Codeforces Round #241 (Div. 2)

A. Guess a number!

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

A TV show called "Guess a number!" is gathering popularity. The whole Berland, the old and the young, are watching the show.

The rules are simple. The host thinks of an integer y and the participants guess it by asking questions to the host. There are four types of acceptable questions:

- Is it true that y is strictly larger than number x?
- Is it true that y is strictly smaller than number x?
- Is it true that y is larger than or equal to number x?
- Is it true that y is smaller than or equal to number x?

On each question the host answers truthfully, "yes" or "no".

Given the sequence of questions and answers, find any integer value of y that meets the criteria of all answers. If there isn't such value, print "Impossible".

Input

The first line of the input contains a single integer n ($1 \le n \le 10000$) — the number of questions (and answers). Next n lines each contain one question and one answer to it. The format of each line is like that: "*sign x answer*", where the *sign* is:

- ">" (for the first type queries),
- "<" (for the second type queries),
- ">=" (for the third type queries),
- "<=" (for the fourth type queries).

All values of x are integer and meet the inequation $-10^9 \le x \le 10^9$. The *answer* is an English letter "Y" (for "yes") or "N" (for "no").

Consequtive elements in lines are separated by a single space.

Output

Print any of such integers y, that the answers to all the queries are correct. The printed number y must meet the inequation $-2 \cdot 10^9 \le y \le 2 \cdot 10^9$. If there are many answers, print any of them. If such value doesn't exist, print word "Impossible" (without the quotes).

Sample test(s)

input
4 >= 1 Y < 3 N <= -3 N > 55 N
output
17

input
2 > 100 Y < -100 Y
output
Impossible

B. Art Union

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

A well-known art union called "Kalevich is Alive!" manufactures objects d'art (pictures). The union consists of n painters who decided to organize their work as follows.

Each painter uses only the color that was assigned to him. The colors are distinct for all painters. Let's assume that the first painter uses color 1, the second one uses color 2, and so on. Each picture will contain all these n colors. Adding the j-th color to the i-th picture takes the j-th painter t_{ij} units of time.

Order is important everywhere, so the painters' work is ordered by the following rules:

- Each picture is first painted by the first painter, then by the second one, and so on. That is, after the j-th painter finishes working on the picture, it must go to the $(j+1)$-th painter (if $j < n$);
- each painter works on the pictures in some order: first, he paints the first picture, then he paints the second picture and so on;
- each painter can simultaneously work on at most one picture. However, the painters don't need any time to have a rest;
- as soon as the j-th painter finishes his part of working on the picture, the picture immediately becomes available to the next painter.

Given that the painters start working at time 0, find for each picture the time when it is ready for sale.

Input

The first line of the input contains integers m, n $(1 \le m \le 50000, 1 \le n \le 5)$, where m is the number of pictures and n is the number of painters. Then follow the descriptions of the pictures, one per line. Each line contains n integers $t_{i1}, t_{i2}, ..., t_{in}$ $(1 \le t_{ij} \le 1000)$, where t_{ij} is the time the j-th painter needs to work on the i-th picture.

Output

Print the sequence of m integers $r_1, r_2, ..., r_m$, where r_i is the moment when the n-th painter stopped working on the i-th picture.

Sample test(s)

input
5 1 1 2 3 4 5
output
1 3 6 10 15

input
4 2 2 5 3 1 5 3 10 1
output
7 8 13 21

C. Booking System

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Innovation technologies are on a victorious march around the planet. They integrate into all spheres of human activity!

A restaurant called "Dijkstra's Place" has started thinking about optimizing the booking system.

There are n booking requests received by now. Each request is characterized by two numbers: c_i and p_i — the size of the group of visitors who will come via this request and the total sum of money they will spend in the restaurant, correspondingly.

We know that for each request, all c_i people want to sit at the same table and are going to spend the whole evening in the restaurant, from the opening moment at 18:00 to the closing moment.

Unfortunately, there only are k tables in the restaurant. For each table, we know r_i — the maximum number of people who can sit at it. A table can have only people from the same group sitting at it. If you cannot find a large enough table for the whole group, then all visitors leave and naturally, pay nothing.

Your task is: given the tables and the requests, decide which requests to accept and which requests to decline so that the money paid by the happy and full visitors was maximum.

Input

The first line of the input contains integer n $(1 \le n \le 1000)$ — the number of requests from visitors. Then n lines follow. Each line contains two integers: c_i, p_i $(1 \le c_i, p_i \le 1000)$ — the size of the group of visitors who will come by the i-th request and the total sum of money they will pay when they visit the restaurant, correspondingly.

