ADA - Análisis y Diseño de Algoritmos, 2026-1
Tarea 1: Semanas 1 y 2

Para entregar el domingo 15 de febrero de 2026

Problemas conceptuales a las 23:59 por BrightSpace

Problemas prácticos a las 23:59 en la arena de programación

Tanto los ejercicios como los problemas deben ser resueltos, pero únicamente las soluciones de los problemas deben ser entregadas. La intención de los ejercicios es entrenarlo para que domine el material del curso; a pesar de que no debe entregar soluciones a los ejercicios, usted es responsable del material cubierto en ellos.

Instrucciones para la entrega

Para esta tarea y todas las tareas futuras, la entrega de soluciones es *individual*. Por favor escriba claramente su nombre, código de estudiante y sección en cada hoja impresa entregada o en cada archivo de código (a modo de comentario). Adicionalmente, agregue la información de fecha y nombres de compañeros con los que colaboró; igualmente cite cualquier fuente de información que utilizó.

¿Cómo describir un algoritmo?

En algunos ejercicios y problemas se pide "dar un algoritmo" para resolver un problema. Una solución debe tomar la forma de un pequeño ensayo (es decir, un par de párrafos). En particular, una solución debe resumir en un párrafo el problema y cuáles son los resultados de la solución. Además, se deben incluir párrafos con la siguiente información:

- una descripción del algoritmo en castellano y, si es útil, pseudo-código;
- por lo menos un diagrama o ejemplo que muestre cómo funciona el algoritmo;
- una demostración de la corrección del algoritmo; y
- un análisis de la complejidad temporal del algoritmo.

Recuerde que su objetivo es comunicar claramente un algoritmo. Las soluciones algorítmicas correctas y descritas *claramente* recibirán alta calificación; soluciones complejas, obtusas o mal presentadas recibirán baja calificación.

Ejercicios

La siguiente colección de ejercicios, tomados del libro de Cormen et al. es para repasar y afianzar conceptos, pero no deben ser entregados como parte de la tarea.

1.2-2, 1.2-3 (página 15), 2.2-3 (página 33).

Problemas conceptuales

1. Escribir el código de honor del curso.
2. Ejercicio 2.4: Growth rate (Kleinberg & Tardos, página 67).
3. Ejercicio 1.9: Sorting pancakes (Erickson, página 49).

Problemas prácticos

Hay cinco problemas prácticos cuyos enunciados aparecen a partir de la siguiente página.

A - The Cat in the Hat

Source file name: `cat.py`
Time limit: x seconds

(An homage to Theodore Seuss Geisel)

The Cat in the Hat is a nasty creature,
But the striped hat he is wearing has a rather nifty feature.
With one flick of his wrist he pops his top off.
Do you know what's inside that Cat's hat?
A bunch of small cats, each with its own striped hat.
Each little cat does the same as line three,
All except the littlest ones, who just say "Why me?"
Because the littlest cats have to clean all the grime,
And they're tired of doing it time after time!

A clever cat walks into a messy room which he needs to clean. Instead of doing the work alone, it decides to have its helper cats do the work. It keeps its (smaller) helper cats inside its hat. Each helper cat also has helper cats in its own hat, and so on. Eventually, the cats reach a smallest size. These smallest cats have no additional cats in their hats. These unfortunate smallest cats have to do the cleaning.

The number of cats inside each (non-smallest) cat's hat is a constant, N. The height of these cats-in-a-hat is 1 $N + 1$ times the height of the cat whose hat they are in. The smallest cats are of height one; these are the cats that get the work done. All heights are positive integers.

Given the height of the initial cat and the number of worker cats (of height one), find the number of cats that are not doing any work (cats of height greater than one) and also determine the sum of all the cats' heights (the height of a stack of all cats standing one on top of another).

Input

The input consists of a sequence of cat-in-hat specifications. Each specification is a single line consisting of two positive integers, separated by white space. The first integer is the height of the initial cat, and the second integer is the number of worker cats.

A pair of `'0'`s on a line indicates the end of input.

The input must be read from standard input.

Output

For each input line (cat-in-hat specification), print the number of cats that are not working, followed by a space, followed by the height of the stack of cats. There should be one output line for each input line other than the `'0 0'` that terminates input.

The output must be written to standard output.

Sample Input	**Sample Output**
`216 125`	`31 671`
`5764801 1679616`	`335923 30275911`
`0 0`	

B - Business Center

Source file name: `business.py`
Time limit: x seconds

Sheila is a student and she drives a typical student car: it is old, slow, rusty, and falling apart. Recently, the needle on the speedometer fell off. She glued it back on, but she might have placed it at the wrong angle. Thus, when the speedometer reads s, her true speed is $s + c$, where c is an unknown constant (possibly negative).

