ADA - Análisis y Diseño de Algoritmos, 2024-1
Tarea 2: Semanas 3 y 4

Para entregar el domingo 18 de febrero de 2024

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.

3.2-2, 3.2-3, 3.3-1, 3.3-2

Problemas conceptuales

1. Ejercicio 1.6 (A, E, I): *Recursion Trees* (Erickson, página 49).
2. Ejercicio 1.16: *Weighted Median* (Erickson, página 52).
3. Ejercicio 1.38: *Central Vertices* (Erickson, página 64).

Problemas prácticos

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

A - Opening Doors

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

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

B - A Mid-summer Night's Dream

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

This is year 2200AD. Science has progressed a lot in two hundred years. Two hundred years is mentioned here because this problem is being sent back to 2000AD with the help of time machine. Now it is possible to establish direct connection between man and computer CPU. People can watch other peoples dream on 3D displayer (that is the monitor today) as if they were watching a movie.

One problem in this century is that people have become so dependent on computers that their analytical ability is approaching zero. Computers can now read problems and solve them automatically. But they can solve only difficult problems. There are no easy problems now. Our chief scientist is in great trouble as he has forgotten the number of his combination lock. For security reasons computers today cannot solve combination lock related problems.

In a mid-summer night the scientist has a dream where he sees a lot of unsigned integer numbers flying around. He records them with the help of his computer, Then he has a clue that if the numbers are $(X_1, X_2, \ldots, X_n)$ he will have to find an integer number A (this A is the combination lock code) such that

$$(|X_1 - A| + |X_2 - A| + \cdots + |X_n - A|)$$

is minimum.

Input

Input will contain several blocks. Each block will start with a number n ($0 < n \leq 1\,000\,000$) indicating how many numbers he saw in the dream. Next there will be n numbers. All the numbers will be less that 65 536. The input will be terminated by end of input.

The input must be read from standard input.

Output

For each set of input there will be one line of output. That line will contain the minimum possible value for A. Next it will contain how many numbers are there in the input that satisfy the property of A (the summation of absolute deviation from A is minimum). And finally you have to print how many possible different integer values are there for A (these values need not be present in the input). These numbers will be separated by single space.

The output must be written to standard output.

Sample Input	Sample Output

```
2
10
10
4
1
2
2
4
```
	```
	10 2 1
	2 2 1
	```

C - Destroy the Moon to Save the Earth

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

A giant asteroid of 25 km diameter is approaching the Earth, threatening the lives of all humans, animals, plants and pokemons. In 30 years, the impact will be so brutal that it will immediately destroy half of the planet's surface. The other half will suffer giant tsunamis, global earthquakes, and the fall of space debris. The few unfortunates who survive will have to face an endless winter, in a desert planet. There is no hope ... or is there?

You have a plan! Slightly modifying the orbit of the Moon, we will be able to use her as a shield against the asteroid. We will nearly destroy the Moon, to save the planet Earth.

The greatest scientists in the world have calculated precisely the time when the impact will occur, in number of days. We also know the speed of rotation of the Moon around Earth, which is approximately 1 km/s. This speed can be increased or decreased by removing or adding mass to the Moon. More specifically, each ton of mass we add, the speed is reduced in 1 mm/s. And each ton we remove, the speed is increased in 1 mm/s. It is a very small change, but after many years it can make a big distance.

Finally, another essential parameter is the distance to the impact point. Supposing for simplicity a linear movement of the Moon, this distance can be defined as the distance from the Moon to the point where it should be to impact the asteroid, in the day of impact. For example, if the time to impact is 30 years and the distance to the impact is 2 500 km, then we need to increase the speed of the Moon so that after 30 years it has travelled 2 500 km more.

You have to compute the number of tons that have to be added or removed to destroy the Moon and save the Earth. The mass is assumed to be applied exactly at time 0.