The next line contains integer k $(1 \le k \le 1000)$ — the number of tables in the restaurant. The last line contains k space-separated integers: $r_1, r_2, ..., r_k$ $(1 \le r_i \le 1000)$ — the maximum number of people that can sit at each table.

Output

In the first line print two integers: m, s — the number of accepted requests and the total money you get from these requests, correspondingly.

Then print m lines — each line must contain two space-separated integers: the number of the accepted request and the number of the table to seat people who come via this request. The requests and the tables are consecutively numbered starting from 1 in the order in which they are given in the input.

If there are multiple optimal answers, print any of them.

Sample test(s)

input
3 10 50 2 100 5 30 3 4 6 9

output
2 130 2 1 3 2

D. Population Size

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarpus develops an interesting theory about the interrelation of arithmetic progressions with just everything in the world. His current idea is that the population of the capital of Berland changes over time like an arithmetic progression. Well, or like multiple arithmetic progressions.

Polycarpus believes that if he writes out the population of the capital for several consecutive years in the sequence $a_1, a_2, ..., a_n$, then it is convenient to consider the array as several arithmetic progressions, written one after the other. For example, sequence $(8, 6, 4, 2, 1, 4, 7, 10, 2)$ can be considered as a sequence of three arithmetic progressions $(8, 6, 4, 2)$, $(1, 4, 7, 10)$ and (2), which are written one after another.

Unfortunately, Polycarpus may not have all the data for the n consecutive years (a census of the population doesn't occur every year, after all). For this reason, some values of a_i may be unknown. Such values are represented by number `-1`.

For a given sequence $a = (a_1, a_2, ..., a_n)$, which consists of positive integers and values `-1`, find the minimum number of arithmetic progressions Polycarpus needs to get a. To get a, the progressions need to be written down one after the other. Values `-1` may correspond to an arbitrary positive integer and the values $a_i > 0$ must be equal to the corresponding elements of sought consecutive record of the progressions.

Let us remind you that a finite sequence c is called an arithmetic progression if the difference $c_{i+1} - c_i$ of any two consecutive elements in it is constant. By definition, any sequence of length 1 is an arithmetic progression.

Input

The first line of the input contains integer n ($1 \le n \le 2 \cdot 10^5$) — the number of elements in the sequence. The second line contains integer values $a_1, a_2, ..., a_n$ separated by a space ($1 \le a_i \le 10^9$ or $a_i = -1$).

Output

Print the minimum number of arithmetic progressions that you need to write one after another to get sequence a. The positions marked as `-1` in a can be represented by any positive integers.

Sample test(s)

input
```
9
8 6 4 2 1 4 7 10 2
```
output
```
3
```

input
```
9
-1 6 -1 2 -1 4 7 -1 2
```
output
```
3
```

input
```
5
-1 -1 -1 -1 -1
```
output
```
1
```

input
```
7
-1 -1 4 5 1 2 3
```
output
```
2
```

E. President's Path

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Good old Berland has n cities and m roads. Each road connects a pair of distinct cities and is bidirectional. Between any pair of cities, there is at most one road. For each road, we know its length.

We also know that the President will soon ride along the Berland roads from city s to city t. Naturally, he will choose one of the shortest paths from s to t, but nobody can say for sure which path he will choose.

The Minister for Transport is really afraid that the President might get upset by the state of the roads in the country. That is the reason he is planning to repair the roads in the possible President's path.

Making the budget for such an event is not an easy task. For all possible distinct pairs s, t ($s < t$) find the number of roads that lie on at least one shortest path from s to t.

Input

The first line of the input contains integers n, m ($2 \le n \le 500$, $0 \le m \le n \cdot (n - 1) / 2$) — the number of cities and roads, correspondingly. Then m lines follow, containing the road descriptions, one description per line. Each description contains three integers x_i, y_i, l_i ($1 \le x_i, y_i \le n$, $x_i \ne y_i$, $1 \le l_i \le 10^6$), where x_i, y_i are the numbers of the cities connected by the i-th road and l_i is its length.

Output

Print the sequence of $n(n-1)/2$ integers c_{12}, c_{13}, ..., c_{1n}, c_{23}, c_{24}, ..., c_{2n}, ..., $c_{n-1,n}$, where c_{st} is the number of roads that can lie on the shortest path from s to t. Print the elements of sequence c in the described order. If the pair of cities s and t don't have a path between them, then $c_{st} = 0$.

Sample test(s)

input
5 6 1 2 1 2 3 1 3 4 1 4 1 1 2 4 2 4 5 4

output
1 4 1 2 1 5 6 1 2 1