AIM Tech Round 5 (rated, Div. 1 + Div. 2)

A. Find Square

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

Consider a table of size $n \times m$, initially fully white. Rows are numbered 1 through n from top to bottom, columns 1 through m from left to right. Some square inside the table with **odd** side length was painted black. Find the center of this square.

Input
The first line contains two integers n and m ($1 \le n, m \le 115$) — the number of rows and the number of columns in the table.

The i-th of the next n lines contains a string of m characters $s_{i1}s_{i2} \ldots s_{im}$ (s_{ij} is 'W' for white cells and 'B' for black cells), describing the i-th row of the table.

Output
Output two integers r and c ($1 \le r \le n$, $1 \le c \le m$) separated by a space — the row and column numbers of the center of the black square.

Examples

input
5 6 WWBBBW WWBBBW WWBBBW WWWWWW WWWWWW
output
2 4

input
3 3 WWW BWW WWW
output
2 1

B. Unnatural Conditions

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

Let $s(x)$ be sum of digits in decimal representation of positive integer x. Given two integers n and m, find some positive integers a and b such that

- $s(a) \ge n$,
- $s(b) \ge n$,
- $s(a + b) \le m$.

Input
The only line of input contain two integers n and m ($1 \le n, m \le 1129$).

Output
Print two lines, one for decimal representation of a and one for decimal representation of b. Both numbers must not contain leading zeros and must have length no more than 2230.

Examples

input
6 5
output
6

7

input
8 16
output
35 53

Note
In the first sample, we have $n = 6$ and $m = 5$. One valid solution is $a = 6$, $b = 7$. Indeed, we have $s(a) = 6 \geq n$ and $s(b) = 7 \geq n$, and also $s(a + b) = s(13) = 4 \leq m$.

C. Rectangles

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given n rectangles on a plane with coordinates of their bottom left and upper right points. Some $(n - 1)$ of the given n rectangles have some common point. A point belongs to a rectangle if this point is strictly inside the rectangle or belongs to its boundary.

Find any point with integer coordinates that belongs to at least $(n - 1)$ given rectangles.

Input
The first line contains a single integer n ($2 \leq n \leq 132\,674$) — the number of given rectangles.

Each the next n lines contains four integers x_1, y_1, x_2 and y_2 ($-10^9 \leq x_1 < x_2 \leq 10^9$, $-10^9 \leq y_1 < y_2 \leq 10^9$) — the coordinates of the bottom left and upper right corners of a rectangle.

Output
Print two integers x and y — the coordinates of any point that belongs to at least $(n - 1)$ given rectangles.

Examples

input
3 0 0 1 1 1 1 2 2 3 0 4 1
output
1 1

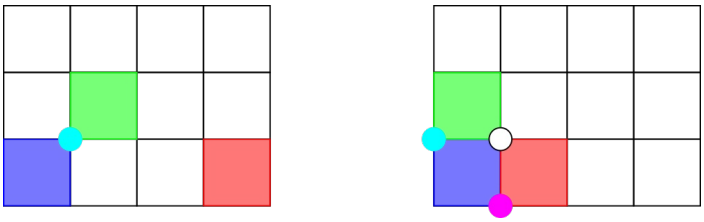
input
3 0 0 1 1 0 1 1 2 1 0 2 1
output
1 1

