A. Pet Friendly Office

Limits: 1 sec., 512 MiB

Solidgate is a fascinating fintech company — they love innovation, technology, and smart folks! So they invited Jerry to join them. But everyone knows: where there's Jerry, there's also Tom. To make sure he is smart enough, Solidgate decided to give Tom a very challenging task before letting him in.

Tom is having the remote interview.

Help Tom solve the following problem to earn his place at Solidgate:

An array is called **good** if it does not contain two adjacent elements that are equal. In one operation, you may choose any element of the array and either increase or decrease it by 1. Find the minimum number of operations required to make the given array good.

Input

The first line contains a single integer n — the length of the array.

The second line contains n integers $a_1, a_2, \ldots, a_n$.

Output

Print a single integer — the minimum number of operations needed to make the array good.

Constraints

$1 \le n \le 10^5$,
$0 \le a_i \le 10^9$.

Samples

Input (*stdin*)	Output (*stdout*)
4 0 0 0 0	2
7 4 4 7 7 7 4 4	3

Notes

In the first example, the given array can be transformed into the array $[0, 1, -1, 0]$ in two operations.

In the second example, the given array can be transformed into the array $[5, 4, 7, 6, 7, 3, 4]$ in three operations.

B. Consecutive Melody

Limits: 2 sec., 512 MiB

Uncle Pecos is composing a new country song. His melody is written as a sequence of notes, each represented by an integer between 0 and k, inclusive.

Unfortunately, some of the notes got smudged — they are now unknown. A sequence of notes is called consecutive if every note (except the first one) is exactly 1 greater than the previous one, and all notes are between 0 and k.

How many different ways can Uncle Pecos choose a fragment of his melody and fill in the missing notes so that it becomes consecutive?

Input

The first line contains two integers n and k — the length of the melody and the maximum possible note value.

The second line contains n integers a_i — the notes of the melody, where $a_i = -1$ denotes an unknown note, and a non-negative value of a_i represents a known note.

Output

Print a single integer — the number of different ways to pick a subarray and fill in the missing notes so that the melody becomes consecutive.

Constraints

$1 \le n, k \le 10^6$,
$-1 \le a_i \le k$.

Samples

Input (*stdin*)	Output (*stdout*)
3 4 1 -1 2	9

Notes

There are 9 different ways to obtain a consecutive subarray

- 1 way: to pick the first note.
- 5 ways: to pick the second note alone and assign any value from 0 to 4.
- 1 way: to pick the third note.
- 1 way: to pick the first and second notes and assign value 2 to the second note.
- 1 way: to pick the second and third notes and assign value 1 to the second note.

C. Cheese Trap

Limits: 2 sec., 512 MiB

Tom decided to set up a clever trap for Jerry. He placed n pieces of cheese in a circle. Each piece gives Jerry some pleasure value a_i — which can even be negative if Jerry dislikes that type of cheese.

Tom has not fixed the exact values a_i yet, but he knows Jerry's behavior very well: if the first piece available to Jerry is the i-th one, Jerry will start eating clockwise, consuming some (possibly empty) prefix of that linearized sequence to maximize his total pleasure.

Let $mx(x)$ denote the maximum sum of values on a (possibly empty) prefix of an array x.

For each i, define the sequence b_i as the cyclic shift of a that starts from its i-th element:

$b_i = [a_i, a_{i+1}, \ldots, a_n, a_1, \ldots, a_{i-1}]$, so $b_1 = a$.

It is known that if the first available piece is the i-th one, then Jerry's total pleasure will be $c_i = mx(b_i)$.

Only thing that Tom decided is an array c, so you need to help him and find **any** array a that have such array c.

Input

The first line contains a single integer n — the length of the array c.

The second line contains n integers $c_1, c_2, \ldots, c_n$ — the values of array c.

Output

Print any array a with $|a_i| \le 10^{18}$ such that the array c can be obtained from a according to the statement.

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

It can be shown that for the given constraints, there exists at least one array a satisfying the conditions.

Constraints

$1 \le n \le 2 \cdot 10^5$,
$0 \le c_i \le 10^{12}$.

Samples

Input (*stdin*)	Output (*stdout*)
3 5 2 1	3 2 -4
4 7 7 7 7	4 1 1 1

D. Medicine for Tom

Limits: 2 sec., 512 MiB

Tom the cat has caught a terrible cold after another failed attempt to catch Jerry. The doctor prescribed him n pills, where the i-th pill contains a_i milligrams of an active substance. Each time Tom takes a pill, the concentration of the substance in his blood instantly increases by a_i. However, the effect gradually fades — every hour, the concentration becomes twice smaller than before.

Tom is taking pills optimally.