Sheila made a careful record of a recent journey and wants to use this to compute c. The journey consisted of n segments. In the i-th segment she traveled a distance of d_i and the speedometer read s_i for the entire segment. This whole journey took time t. Help Sheila by computing c.

Note that while Sheila's speedometer might have negative readings, her true speed was greater than zero for each segment of the journey

Input

The input file contains several test cases. The first line of a test case contains two integers n ($1 \leq n \leq 1000$), the number of sections in Sheila's journey, and t ($1 \leq t \leq 10^6$), the total time. This is followed by n lines, each describing one segment of Sheila's journey. The i-th of these lines contains two integers d_i ($1 \leq d_i \leq 1000$) and s_i ($|s_i| \leq 1000$), the distance and speedometer reading for the i-th segment of the journey. Time is specified in hours, distance in miles, and speed in miles per hour.

The input must be read from standard input.

Output

For each test case, on a line by itself, output the constant c in miles per hour. Your answer should have an absolute or relative error of less than 10^{-9} and be printed with exactly 4 decimal digits after the period.

The output must be written to standard output.

Sample Input	Sample Output

`3 5`
`4 -1`
`4 0`
`10 3`
`4 10`
`5 3`
`2 2`
`3 6`
`3 1`
	`3.0000`
	`-0.5086`

C - Duathlon

Source file name: `duathlon.py`
Time limit: x seconds

A duathlon is a race that involves running r km and cycling k km. n contestants have entered the race; each contestant has different running and cycling speeds. One of the contestants has bribed the organizers to set r and k so that he can win by the maximum margin. You are to determine if this is possible and, if so, give r and k.

Input

The input contains several sets of input. The description of each set is given below:

The first line of each set contains an integer t, the total distance of the race, in km. That is, $r + k = t$. The next line contains an integer n, the number of competitors. For each contestant, a line follows with two real numbers giving the running and cycling speed for that contestant. The last line of input gives the running and cycling speed of the contestant who has bribed the organizers. You may assume t does not exceed 100 km and n does not exceed 20. Input is terminated by end of file. Two consecutive sets may or may not be separated by a blank line.

The input must be read from standard input.

Output

For each set of input produce one line of output. The output description for each set is given below:

If it is possible to fix the race as describe above, print a message giving r and k accurate to two decimal places, and the amount of seconds by which the cheater will win the race, (0 in case some competitor ties him), as in the sample below. If it is not possible, print `'The cheater cannot win.'`

There is no blank line between outputs for two consecutive sets.

The output must be written to standard output.

Sample Input

```
100
3
10.0 40.0
20.0 30.0
15.0 35.0

100
3
10.0 40.0
20.0 30.0
15.0 25.0
```

Sample Output

```
The cheater can win by 612 seconds with r = 14.29km and k = 85.71km.
The cheater cannot win.
```

D - N + NOD(N)

Source file name: `nod.py`
Time limit: x seconds

Consider an integer sequence N where,

$$N_0 = 1$$
$$N_i = N_{i-1} + \text{NOD}(N_{i-1}) \qquad \text{, for } i > 0.$$

Here, NOD(x) indicates the number of divisors of x. For example, NOD(100) $=$ 9 since the divisors of 100 are $1, 2, 4, 5, 10, 20, 25, 50, 100$. Therefore, the first few terms of the N sequence are $1, 2, 4, 7, 9, 12, 18$.

Given two integers A and B, find out the number of integers in the above sequence that lie within the range $[A, B]$.

Input

The first line of input is an integer T ($T \geq 0$) that indicates the number of test cases. Each case contains two integers, A followed by B ($1 \leq A \leq B \leq 1\,000\,000$).

The input must be read from standard input.

Output

For each case, output the case number first followed by the required result.

The output must be written to standard output.

Sample Input	Sample Output

`3`
`1 18`
`1 100`
`3000 4000`
	`Case 1: 7`
	`Case 2: 20`
	`Case 3: 87`

E - The Closest Pair Problem

Source file name: `pair.py`
Time limit: x seconds

Given a set of points in a two dimensional space, you will have to find the distance between the closest two points.

Input

The input contains several sets of input. Each set of input starts with an integer N ($0 \leq N \leq 10\,000$), which denotes the number of points in this set. The next N line contains the coordinates of N two-dimensional points. The first of the two numbers denotes the x-coordinate and the latter denotes the y-coordinate. The input is terminated by a set whose $N = 0$ and should not be processed. The value of the coordinates will be less than 40 000 and non-negative.

The input must be read from standard input.

Output

For each set of input produce a single line of output containing a floating point number (rounded up to four decimals after the decimal point), denoting the distance between the closest two points. If there is no such two points in the input whose distance is less than 10 000, print the line '`INFINITY`'.

The output must be written to standard output.

Sample Input	Sample Output

```
3
0 0
10000 10000
20000 20000
5
0 2
6 67
43 71
39 107
189 140
0
```
	```
	INFINITY
	36.2215
	```