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

Para entregar el domingo 10 de agosto 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 2.4: Growth rate (Kleinberg & Tardos, página 67).
3. Ejercicio 5.3: Fraud detection (Kleinberg & Tardos, página 246).

Problemas prácticos

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

A - Angry Bids

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

Deciding the price of a product is not an easy matter. If the price is high, then consumers get angry because they think it is too expensive. But if the price is low, then producers get angry because they do not earn enough money.

In this problem, you have to bid for the price that produces the minimum number of angry people.

For a given product, we have a set of producers and a set of consumers. Each producer has suggested his/her ideal price for the product. If your bid is below that ideal price, then the producer will get angry. Also, each consumer has suggested his/her ideal price for the product. If your bid is above that ideal price, then the consumer will get angry.

You have to find the price that produces the minimum number of angry people (either producers or consumers). Your bid price should always be between 0 and 10^9. And if there is more than one optimal solution, you have to output the lowest one.

Input

The input consists of a series of test cases. The first line contains a number that indicates the number of test cases.

Each test case consists of three lines. The first one contains two integers, P and C, indicating the number of producers and consumers of the product, respectively. These numbers will be between 0 and 10^4. Then, the second line contains P integer numbers, the ideal prices for the producers. And the third line contains C integer numbers, the ideal prices for the consumers. The prices are between 1 and 10^8.

The input must be read from standard input.

Output

For each input case, you have to output two numbers separated by one space: the bid price that produces the minimum number of angry people; and the total number of angry people for that price. Remember that if there is more than one optimal solution, you have to output the one with lowest price; and the result cannot be a negative number.

The output must be written to standard output.

Sample Input	**Sample Output**
4 3 3 10 16 12 21 20 25 4 3 104 120 98 132 120 88 140 8 8 36 27 52 72 36 37 26 38 35 75 36 39 44 82 23 62 0 2 28 71	16 0 120 2 38 5 0 0

B - Berland Collider

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

Recently the construction of Berland collider has been completed. Collider can be represented as a long narrow tunnel that contains n particles. We associate with collider 1-dimensional coordinate system, going from left to right. For each particle we know its coordinate and velocity at the moment of start of the collider. The velocities of the particles don't change after the launch of the collider. Berland scientists think that the big bang will happen at the first collision of particles, whose velocities differs in directions. Help them to determine how much time elapses after the launch of the collider before the big bang happens.

Input

The first line of each testcase contains a single integer n ($1 \le n \le 5 \cdot 10^5$) — the amount of particles in the collider. Next n lines contain description of particles. Each particle is described by two integers x_i, v_i ($-10^9 \le x_i, v_i \le 10^9$, $v_i \neq 0$) — coordinate and velocity respectively. All the coordinates are distinct. The particles are listed in order of increasing of coordinates. All the coordinates are in meters, and all the velocities in meters per second. The negative velocity means that after the start of collider the particle will move to the left, and the positive that the particle will move to the right.

The input must be read from standard input.

Output

For each testcase, if there will be no big bang, output `-1`. Otherwise output one number — how much time in seconds elapses after the launch of the collider before the big bang happens. Your answer must have a relative or absolute error less than 10^{-9}.

The output must be written to standard output.

Sample Input	Sample Output
3 -5 9 0 1 5 -1 6 1 3 2 3 3 3 4 -3 5 -1 6 -100	1.000000000000000000000 0.02912621359223301065

C - Conquering Keokradong

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

This winter we are going on a trip to Bandorban. The main target is to climb up to the top of Keokradong. So, we will use a trail. The trail is a continuous marked footpath that goes from Bandorban to Keokradong.

Part of the experience is also the route planning of the trip. We have a list of all possible campsites that we can use along the way and we want to do this trip so that we only stop K nights to camp. We also know in advance the distance between consecutive campsites and we are only allowed to camp at a campsite. Our goal is to plan the trip so that we minimize the maximum amount of walking done in a single day. In other words, if our trip involves 2 nights (3 days of walking), and we walk 9, 10, 5 miles on each day respectively, the cost (maximum amount of walking done in one day) is 10. Another schedule that involves walking 9, 6, 9 miles on each day has cost 9.

Given the distances between N consecutive campsites of a trail and given the number of nights for your trip, K, your task is to devise a camping strategy for the specified trail such that it minimizes the maximum amount of walking done in a single day. Note that the first distance value given is the distance from our start-point of the trail to our 1st campsite, and the last distance value given is the distance from our Nth campsite to our end-point of the trail.

