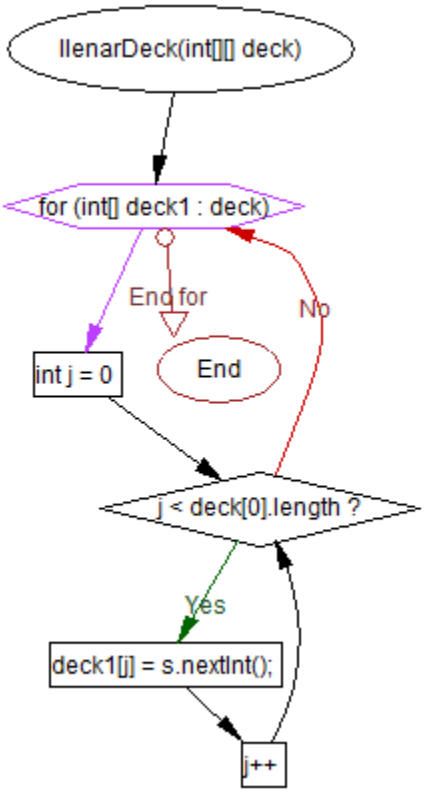
Output

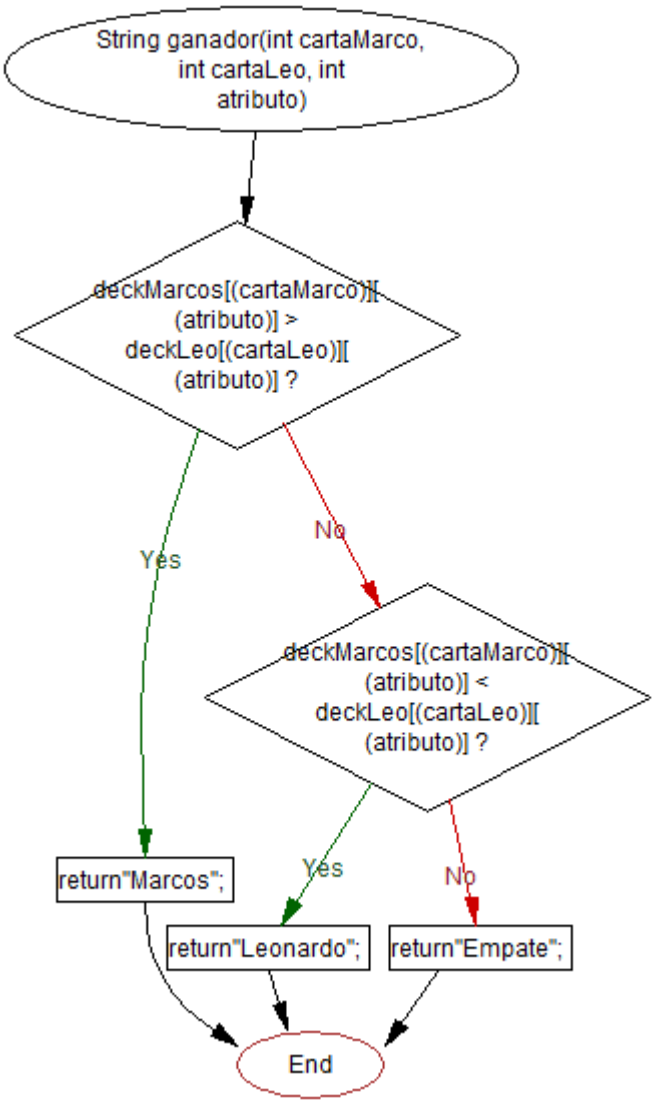
For each test case, print a line containing "Marcos" if Marcos wins the match, "Leonardo" if Leonardo wins the match, or "Empate" in the case of a tie (without quotes).

Input Sample	Output Sample
3 2 2 3 8 1 6 7 9 1 2 3 8 4 1 1 2 2	Marcos

DIAGRAMAS DE FLUJO


START
int atributes = s.nextInt();
deckMarcos = new int[s.nextInt()][atributes];
deckLeo = new int[s.nextInt()][atributes];
llenarDeck(deckMarcos);
llenarDeck(deckLeo);
System.out.println(ganador(s.nextInt()-1,s.nextInt()-1,s.nextInt()-1));
End


llenarDeck(int[][] deck)
for (int[] deck1 : deck)
End for
No
End
int j = 0
j < deck[0].length ?
Yes
deck1[j] = s.nextInt();
j++


String ganador(int cartaMarco, int cartaLeo, int atributo)
deckMarcos[(cartaMarco)][(atributo)] > deckLeo[(cartaLeo)][(atributo)] ?
Yes
No
deckMarcos[(cartaMarco)][(atributo)] < deckLeo[(cartaLeo)][(atributo)] ?
return"Marcos";
Yes
No
return"Leonardo";
return"Empate";
End

CODIGO

```
import java.util.Scanner;

public class Main {

    static Scanner s = new Scanner(System.in);
    static int[][] deckMarcos;
    static int[][] deckLeo;

    public static void main(String[] args) {
        int atributes = s.nextInt();
        deckMarcos = new int[s.nextInt()][atributes];
        deckLeo = new int[s.nextInt()][atributes];
        llenarDeck(deckMarcos);
        llenarDeck(deckLeo);
        System.out.println(ganador(s.nextInt()-1,s.nextInt()-1,s.nextInt()-1));
    }

    private static void llenarDeck(int[][] deck) {
        for (int[] deck1 : deck) {
            for (int j = 0; j < deck[0].length; j++) {
                deck1[j] = s.nextInt();
            }
        }
    }

    private static String ganador(int cartaMarco, int cartaLeo, int atributo) {
        if (deckMarcos[(cartaMarco)][(atributo)] > deckLeo[(cartaLeo)][(atributo)]){
            return"Marcos";
        } else if (deckMarcos[(cartaMarco)][(atributo)] < deckLeo[(cartaLeo)][(atributo)]){
            return"Leonardo";
        }else {
            return"Empate";
        }
    }
}
```