

Lookery Cup 2015

A. Face Detection

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

The developers of Lookery have to write an efficient algorithm that detects faces on a picture. Unfortunately, they are currently busy preparing a contest for you, so you will have to do it for them.

In this problem an image is a rectangular table that consists of lowercase Latin letters. A face on the image is a 2×2 square, such that from the four letters of this square you can make word "face".

You need to write a program that determines the number of faces on the image. The squares that correspond to the faces can overlap.

Input

The first line contains two space-separated integers, n and m ($1 \leq n, m \leq 50$) — the height and the width of the image, respectively.

Next n lines define the image. Each line contains m lowercase Latin letters.

Output

In the single line print the number of faces on the image.

Examples

| |
|-------------------------------------|
| input |
| 4 4 xxxx xfax xcex xxxx |
| output |
| 1 |
| input |
| 4 2 xx cf ae xx |
| output |
| 1 |
| input |
| 2 3 fac cef |
| output |
| 2 |
| input |
| 1 4 face |
| output |
| 0 |

Note

In the first sample the image contains a single face, located in a square with the upper left corner at the second line and the second column:

In the second sample the image also contains exactly one face, its upper left corner is at the second row and the first column.

In the third sample two faces are shown:



In the fourth sample the image has no faces on it.

B. Looksery Party

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

The Looksery company, consisting of n staff members, is planning another big party. Every employee has his phone number and the phone numbers of his friends in the phone book. Everyone who comes to the party, sends messages to his contacts about how cool it is. At the same time everyone is trying to spend as much time on the fun as possible, so they send messages to everyone without special thinking, moreover, each person even sends a message **to himself or herself**.

Igor and Max, Looksery developers, started a dispute on how many messages each person gets. Igor indicates n numbers, the i -th of which indicates how many messages, in his view, the i -th employee is going to take. If Igor guesses correctly at least one of these numbers, he wins, otherwise Max wins.

You support Max in this debate, so you need, given the contact lists of the employees, to determine whether there is a situation where Igor loses. Specifically, you need to determine which employees should come to the party, and which should not, so after all the visitors send messages to their contacts, each employee received a number of messages that is different from what Igor stated.

Input

The first line contains a single integer n ($1 \leq n \leq 100$) — the number of employees of company Looksery.

Next n lines contain the description of the contact lists of the employees. The i -th of these lines contains a string of length n , consisting of digits zero and one, specifying the contact list of the i -th employee. If the j -th character of the i -th string equals 1, then the j -th employee is in the i -th employee's contact list, otherwise he isn't. It is guaranteed that the i -th character of the i -th line is always equal to 1.

The last line contains n space-separated integers: a_1, a_2, \dots, a_n ($0 \leq a_i \leq n$), where a_i represents the number of messages that the i -th employee should get according to Igor.

Output

In the first line print a single integer m — the number of employees who should come to the party so that Igor loses the dispute.

In the second line print m space-separated integers — the numbers of these employees in an arbitrary order.

If Igor wins the dispute in any case, print -1.

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

Examples

| |
|--|
| input |
| 3 101 010 001 0 1 2 |
| output |
| 1 1 |
| input |
| 1 1 1 |
| output |
| 0 |
| input |
| 4 1111 0101 1110 0001 1 0 1 0 |
| output |
| 4 1 2 3 4 |

Note

In the first sample Igor supposes that the first employee will receive 0 messages. Since he isn't contained in any other contact list he must come to the party in order to receive one message from himself. If he is the only who come to the party then he will receive 1 message, the second employee will receive 0 messages and the third will also receive 1 message. Thereby Igor won't guess any number.

In the second sample if the single employee comes to the party he receives 1 message and Igor wins, so he shouldn't do it.

In the third sample the first employee will receive 2 messages, the second — 3, the third — 2, the fourth — 3.

C. The Game Of Parity

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

There are n cities in Westeros. The i -th city is inhabited by a_i people. Daenerys and Stannis play the following game: in one single move, a player chooses a certain town and burns it to the ground. Thus all its residents, sadly, die. Stannis starts the game. The game ends when Westeros has exactly k cities left.

The prophecy says that if the total number of surviving residents is even, then Daenerys wins: Stannis gets beheaded, and Daenerys rises on the Iron Throne. If the total number of surviving residents is odd, Stannis wins and everything goes in the completely opposite way.

Lord Petyr Baelish wants to know which candidates to the throne he should support, and therefore he wonders, which one of them has a winning strategy. Answer to this question of Lord Baelish and maybe you will become the next Lord of Harrenholl.

Input

The first line contains two positive space-separated integers, n and k ($1 \leq k \leq n \leq 2 \cdot 10^5$) — the initial number of cities in Westeros and the number of cities at which the game ends.

The second line contains n space-separated positive integers a_i ($1 \leq a_i \leq 10^6$), which represent the population of each city in Westeros.

Output

Print string "Daenerys" (without the quotes), if Daenerys wins and "Stannis" (without the quotes), if Stannis wins.

Examples

| |
|---------------|
| input |
| 3 1 1 2 1 |
| output |
| Stannis |

| |
|---------------|
| input |
| 3 1 2 2 1 |
| output |
| Daenerys |

| |
|-------------------------|
| input |
| 6 3 5 20 12 7 14 101 |
| output |
| Stannis |

Note

In the first sample Stannis will use his move to burn a city with two people and Daenerys will be forced to burn a city with one resident. The only survivor city will have one resident left, that is, the total sum is odd, and thus Stannis wins.