Formally, let t_i be the moment (real number in hours) when the i-th pill was taken. At any moment T, the concentration of the substance in the blood can be calculated as:

$$\sum_{i:t_i<T} a_i \cdot 2^{-(T-t_i)}$$

To fully recover, Tom must keep the concentration at least x milligrams for as long as possible. Help Tom determine for how many hours he can maintain that level if he chooses the best possible schedule for taking the pills.

Input

The first line contains two integers n and x — the number of pills and the minimum concentration.
The second line contains n integers a_i — the number of substance in pills.

Output

Print a single real number — the total number of hours during which the concentration is at least x if Tom takes the pills in the optimal way.

Your answer will be considered correct if its absolute or relative error does not exceed 10^{-4}.

Constraints

$1 \le n \le 10^3$,
$1 \le x, a_i \le 10^3$.

Samples

Input (*stdin*)	Output (*stdout*)
4 7 7 4 4 7	2.3041533932

E. Unexpected Win

Limits: 2 sec., 512 MiB

Spike and Jerry decided to organize a tennis tournament among n cats to determine who is the greatest cat player. Every pair of cats plays exactly one match, and in each match exactly one cat wins.

Each cat v has a hidden skill level a_v, where a is a permutation of integers from 1 to n. The array a is not known to you in advance.

Using at most n queries, find any pair of cats (x, y) such that:

- $a_x < a_y$, and
- cat x defeated cat y in the tournament.

If no such pair exists, you must output $(-1, -1)$.

Your need to find (x, y) for t tournaments.

Input

Interaction starts by reading a single integer t — the number of tournaments.

Then, for each tournament:

The first line contains a single integer n — the number of cats.

The next n lines describe the tournament results $g_{i,j}$. Each line contains a binary string of length n, where:

- $g_{i,j} =$ `1` means that cat i defeated cat j,
- $g_{i,i} =$ `0` for all i,
- for every $i \neq j$, exactly one of $g_{i,j}$ and $g_{j,i}$ equals `1`.

You can make queries of the following form: `? v x` where $1 \le v, x \le n$. After each query, read an integer — the judge's response:

- 1 if $a_v = x$,
- 0 if $a_v \neq x$,
- -1 if you did something wrong.

After printing a query do not forget to output end of line and flush the output. The interactor is non-adaptive. The tournament results do not change during the interaction.

Output

For each tournament print `! x y` when you find a pair (x, y). Pair (x, y) can possibly be $(-1, -1)$. The judge will respond with:

- 1 if your pair is correct,
- -1 if your answer is wrong.

So if in any point you get `-1` from the judges, then you did something wrong and you should terminate your program in order to get `Wrong Answer` verdict.

Constraints

$1 \le t, n \le 100$.

Notes

In the following interaction, $n = 3$, $p = [1, 3, 2]$, $g = 010, 001, 100$.

Input	Output	Description
1		t is given.
3		n is given.
010 001 100		g is given.
	? 1 3	Ask if $p_1 = 3$.
0		The judge responds with No.
	? 1 1	Ask if $p_1 = 1$.
1		The judge responds with Yes.
	2 3	Ask if $p_2 = 3$.
1		The judge responds Yes.
	! 1 2	Answer is given. The unexpected win is pair $(1, 2)$
1		The judge responds that the answer is correct.

F. Mammy Two Shoes' Rooms

Limits: 2 sec., 512 MiB

Mammy Two Shoes has two mischievous cats — Tom and Butch.

She owns $2 \cdot n$ rooms arranged along a straight hallway, each represented by a segment (l_i, r_i).

Mammy wants to give both cats their own rooms, but Tom and Butch can't stand each other — if their rooms overlap, a fight is guaranteed! To keep peace in the house, she wants to form pairs of rooms so that the two rooms in each pair do not overlap (they may only touch at the edges).

Each room can be used in exactly one pair — no room can belong to two different pairs.

Since Mammy likes having options, she wants to know whether it's possible to make exactly n such non-conflicting pairs.

Input

Each test consists of multiple test cases. The first line contains a single integer t — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n — the number of pairs Mammy wants to form (that is, the hallway contains $2 \cdot n$ rooms).

The next $2 \cdot n$ lines of each test case each contain two integers l_i and r_i — the endpoints of the given segments representing the rooms.

Output

For each testcase print `possible` if rooms can be paired. Otherwise print `impossible`.

Constraints

$1 \le t, n \le 2 \cdot 10^5$,
$0 \le l_i < r_i \le 10^6$,
the sum of n over all test cases doesn't exceed $2 \cdot 10^5$.

Samples

Input (*stdin*)	Output (*stdout*)
4 1 4 7 7 47 3 0 1 1 6 2 3 4 5 4 5 5 6 3 0 2 0 2 1 2 2 5 3 4 5 6 3 0 1 0 2 1 2 1 2 1 4 2 3	possible possible possible impossible

G. Shifting Cheese

Limits: 2 sec., 1024 MiB

Jerry neatly arranged n pieces of cheese in a row, each having a **unique** size. He decided to measure how *unordered* the arrangement was — for him, that means counting all pairs of cheeses where a larger one is placed before a smaller one. Jerry found exactly a such pairs.