input
4 0 0 5 5 0 0 4 4 1 1 4 4 1 1 4 4
output
1 1

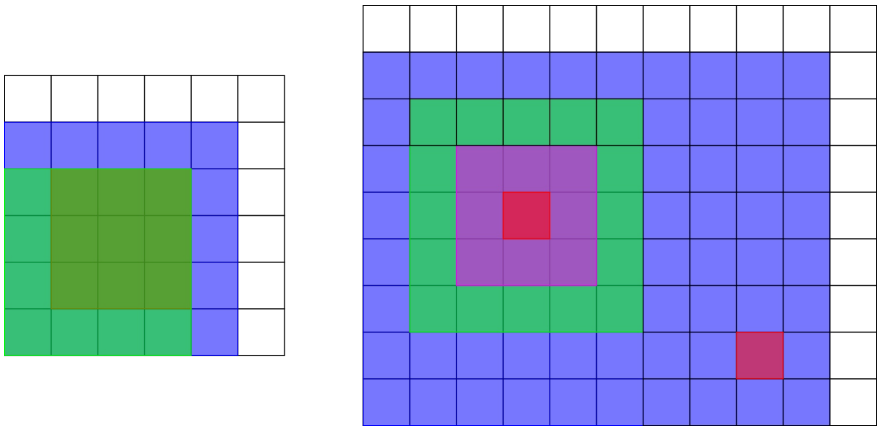
input
5 0 0 10 8 1 2 6 7 2 3 5 6 3 4 4 5 8 1 9 2
output
3 4

Note

The picture below shows the rectangles in the first and second samples. The possible answers are highlighted.



The picture below shows the rectangles in the third and fourth samples.



D. Order book

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let's consider a simplified version of order book of some stock. The order book is a list of orders (offers) from people that want to buy or sell one unit of the stock, each order is described by *direction* (BUY or SELL) and price.

At every moment of time, every SELL offer has higher price than every BUY offer.

In this problem no two ever existed orders will have the same price.

The lowest-price SELL order and the highest-price BUY order are called the *best offers*, marked with black frames on the picture below.

The presented order book says that someone wants to sell the product at price 12 and it's the best SELL offer because the other two have higher prices. The best BUY offer has price 10.

There are two possible actions in this orderbook:

1. Somebody adds a new order of some direction with some price.
2. Somebody accepts the best possible SELL or BUY offer (makes a deal). It's impossible to accept not the best SELL or BUY offer (to make a deal at worse price). After someone accepts the offer, it is removed from the orderbook forever.

It is allowed to add new BUY order only with prices less than the best SELL offer (if you want to buy stock for higher price, then instead of adding an order you should accept the best SELL offer). Similarly, one couldn't add a new SELL order with price less or equal to the best BUY offer. For example, you can't add a new offer "SELL 20" if there is already an offer "BUY 20" or "BUY 25" — in this case you just accept the best BUY offer.

You have a damaged order book log (in the beginning the are no orders in book). Every action has one of the two types:

1. "ADD p" denotes adding a new order with price p and unknown direction. The order must not contradict with orders still not removed from the order book.

2. "ACCEPT p" denotes accepting an existing best offer with price p and unknown direction.

The directions of all actions are lost. Information from the log isn't always enough to determine these directions. Count the number of ways to correctly restore all ADD action directions so that all the described conditions are satisfied at any moment. Since the answer could be large, output it modulo $10^9 + 7$. If it is impossible to correctly restore directions, then output 0.

Input
The first line contains an integer n ($1 \le n \le 363\,304$) — the number of actions in the log.

Each of the next n lines contains a string "ACCEPT" or "ADD" and an integer p ($1 \le p \le 308\,983\,066$), describing an action type and price.

All ADD actions have different prices. For ACCEPT action it is guaranteed that the order with the same price has already been added but has not been accepted yet.

Output
Output the number of ways to restore directions of ADD actions modulo $10^9 + 7$.

Examples

input
6 ADD 1 ACCEPT 1 ADD 2 ACCEPT 2 ADD 3 ACCEPT 3
output
8

input
4 ADD 1 ADD 2 ADD 3 ACCEPT 2
output
2

input
7 ADD 1 ADD 2 ADD 3 ADD 4 ADD 5 ACCEPT 3 ACCEPT 5
output
0

Note
In the first example each of orders may be BUY or SELL.

In the second example the order with price 1 has to be BUY order, the order with the price 3 has to be SELL order.

