ADA - Análisis y Diseño de Algoritmos, 2025-1
Tarea 1: Semanas 1 y 2

Para entregar el domingo 9 de febrero de 2025

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 2.3: Growth rate (Kleinberg & Tardos, página 67).
3. Ejercicio 2.2: Text segmentation problem (Erickson, página 93).

Problemas prácticos

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

A - Bit Maps

Source file name: `bit.py`
Time limit: 3 seconds

The bitmap is a data structure that arises in many areas of computing. In the area of graphics, for example, a bitmap can represent an image by having a `'1'` represent a black pixel and a `'0'` represent a white pixel.

Consider the following two ways of representing a rectangular bit map. In the first, it is simply represented as a two dimensional array of 1's and `0`'s. The second is based on a decomposition technique. First, the entire bit map is considered. If all bits within it are 1, a `'1'` is output. If all bits within it are 0, a `'0'` is output. Otherwise, a `D` is output, the bit map is divided into quarters (as described below), and each of those is processed in the same way as the original bit map. The quarters are processed in top left, top right, bottom left, bottom right order. Where a bit map being divided has an even number of rows and an even number of columns, all quarters have the same dimensions. Where the number of columns is odd, the left quarters have one more column than the right. Where the number of rows is odd the top quarters have one more row than the bottom. Note that if a region having only one row or one column is divided then two halves result, with the top half processed before the bottom where a single column is divided, and the left half before the right if a single row is divided.

Write a program that will read in bitmaps of either form and transform them to the other form.

Input

Input will consist of a series of bit maps. Each bit map begins with a line giving its format (`'B'` or `'D'`) and its dimensions (rows and columns). Neither dimension will be greater than 200. There will be at least one space between each of the items of information. Following this line will be one or more lines containing the sequence of `'1'`, `'0'` and `'D'` characters that represent the bit map, with no intervening spaces. Each line (except the last, which may be shorter) will contain 50 characters. A `'B'` type bitmap will be written left to right, top to bottom.

The file will be terminated by a line consisting of a single `'#'`.

The input must be read from standard input.

Output

Output will consist of a series of bitmaps, one for each bit map of the input. Output of each bit map begins on a new line and will be in the same format as the input. The width and height are to be output right justified in fields of width four.

The output must be written to standard output.

Sample Input	**Sample Output**

`B 3 4`
`001000011011`
`D 2 3`
`DD10111`
`#`
	`D 3 4`
	`D0D1001D101`
	`B 2 3`
	`101111`

B - Cantor

Source file name: `cantor.py`
Time limit: 1 second

The ternary expansion of a number is that number written in base 3. A number can have more than one ternary expansion. A ternary expansion is indicated with a subscript 3. For example, $1 = 1_3 = 0.222\ldots_3$, and $0.875 = 0.212121\ldots_3$. The Cantor set is defined as the real numbers between 0 and 1 inclusive that have a ternary expansion that does not contain a 1. If a number has more than one ternary expansion, it is enough for a single one to not contain a 1. For example, $0 = 0.000\ldots_3$ and $1 = 0.222\ldots_3$, so they are in the Cantor set. But $0.875 = 0.212121\ldots_3$ and this is its only ternary expansion, so it is not in the Cantor set. Your task is to determine whether a given number is in the Cantor set.

Input

The input consists of several test cases. Each test case consists of a single line containing a number x written in decimal notation, with $0 \leq x \leq 1$, and having at most 6 digits after the decimal point. The last line of input is '`END`'. This is not a test case.

The input must be read from standard input.

Output

For each test case, output '`MEMBER`' if x is in the Cantor set, and '`NON-MEMBER`' if x is not in the Cantor set.

The output must be written to standard output.

Sample Input	Sample Output

`0`
`1`
`0.875`
`END`
	`MEMBER`
	`MEMBER`
	`NON-MEMBER`

C - Interesting Drink

Source file name: `drink.py`
Time limit: 1 second

Vasiliy likes to rest after a hard work, so you may often meet him in some bar nearby. As all programmers do, he loves the famous drink "Beecola", which can be bought in n different shops in the city. It's known that the price of one bottle in the shop i is equal to x_i coins.

Vasiliy plans to buy his favorite drink for q consecutive days. He knows, that on the i-th day he will be able to spent m_i coins. Now, for each of the days he want to know in how many different shops he can buy a bottle of "Beecola".

Input

The input has several test cases. The first line of each test case contains a single integer n ($1 \le n \le 100\,000$) — the number of shops in the city that sell Vasiliy's favourite drink. The second line contains n integers x_i ($1 \le x_i \le 100\,000$) — prices of the bottles of the drink in the i-th shop. The third line contains a single integer q ($1 \le q \le 100\,000$) — the number of days Vasiliy plans to buy the drink. Then follow q lines each containing one integer m_i ($1 \le m_i \le 10^9$) — the number of coins Vasiliy can spend on the i-th day.

