ADA - Análisis y Diseño de Algoritmos, 2019-1
Tarea 1: Semanas 1 y 2

Para entregar el viernes 1 de febrero/domingo 3 de febrero de 2019

Problemas conceptuales a las 16:00 (1 de febrero) en el Departamento de Electrónica y Ciencias de la Computación

Problemas prácticos a las 23:59 (3 de febrero) 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

1.1-2 (página 11), 1.2-2, 1.2-3, 1-1 (página 14), 2.1-1, 2.1-2, 2.1-3 (página 22), 2.2-1, 2.2-2 (página 29), 2.3-1 (página 37), 2.3-4, 3.1-1 (página 52), 3.1-4, 3.1-8 (página 53), 3.2-1, 3.2-7, 3-2 columnas O, Ω y Θ (página 61), 3-3 parte *a* (página 62).

Problemas conceptuales

1. Escribir el código de honor del curso.
2. Ejercicios 2 y 4: Orden asintótico y eficiencia algorítmica (Kleinberg & Tardos páginas 67 y 68).
3. Ejercicio 6: *Array Sums* (Kleinberg & Tardos página 68 y 69).
4. Ejercicio 7: *Folk Songs* (Kleinberg & Tardos página 69).
5. Ejercicio 14: *Inversions* (Erickson, Capítulo 1)
6. Ejercicios 4a, 4b y 13: Demostraciones por inducción (Erickson, Apéndice I.1).

Problemas prácticos

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

A - The Tower of Babylon

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

Perhaps you have heard of the legend of the Tower of Babylon. Nowadays many details of this tale have been forgotten. So now, in line with the educational nature of this contest, we will tell you the whole story:

> The Babylonians had n types of blocks and an unlimited supply of blocks of each type. Each type-i block was a rectangular solid with linear dimensions (x_i, y_i, z_i). A block could be reoriented so that any two of its three dimensions determined the dimensions of the base and the other dimension was the height.
>
> They wanted to construct the tallest tower possible by stacking blocks. The problem was that, in building a tower, one block could only be placed on top of another block as long as the two base dimensions of the upper block were both strictly smaller than the corresponding base dimensions of the lower block. This meant, for example, that blocks oriented to have equal-sized bases couldn't be stacked.

Your job is to write a program to determine the height of the tallest tower that can be built with a set of blocks.

Input

The input will contain one or more test cases. The first line of each test case contains an integer n representing the number of different blocks in the following data set. The maximum value for n is 30. Each of the next n lines contains three integers representing the values x_i, y_i, z_i. Input is terminated by a value of zero (0) for n.

The input must be read from standard input.

Output

For each test case, print one line containing with the format '`Case c: maximum-height = h`' where c is the case number (starting from 1) and h the maximum height.

The output must be written to standard output.

Sample Input	**Sample Output**
1 10 20 30 2 6 8 10 5 5 5 7 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5 6 6 6 7 7 7 5 31 41 59 26 53 58 97 93 23 84 62 64 33 83 27 0	Case 1: maximum height = 40 Case 2: maximum height = 21 Case 3: maximum height = 28 Case 4: maximum height = 342

B - The Playboy Chimp

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

Once upon a time, there lived a chimpanzee called Luchu Bandor (a.k.a. *Playboy Chimp*). Luchu was unhappily married to Bunty Mona, a short but cute little lady chimp. Luchu was tall and handsome –he was feeling uncomfortable taking Bunty to public places along with him. People would stare at them all the while.

At one point, Luchu could not stand it anymore and he decided to do some justice to his name. He started looking for a new hope in the Lady Chimps' High School. Every day Luchu would climb up a bamboo tree and wait for the morning drill to start. From there he could see each and every lady chimp doing their routine drill.

Now, Luchu was looking for the tallest lady chimp that would be shorter than him; he would also like to consider someone a little taller than him. But someone of his same height will never be on his list. Every morning Luchu picks up a line of lady chimps and finds the best two according to his set criterion. His job has been made easy by the fact that the lady chimps in each line are ordered by their height, the shortest one is in the front and the tallest one is at the back.

Your task is to help Luchu on one particular day to find two lady chimps: the tallest one shorter than him and the shortest one taller than him.

Input

There will be only one set of input for this problem. The first line of input gives you a number N ($1 \leq N \leq 50000$), the number of lady chimps on the line. In the next line you would have N integers (in the range 1 to 2^{31-1}) giving the heights of the N chimps. There would be a single space after every number. You can assume that the chimps are ordered in non-decreasing order of their heights. In the next line you would have an integer Q ($1 \leq Q \leq 25000$) giving the number of queries. Then in the next line Q queries will follow. Then you would have Q numbers giving the height of Luchu! Don't worry, Luchu is from the land where people can have 3 birthdates; Q heights for a chimpanzee will make no difference here. The Q numbers are listed on a line and their range from 1 to 2^{31-1}, and as before you would find a single space after every query number. The query numbers are not supposed to come in any particular order.

The input must be read from standard input.

Output

For each query height, print two numbers in one line. The first one would be the height of the tallest lady chimp that is shorter than Luchu, and the next number would be the height of the shortest lady chimp that is taller than him. These two numbers are to be separated by a single space. Whenever it is impossible to find any of these two heights, replace that height with an uppercase '*X*'.