E. Restore Array

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

While discussing a proper problem A for a Codeforces Round, Kostya created a cyclic array of positive integers $a_1, a_2, \ldots, a_n$. Since the talk was long and not promising, Kostya created a new cyclic array $b_1, b_2, \ldots, b_n$ so that $b_i = (a_i \mod a_{i+1})$, where we take $a_{n+1} = a_1$. Here mod is the modulo operation. When the talk became interesting, Kostya completely forgot how array a had looked like. Suddenly, he thought that restoring array a from array b would be an interesting problem (unfortunately, not A).

Input
The first line contains a single integer n ($2 \le n \le 140582$) — the length of the array a.

The second line contains n integers $b_1, b_2, \ldots, b_n$ ($0 \le b_i \le 187126$).

Output
If it is possible to restore some array a of length n so that $b_i = a_i \mod a_{(i \mod n)+1}$ holds for all $i = 1, 2, \ldots, n$, print «YES» in

the first line and the integers $a_1, a_2, \ldots, a_n$ in the second line. All a_i should satisfy $1 \le a_i \le 10^{18}$. We can show that if an answer exists, then an answer with such constraint exists as well.

It it impossible to restore any valid a, print «NO» in one line.

You can print each letter in any case (upper or lower).

Examples

input
4 1 3 1 0
output
YES 1 3 5 2

input
2 4 4
output
NO

Note

In the first example:

- $1 \mod 3 = 1$
- $3 \mod 5 = 3$
- $5 \mod 2 = 1$
- $2 \mod 1 = 0$

F. Make Symmetrical

time limit per test: 6 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Consider a set of points A, initially it is empty. There are three types of queries:

1. Insert a point (x_i, y_i) to A. It is guaranteed that this point does not belong to A at this moment.
2. Remove a point (x_i, y_i) from A. It is guaranteed that this point belongs to A at this moment.
3. Given a point (x_i, y_i), calculate the minimum number of points required to add to A to make A symmetrical with respect to the line containing points $(0, 0)$ and (x_i, y_i). Note that these points are not actually added to A, i.e. these queries are independent from each other.

Input

The first line contains a single integer q ($1 \le q \le 2 \cdot 10^5$) — the number of queries.

Each of the following q lines describes a query and contains three integers t_i, x_i and y_i ($t_i \in \{1, 2, 3\}$, $1 \le x_i, y_i \le 112\,904$) — the type of the query and the coordinates of the point. Type 1 is addition of the point, type 2 is removal of the point, type 3 is the query to compute the minimum number of points required to make A symmetrical.

It is guaranteed that there are no more than 10^5 queries of type 3 and no more than 10^5 queries having type 1 or 2.

Output

For each query of the third type output a line with a single integer — the answer to this query.

Examples

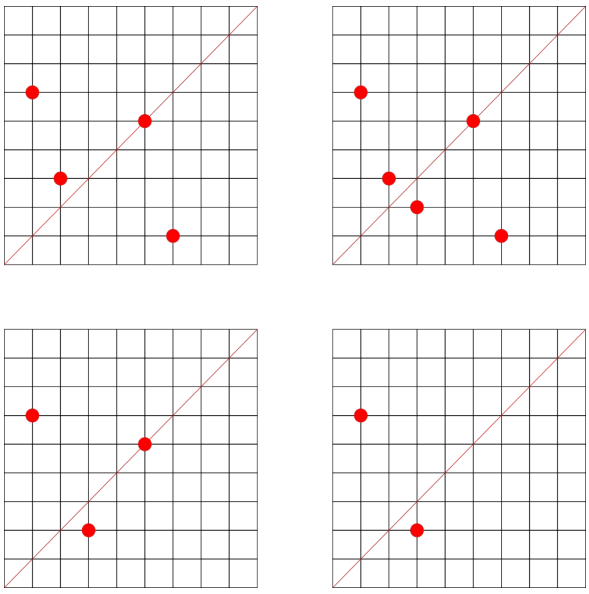
input
12 1 1 6 1 6 1 1 5 5 1 2 3 3 4 4 1 3 2 3 7 7 2 2 3 2 6 1 3 8 8 2 5 5 3 1 1
output
1 0

```
2
2
```

input
6 1 1 2 3 1 1 1 1 1 3 2 2 2 1 1 3 2 4
output
1 1 0

Note

The first example is shown on the picture below.