Jerry often plays tricks on Tom, but this time it was Tom's turn to take revenge. He took the first k pieces of cheese and moved them to the end of the row — without changing their internal order. After this shift, the new arrangement had exactly b *unordered* pairs.

Now Tom wonders how many different initial arrangements of the cheese could make a and b unordered pairs.

Input

The first line contains two integers n and k.
The second line contains two integers a and b.

Output

Print a single integer — the number of initial arrangements modulo prime number `998244353`.

Constraints

$2 \le n \le 500$,
$1 \le k \le \min(n-1, 20)$,
$0 \le a, b \le \frac{n \cdot (n-1)}{2}$.

Samples

Input (*stdin*)	Output (*stdout*)
`4 3` `0 3`	`1`

H. Banks

Limits: 3 sec., 512 MiB

Tom and Jerry somehow ended up running a network of banks!

The banks are connected by roads that form a **tree** — meaning there is exactly one path between any two banks. Among these banks, there are two central ones: Tom controls one, and Jerry controls the other. Both central banks can print money. It is guaranteed that Tom's and Jerry's banks are directly connected by a single road.

After inspecting the system, Tom and Jerry discovered that all of the banks are completely empty. So, they created a plan of how much money each bank should receive by the start of the year. Specifically, each bank i must end up with exactly a_i dollars. Every bank needs some money to function, so all a_i are greater than 0.

You should fulfill this plan as follows:

- First, decide how many dollars Tom's and Jerry's banks will print and place on their own accounts.
- Then, you can perform at most n transfers between banks, following this rule:
  1. You compute the number d — the maximum distance between any two non-empty banks (banks with a positive balance).
  2. You choose **any** two banks x and y at a distance of exactly d, and transfer any positive integer amount of money from bank x to bank y. Of course, bank x must have at least this amount before the transfer.

Determine how many dollars Tom and Jerry each need to print, and construct a sequence of transfers to satisfy all banks' demands. If it is impossible to do so, output `impossible`.

Input

The first line contains a single integer n — the number of banks.

The second line contains two integers — indices of the central banks (between 1 and n, inclusive). It is guaranteed that they are distinct and connected by a direct road.

Each of the next $(n-1)$ lines each contain two integers u and v — banks connected by a road.

The last line contains n positive integers $a_1, a_2, \ldots, a_n$ — the amount of money required by each bank.

Output

If it's impossible to fulfill the plan, output `impossible`.

Otherwise, in the first line output two integers — the amounts of money that the first and second central banks should print, respectively.

In the second line output an integer k — the number of transfers ($0 \le k \le n$).

On the each of the next k lines output three integers x, y, and m, denoting that m dollars must be transferred from bank x to bank y. Transfers must be printed in the order of their execution.

If there are multiple solutions, you can print any of them.

Constraints

$2 \le n \le 4000$,
$1 \le a_i \le 10^9$.

Samples

Input (*stdin*)	Output (*stdout*)
7 4 2 4 2 6 2 3 4 7 1 3 7 2 5 4 7 3 2 4 7 4	24 7 5 4 3 3 4 7 8 4 1 11 7 5 4 1 6 7

I. Small Entrance

Limits: 1 sec., 512 MiB

Tom invites exactly one of his cat friends to his house each day. On day i, the friend's height is a_i.

Since Tom cannot take guests through the main door, he leads them through the back entrance. The entrance initially has height h. Whenever a friend's height is strictly greater than the current entrance height, the friend bumps into the doorframe, and the entrance height increases by 1.

Note that the entrance height can increase at most once per day: after the first bump that day, the friend becomes more careful and does not bump again.

Determine the number of days when friend bumps into doorframe.

Input

The first line contains two integers n and h — the number of days and the initial height of the entrance.

The second line contains n integers $a_1, a_2, \ldots, a_n$ — the heights of the friends on days 1 through n.

Output

Print a single integer — the number of days when friend bumps into doorframe.

Constraints

$1 \le n \le 10^5$,
$1 \le h, a_i \le 10^9$.

Samples

Input (*stdin*)	Output (*stdout*)
7 4 3 5 6 7 6 5 10	4

Notes

In the first example, on the second, third, fourth, and seventh days, Tom's friends will bump into the doorframe.

J. Tom, Jerry and the Unshakable Blueprint

Limits: 2 sec., 512 MiB

Tom has sketched a huge map of secret tunnels and mouse-holes in the house. The map can be represented as a connected undirected graph with n vertices and m edges. Each hole is a vertex, and each bidirectional tunnel is an edge. The whole map is connected.