Input

The input contains several test cases. Each test case in described in a line with the following values: $T\ S\ D$. All of them are integer numbers. The number T is the time to impact, measured in days. The number S is the current speed of rotation of the Moon, in mm/s. And D is the distance to the impact point, in km, which can be positive or negative.

The input must be read from standard input.

Output

For each test case, the program has to produce one line. If we have to add X tons of mass to the Moon (or if the result is 0), the output should be '`Add X tons`'. If we have to remove X tons of mass, the output should be '`Remove X tons`'. The number of tons X, has to be truncated to an integer; e.g., if the value is 34.95, you have to output 34.

The output must be written to standard output.

Sample Input	Sample Output
5	Remove 2 tons
10950 1022083 2500	Remove 3 tons
7300 1022083 2500	Add 17 tons
356 1027123 -525	Add 1 tons
12540 1082726 -2100	Add 0 tons
1000 1010023 0	

D - Pinary

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

“Pinary” number is a positive number using only two digits “0” and “1” with usual rule that it must not begin with a 0, and the additional rule that two successive digits must not be both “1”. This means that the factor “11” is forbidden in the string. So the allowed Pinary writings are 1, 10, 100, 101, 1000, 1001, . . . , 1000101010101000010001. For example, “100101000” is a Pinary number, but neither “0010101” nor “10110001” are Pinary numbers.

Each Pinary number represents a positive integer in the order they appear (using length order, lexicographic order), that is, 1 is mapped to 1, 10 is mapped to 2. And 100, 101 and 1000 are mapped to 3, 4 and 5, respectively. You are to write a program to generate Pinary number representations for integers given.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with a line containing a postive integer $2 < K < 90\,000\,000$.

The input must be read from standard input.

Output

Your program is to write to standard output. Print exactly one line for each test case. For each test case, print the Pinary number representation for input integer.

The output must be written to standard output.

Sample Input	Sample Output

```
3
7
2000
22
```
	```
	1010
	1001000001001000
	1000001
	```

E - Slalom

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

You are competing in a ski slalom, and you need to select the best skis for the race. The format of the race is that there are N pairs of left and right gates, where each right gate is W metres to the right of its corresponding left gate, and you may neither pass to the left of the left gate nor to the right of the right gate. The i-th pair of gates occurs at distance y_i down the hill, with the horizontal position of the i-th left gate given by x_i. Each gate is further down the hill than the previous gate (i.e., $y_i < y_{i+1}$ for all i).

You may select from S pairs of skis, where the j-th pair has speed s_j. Your movement is governed by the following rule: if you select a pair of skis with speed s_j , you move with a constant downward velocity of s_j metres per second. Additionally, at any time you may move at a horizontal speed of at most v_h metres per second.

You may start and finish at any two horizontal positions. Determine which pair of skis will allow you to get through the race course, passing through all the gates, in the shortest amount of time.

Input

The first line of input contains a single integer, the number of test cases to follow. The first line of each test case contains the three integers W, v_h, and N, separated by spaces, with $1 \leq W \leq 10^8$, $1 \leq v_h \leq 10^6$, and $1 \leq N \leq 10^5$. The following N lines of the test case each contain two integers x_i and y_i, the horizontal and vertical positions respectively of the i-th left gate, with $1 \leq x_i, y_i \leq 10^8$. The next line of the test case contains an integer S, the number of skis, with $1 \leq S \leq 10^6$. The following S lines of the test case each contain one integer s_j, the speed of the j-th pair of skis, with $1 \leq s_j \leq 10^6$.

The input must be read from standard input.

Output

Output one line for each test case. If it is impossible to complete the race with any pair of skis, print the line '`IMPOSSIBLE`'. Otherwise, print the vertical speed s_j of the pair of skis that allows you to get through the race course in the shortest time.

The output must be written to standard output.

Sample Input	Sample Output

```
2
3 2 3
1 1
5 2
1 3
3
3
2
1
3 2 3
1 1
5 2
1 3
1
3
```
	```
	2
	IMPOSSIBLE
	```