The output must be written to standard output.

Sample Input	**Sample Output**
4 1 4 5 7 4 4 6 8 10	1 5 5 7 7 X 7 X

C - Fill the Containers

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

A conveyor belt has a number of vessels of different capacities each filled to brim with milk. The milk from conveyor belt is to be filled into m containers. The constraints are:

- Whenever milk from a vessel is poured into a container, the milk in the vessel must be completely poured into that container only. That is milk from same vessel can not be poured into different containers.
- The milk from the vessel must be poured into the container in order which they appear in the conveyor belt. That is, you cannot randomly pick up a vessel from the conveyor belt and fill the container.
- The ith container must be filled with milk only from those vessels that appear earlier to those that fill jth container, for all $i < j$.

Given the number of containers m, you have to fill the containers with milk from all the vessels, without leaving any milk in the vessel. The containers need not necessarily have same capacity. You are given the liberty to assign any possible capacities to them.

For example, if you are given the containers $1, 2, 3, 4, 5$ and $m = 3$ vessels, you are free to assign the capacity of each vessel at will. The best configuration, the one that minimizes the maximum capacity, is obtained by placing the $1, 2, 3$ containers in the first vessel, 4 in the second one, and 5 in the third one: in this case, the maximum capacity is 6, which is optimal among all possible configurations with three vessels for the given containers.

Your job is to find out the minimal possible capacity of the container which has maximal capacity. (If this sounds confusing, read down for more explanations.)

Input

A single test case consist of 2 lines. The first line specifies $1 \leq n \leq 1\,000$ the number of vessels in the conveyor belt and then m which specifies the number of containers to which, you have to transfer the milk ($1 \leq m \leq 1\,000\,000$). The next line contains, the capacity $1 \leq c \leq 1\,000\,000$ of each vessel in the order in which they appear in the conveyor belt. Note that, milk is filled to the brim of any vessel. So the capacity of the vessel is equal to the aumont of milk in it.

The input must be read from standard input.

Output

For each test case, print the minimal possible capacity of the container with maximal capacity. That is there exists a maximal capacity of the containers, below which you can not fill the containers without increasing the number of containers. You have to find such capacity and print it on a single line.

The output must be written to standard output.

Sample Input	**Sample Output**
5 3 1 2 3 4 5 3 2 4 78 9	6 82

D - The Jackpot

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

As Manuel wants to get rich fast and without too much work, he decided to make a career in gambling. Initially, he plans to study the gains and losses of players, so that, he can identify patterns of consecutive wins and elaborate a win-win strategy. But Manuel, as smart as he thinks he is, does not know how to program computers. So he hired you to write programs that will assist him in elaborating his strategy.

Your first task is to write a program that identifies the maximum possible gain out of a sequence of bets. A bet is an amount of money and is either winning (and this is recorded as a positive value), or losing (and this is recorded as a negative value).

Input

The input set consists of several test cases. The first line in a test case contains a positive number $N \leq 10000$, that gives the length of the sequence. The sencond line in a test case contains N blank-separated integers. Each bet is an integer greater than -1000 and less than 1000.

The input is terminated with $N = 0$.

The input must be read from standard input.

Output

For each given input set, the output will echo a line with the corresponding solution. If the sequence shows no possibility to win money, then the output is the message "`Losing streak.`"

The output must be written to standard output.

Sample Input	**Sample Output**

`5`
`12 -4 -10 4 9`
`3`
`-2 -1 -2`
`0`
	`The maximum winning streak is 13.`
	`Losing streak.`

E - Where is the Marble?

Source file name: `marble.py`
Time limit: 2 seconds

Raju and Meena love to play with Marbles. They have got a lot of marbles with numbers written on them. At the beginning, Raju would place the marbles one after another in ascending order of the numbers written on them. Then Meena would ask Raju to find the first marble with a certain number. She would count $1 \ldots 2 \ldots 3$. Raju gets one point for correct answer, and Meena gets the point if Raju fails. After some fixed number of trials the game ends and the player with maximum points wins.

Today it's your chance to play as Raju. Being the smart kid, you'd be taking the favor of a computer. But don't underestimate Meena, she had written a program to keep track how much time you're taking to give all the answers. So now you have to write a program, which will help you in your role as Raju.

Input

There can be multiple test cases. Each test case begins with two blank-separated integers in a line: N is the number of marbles and Q the number of queries Mina would make. The next N lines would contain each the numbers written on the N marbles. These marble numbers will not come in any particular order. Following Q lines will have Q queries. Be assured, none of the input numbers are greater than 10000 and none of them are negative. Input is terminated by a test case where $N = 0$ and $Q = 0$.

The input must be read from standard input.

Output

For each test case output the serial number of the case, starting with 1. For each of the queries, print one line of output. The format of this line will depend upon whether or not the query number is written upon any of the marbles. The two different formats are described below:

- '`x found at y`', if the first marble with number x was found at position y. Positions are numbered $1, 2, \ldots, N$.
- '`x not found`', if the marble with number x is not present.

Look at the output for sample input for details.

The output must be written to standard output.

Sample Input	**Sample Output**
4 1 2 3 5 1 5 5 2 1 3 3 3 1 2 3 0 0	CASE# 1: 5 found at 4 CASE# 2: 2 not found 3 found at 3