G. Guess the number

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem is interactive.

You should guess hidden number x which is between 1 and $M = 10004205361450474$, inclusive.

You could use up to 5 queries.

In each query, you can output an increasing sequence of $k \leq x$ integers, each between 1 and M, inclusive, and you will obtain one of the following as an answer:

- either the hidden number belongs to your query sequence, in this case you immediately win;
- or you will be given where the hidden number is located with respect to your query sequence, that is, either it is less than all numbers from the sequence, greater than all numbers from the sequence, or you will be given such an i that the hidden number x is between the i-th and the $(i+1)$-st numbers of your sequence.

See the interaction section for clarity.

Be aware that the interactor is *adaptive*, i.e. the hidden number can depend on queries the solution makes. However, it is guaranteed that for any solution the interactor works non-distinguishable from the situation when the hidden number is fixed beforehand.

Hacks are allowed only with fixed hidden number. A hack is represented by a single integer between 1 and M. In all pretests the hidden number is fixed as well.

Interaction

You can make up to 5 queries. To make a query print a number k ($1 \leq k \leq 10^4$) and then an increasing sequence $t_0 < t_1 < \ldots < t_{k-1}$ of k numbers, each between 1 and M, inclusive. **If $k > x$, you lose.**

You get one integer as a response.

- If it is -2, it means you made an invalid query or you lost. Exit immediately after receiving -2 and you will see `Wrong answer`

verdict. Otherwise you can get an arbitrary verdict because your solution will continue to read from a closed stream.

- If it is -1, you guessed the number and should terminate too.
- Otherwise you get a number i between 0 and k, inclusive, denoting where the hidden number is, with respect to the printed numbers. If $i = 0$, then $x < t_0$. If $i = k$, then $t_{k-1} < x$. Otherwise $t_{i-1} < x < t_i$.

After printing a query do not forget to output end of line and flush the output. Otherwise you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see documentation for other languages.

Example

input
2 0 -1
output
2 2 3 2 20 30 3 5 7 9

Note

In the first example the number 5 is hidden.

H. Make Square

time limit per test: 7 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

We call an array $b_1, b_2, \ldots, b_m$ good, if there exist two indices $i < j$ such that $b_i \cdot b_j$ is a perfect square.

Given an array $b_1, b_2, \ldots, b_m$, in one action you can perform one of the following:

- multiply any element b_i by any prime p;
- divide any element b_i by prime p, if b_i is divisible by p.

Let $f(b_1, b_2, \ldots, b_m)$ be the minimum number of actions needed to make the array b good.

You are given an array of n integers $a_1, a_2, \ldots, a_n$ and q queries of the form l_i, r_i. For each query output $f(a_{l_i}, a_{l_i+1}, \ldots, a_{r_i})$.

Input

The first line contains two integers n and q ($2 \le n \le 194\,598$, $1 \le q \le 1\,049\,658$) — the length of the array and the number of queries.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 5\,032\,107$) — the elements of the array.

Each of the next q lines contains two integers l_i and r_i ($1 \le l_i < r_i \le n$) — the parameters of a query.

Output

Output q lines — the answers for each query in the order they are given in the input.

Example

input
10 10 34 37 28 16 44 36 43 50 22 13 1 3 4 8 6 10 9 10 3 10 8 9 5 6 1 4 1 7 2 6
output
2

```
0
1
3
0
1
1
1
0
0
```

Note

In the first query of the first sample you can multiply second number by 7 to get 259 and multiply the third one by 37 to get 1036. Then $a_2 \cdot a_3 = 268\,324 = 518^2$.

In the second query subarray is already good because $a_4 \cdot a_6 = 24^2$.

In the third query you can divide 50 by 2 to get 25. Then $a_6 \cdot a_8 = 30^2$.