The input must be read from standard input.

Output

For each test case print q integers. The i-th of them should be equal to the number of shops where Vasiliy will be able to buy a bottle of the drink on the i-th day.

The output must be written to standard output.

Sample Input	**Sample Output**
5 3 10 8 6 11 4 1 10 3 11	0 4 1 5

D - Rain Fall

Source file name: `rain.py`
Time limit: 1 second

Rainfall is measured in millimeters. The rain is collected in a vertical transparent tube with millimeter markings, and once the rain has stopped falling, one can check the height of the water in the tube.

In our problem, the tube unfortunately has a leak at height L millimeters (mm). If the water level is above the leak then water drains from the tube at a rate of K millimeters per hour (mm/h).

We want to figure out how much rain fell during a particular rainfall. We assume that the tube is high enough that it does not overflow. We also assume that rain falls at an (unknown) uniform rate during a rainfall, and that water does not evaporate from the tube. The height of the leak itself is also negligible.

Input

The first line of the input file contains an integer N which denotes the total number of test cases. The description of each test case is given next. A line with five blank-separated positive numbers L, K, T_1, T_2, and H, where:

- L is where the leak is (mm)
- K is the rate at which water leaks (mm/h)
- T_1 is the duration of the rainfall (h)
- T_2 is the time between the end of the rainfall and the observation of the water level (h)
- H is the water level in the tube when we observe it (mm)

Each number is at least 0.01 and at most 1000.00, and each is given with two decimals.

The input must be read from standard input.

Output

For each test case print one line with two blank-separated floating point numbers F_1 and F_2, where F_1 is the smallest rainfall in millimeters that would result in the given observation and F_2 is the largest rainfall in millimeters that would result in the given observation. Values need to be printed with 6 decimals, with either absolute or relative error smaller than 10^{-7}.

The output must be written to standard output.

Sample Input	**Sample Output**

`2`
`80.00 0.50 2.00 1.50 80.00`
`150.00 1.00 100.00 150.00 100.00`
	`80.000000 80.759403`
	`100.000000 100.000000`

E - Underwater Snipers

Source file name: `snipers.py`
Time limit: 4 seconds

King *Motashota* is in a war against the mighty Bachchaloks. He has formed a well-trained army of snipers and plans to use them as much as possible. In one of the missions, he has S snipers. They will be dispatched to get rid of the soldiers guarding the bank of the river 'Nodi'.

From satellite images, Motashota has located positions of all enemy soldiers. Now, the plan is, snipers will take their positions. They are excellent swimmers, so, you can assume that they won't get caught, while taking position. Upon order from *Motashota*, they will start shooting enemy soldiers. A sniper can shoot a soldier if Euclidean distance between the soldier and sniper is no more than D. After the snipers get rid of all the soldiers, they can proceed with the operation. So, it is important for them to position the snipers in such a way that, all soldiers are within the range of at least one sniper.

In addition, when snipers start shooting, the guards will be alert, and thus, snipers can't change their position, they can only continue shooting from their position.

The river bank is defined by the horizontal line $y = k$. All points (x, y) where $y > k$ is in the enemy territory, and if $y < k$, then it's on the water. You will be given location of N soldiers, strictly in the enemy territory, you have to place S soldiers in the water, so that, they can kill all soldiers. For security reasons, the snipers should be as far from the bank as possible. For any sniper in position (x_i, y_i), the distance from the bank is $|y_i - k|$. If for all snipers, the minimum of them is $M = min\{|y_i - k|\}$, you have to maximize M.

Both the soldiers and snipers are really superstitious. They will stay only in integer coordinates.

Input

First line contains an integer $T \geq 0$, the number of test cases. Each test case starts with four integers, k ($-10^8 \leq k \leq 10^8$), N ($1 \leq N \leq 10^4$), S ($1 \leq S \leq 10^4$), and D ($1 \leq D \leq 10^9$), the position of the bank, the number of guards, the number of snipers, and the range of the snipers, respectively. This is followed by N lines, each containing a pair of integers (x_i, y_i) the position of i-th guard ($-10^8 \leq x_i \leq 10^8, k < yi \leq 10^8$). There is a blank line before each test case.

The input must be read from standard input.

Output

For each test case, output the case number followed by an integer, M, which is defined in the statement. If the snipers can't kill all guards, output: '`IMPOSSIBLE`'.

The output must be written to standard output.

Sample Input	**Sample Output**
2 0 3 2 4 1 1 3 2 9 1 0 3 1 4 1 1 3 2 9 1	Case 1: 2 Case 2: IMPOSSIBLE