Jerry wants to pick a **skeleton blueprint** — a spanning tree of the map — to move silently without cycles.

Jerry believes that no matter which spanning tree he chooses, the "traffic profile" of the blueprint — i.e., the **multiset of vertex degrees within that spanning tree** — is always the same. Tom thinks Jerry is wrong and claims that some graphs admit two spanning trees with different degree multisets.

Your task is to settle their bet.

Input

The first line contains two integers n and m — the number of vertices and edges.

The next m lines each contain two integers u and v describing an undirected edge (u, v).

Output

Print `Yes` if all spanning trees of the graph have the same degree multiset, otherwise print `No`.

Constraints

$1 \le n \le 10^5$,
$1 \le m \le 2 \cdot 10^5$.
$1 \le u, v \le n$.
The graph is guaranteed to be connected and simple.

Samples

Input (*stdin*)	Output (*stdout*)
3 3 1 2 2 3 3 1	Yes
4 4 1 2 2 3 3 1 1 4	No
10 11 1 2 2 3 3 1 4 5 5 6 6 4 1 4 2 7 3 8 5 9 6 10	Yes

K. Spike and Tyke

Limits: 2 sec., 512 MiB

Spike the bulldog has found a big tasty bone divided into n parts. The i-th part has a *meatiness value* a_i.

He plans to share it with his son Tyke, but Spike loves cracking the bone apart — every time he breaks it, he feels that deep, crunchy satisfaction!

In one move, Spike chooses a contiguous section of the bone from positions l to r and cracks it at some point p ($l \le p < r$). The bone then splits into two smaller parts:

$$[a_l, a_{l+1}, \ldots, a_p] \quad \text{and} \quad [a_{p+1}, a_{p+2}, \ldots, a_r].$$

Let

$$L = \sum_{i=l}^{p} a_i, \quad R = \sum_{i=p+1}^{r} a_i.$$

When the crack happens, Spike's total satisfaction increases by $L \times R$.

After each crack, the total number of contiguous bone pieces increases by one. Pieces of size 1 cannot be cracked further — they are too small to split.

Spike wants to make exactly k cracks to maximize his total satisfaction. Find the maximum possible total satisfaction after exactly k cracks.

Input

The first line contains two integers n and k — the size of the array and the number of cracks Spike wants to make.

The second line contains n integers $a_1, a_2, \ldots, a_n$ — the meatiness values of the bone pieces.

Output

Output a single integer — the maximum possible total score after exactly k cracks.

Constraints

$1 \le k < n \le 10^5$,
$0 \le a_i \le 10^4$.

Samples

Input (*stdin*)	Output (*stdout*)
3 1 2 1 3	9
7 4 44 47 74 77 747 4 7	211092

L. Light

Limits: 1 sec., 512 MiB

One evening, Uncle Pecos sang Jerry a song about a rectangular room of length n and width m. Around the perimeter of the room, there were $2 \cdot n + 2 \cdot m$ lamps — some were shining, while others were off.

Each lamp could illuminate its entire row or column inside the room. For every cell in the room, Jerry knew how many lamps were lighting it — its illumination level.

Curious as always, Jerry started wondering: what configuration of lamps could have produced this pattern of light?

Your task is to help him find any possible configuration of lamps that matches the given illumination matrix.

First, output the states of the lamps placed along the length (rows), and then those placed along the width (columns).

Input

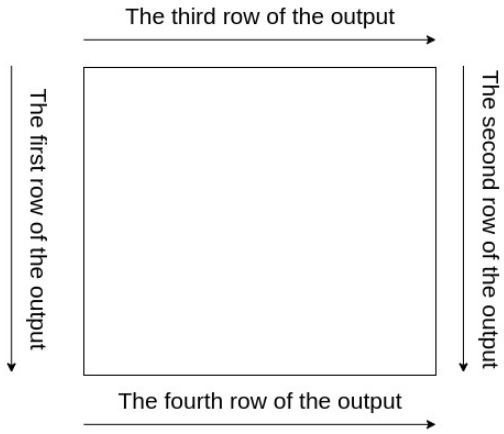
The first line contains two integers n and m.

The next n lines contain m integers each — illumination level of each cell.

Output

In four lines output any configuration of lamps that matches the given illumination matrix.

First, print the states of the lamps placed vertically, from the first to the n-th row. Then, print the states of the lamps placed horizontally, from the first to the m-th column.



If multiple configurations are possible, output any of them.

Constraints

$1 \le n, m \le 100$,
$0 \le a_{i,j} \le 4$,
at least one solution exists for a given input.

Samples

Input (*stdin*)	Output (*stdout*)
2 3 1 2 1 0 1 0	1 0 0 0 0 0 0 0 1 0
4 5 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0