Input

Input starts with an integer T, denoting the number of test cases. Each case contains of two integers, the number of campsites, N ($1 \leq N \leq 1000$) and the number of nights of the trip, K ($1 \leq K \leq min(N, 300)$). The following $N + 1$ lines indicate the distance in miles between consecutive campsite locations. All the integers will be positive and less than 10000.

The input must be read from standard input.

Output

For each case of input you have to print the case number and the minimized cost as described above. Then print $K + 1$ lines, each containing the amount of distance covered in i-th day. As there can be many solutions, the primary target is to find the one which ensures that each day we have to walk some distance. For ties, print the one where the distance covered in first day is maximum, then the distance covered in second day is maximum and so on.

The output must be written to standard output.

Sample Input	**Sample Output**
1 4 3 7 2 6 4 5	Case 1: 8 7 8 4 5

D - Opening Doors

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

21th June!! Holidays at last!!

That's what everyone in the school thought when the bell rang. It was time to leave the class and go away forever (or until September). Dima was running to the exit when he saw that paper on the wall:

```
Don't forget to unlock all the lockers so that we can clean them in summer.
                                                                The principal.
```

Oh, no! Dima forgot to unlock it! So he had to go to the corridor where lockers were. When he got there, all the lockers had been already unlocked. Some of them were open and some of them were closed. Lockers were numbered from 1 to 90 (the number of boys in the school) and he had the last one. So he walked along the long corridor. While he reached his locker, he started to think why some of the doors were open and some closed. Then he invented a method to keep some open and some closed:

> Once all the lockers were unlocked, the boy with locker number 1 opened all doors. Then the boy with locker number 2 closed the doors multiple of 2. Then the boy with locker number 3 changed the status of all doors multiple of 3 (he opens doors 6,12, ... and he closes doors 3, 9, ...) ... and until everyone, including Dima, had done so.

Now he wants to know whether that was what happened or not. As a first-sight method, he wants to see if the door with the biggest number that is open matches the door with the biggest number that keeps open using his method. That's where he needs your help. He could do it by hand, but he also wants to check it next year (and the number of students may change), so he asked you for a program that simulates it for any number of students.

Input

Each line in the input will contain a single number N ($1 \leq N \leq 10^{100}$), indicating the number of boys in the school (you never know what the number of students will be if Earth population keeps growing). Input will be terminated by a test case with $N = 0$; that line shouldn't be processed.

The input must be read from standard input.

Output

For each line in the input, write a line with the number of the door with the biggest number that keeps open.

The output must be written to standard output.

Sample Input	**Sample Output**
1 90 0	1 81

E - Sylvester Construction

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

A Hadamard matrix of order n is an $n \times n$ matrix containing only 1s and -1s, called H_n, such that $H_n H_n^T = nI_n$ where I_n is the $n \times n$ identity matrix. An interesting property of Hadamard matrices is that they have the maximum possible determinant of any $n \times n$ matrix with elements in the range $[-1, 1]$. Hadamard matrices have applications in errorcorrecting codes and weighing design problems.

The Sylvester construction is a way to create a Hadamard matrix of size $2n$ given H_n. H_{2n} can be constructed as:

$$H_{2n} = \begin{pmatrix} H_n & H_n \\ H_n & -H_n \end{pmatrix}$$

for example:

$$H_1 = (1) \qquad\qquad H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

and so on.

In this problem you are required to print a part of a Hadamard matrix constructed in the way described above.

Input

The first number in the input is the number of test cases to follow. For each test case there are five integers: n, x, y, w and h. n will be between 1 and 2^{62} (inclusive) and will be a power of 2. x and y specify the upper left corner of the sub matrix to be printed, w and h specify the width and height respectively. Coordinates are zero based, so $0 \leq x, y < n$. You can assume that the sub matrix will fit entirely inside the whole matrix and that $0 < w, h \leq 100$.

The input must be read from standard input.

Output

For each test case print the sub matrix followed by an empty line.

The output must be written to standard output.

Sample Input	**Sample Output**
3 2 0 0 2 2 4 1 1 3 3 268435456 12345 67890 11 12	1 1 1 -1 -1 1 -1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 1 1 1 -1 1 -1 -1 1 -1 1 1 -1 1 -1 1 -1 -1 -1 -1 1 1 1 1 -1 -1 -1 -1 1 -1 1 1 -1 1 -1 -1 1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 1 -1 1 -1 1 -1 1 1 -1 1 -1 -1 1 1 -1 -1 1 1 1 1 -1 -1 1 1 -1 -1 1 1 -1 1 -1 -1 1 -1 -1 -1 1 1 1 1 1 1 1 1 1 -1 1 1 -1 1 -1 1 -1 1 -1