In the second sample, if Stannis burns a city with two people, Daenerys burns the city with one resident, or vice versa. In any case, the last remaining city will be inhabited by two people, that is, the total sum is even, and hence Daenerys wins.

D. Haar Features

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

The first algorithm for detecting a face on the image working in realtime was developed by Paul Viola and Michael Jones in 2001. A part of the algorithm is a procedure that computes *Haar features*. As part of this task, we consider a simplified model of this concept.

Let's consider a rectangular image that is represented with a table of size $n \times m$. The table elements are integers that specify the brightness of each pixel in the image.

A *feature* also is a rectangular table of size $n \times m$. Each cell of a *feature* is painted black or white.

To calculate the value of the given feature at the given image, you must perform the following steps. First the table of the feature is put over the table of the image (without rotations or reflections), thus each pixel is entirely covered with either black or white cell. The *value* of a feature in the image is the value of $W - B$, where W is the total brightness of the pixels in the image, covered with white feature cells, and B is the total brightness of the pixels covered with black feature cells.

Some examples of the most popular Haar features are given below.



Your task is to determine the number of operations that are required to calculate the feature by using the so-called *prefix rectangles*.

A *prefix rectangle* is any rectangle on the image, the upper left corner of which coincides with the upper left corner of the image.

You have a variable *value*, whose value is initially zero. In one *operation* you can count the sum of pixel values at any prefix rectangle, multiply it by any integer and add to variable *value*.

You are given a feature. It is necessary to calculate the minimum number of *operations* required to calculate the values of this attribute at an arbitrary image. For a better understanding of the statement, read the explanation of the first sample.

Input

The first line contains two space-separated integers n and m ($1 \leq n, m \leq 100$) — the number of rows and columns in the feature.

Next n lines contain the description of the feature. Each line consists of m characters, the j -th character of the i -th line equals to "W", if this element of the feature is white and "B" if it is black.

Output

Print a single number — the minimum number of operations that you need to make to calculate the value of the feature.



Examples

| |
|---|
| input |
| 6 8 BBBBBBBB BBBBBBBB BBBBBBBB WWWWWWWW WWWWWWWW WWWWWWWW |
| output |
| 2 |
| input |
| 3 3 WBW BWW WWW |
| output |
| 4 |
| input |
| 3 6 WWBBWW WWBBWW WWBBWW |
| output |
| 3 |
| input |
| |

| |
|-------------------------------------|
| 4 4 BBBB BBBB BBBB BBBW |
| output |
| 4 |

Note

The first sample corresponds to feature *B*, the one shown in the picture. The value of this feature in an image of size 6×8 equals to the difference of the total brightness of the pixels in the lower and upper half of the image. To calculate its value, perform the following two *operations*:

1. add the sum of pixels in the prefix rectangle with the lower right corner in the 6-th row and 8-th column with coefficient 1 to the variable *value* (the rectangle is indicated by a red frame);

2. add the number of pixels in the prefix rectangle with the lower right corner in the 3-rd row and 8-th column with coefficient - 2 and variable *value*.


Thus, all the pixels in the lower three rows of the image will be included with factor 1, and all pixels in the upper three rows of the image will be included with factor $1 - 2 = -1$, as required.

E. Sasha Circle

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

Berlanders like to eat cones after a hard day. Misha Square and Sasha Circle are local authorities of Berland. Each of them controls its points of cone trade. Misha has n points, Sasha — m . Since their subordinates constantly had conflicts with each other, they decided to build a fence in the form of a circle, so that the points of trade of one businessman are strictly inside a circle, and points of the other one are strictly outside. It doesn't matter which of the two gentlemen will have his trade points inside the circle.

Determine whether they can build a fence or not.

Input

The first line contains two integers n and m ($1 \leq n, m \leq 10000$), numbers of Misha's and Sasha's trade points respectively.

The next n lines contains pairs of space-separated integers M_x, M_y ($-10^4 \leq M_x, M_y \leq 10^4$), coordinates of Misha's trade points.

The next m lines contains pairs of space-separated integers S_x, S_y ($-10^4 \leq S_x, S_y \leq 10^4$), coordinates of Sasha's trade points.

It is guaranteed that all $n + m$ points are distinct.

Output

The only output line should contain either word "YES" without quotes in case it is possible to build a such fence or word "NO" in the other case.

Examples

| input |
|-----------------------------------|
| 2 2 -1 0 1 0 0 -1 0 1 |
| output |
| NO |

| input |
|---|
| 4 4 1 0 0 1 -1 0 0 -1 1 1 -1 1 -1 -1 1 -1 |
| output |
| YES |

Note

In the first sample there is no possibility to separate points, because any circle that contains both points $(-1, 0)$, $(1, 0)$ also contains at least one point from the set $(0, -1)$, $(0, 1)$, and vice-versa: any circle that contains both points $(0, -1)$, $(0, 1)$ also contains at least one point from the set $(-1, 0)$, $(1, 0)$

In the second sample one of the possible solution is shown below. Misha's points are marked with red colour and Sasha's are marked with blue.



F. Yura and Developers

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

Yura has a team of k developers and a list of n tasks numbered from 1 to n . Yura is going to choose some tasks to be done this week. Due to strange Lookery habits the numbers of chosen tasks should be a segment of consecutive integers containing **no less than 2 numbers**, i. e. a sequence of form $l, l + 1, \dots, r$ for some $1 \leq l < r \leq n$.

Every task i has an integer number a_i associated with it denoting how many man-hours are required to complete the i -th task. Developers are not self-confident at all, and they are actually afraid of difficult tasks. Knowing that, Yura decided to pick up a hardest task (the one that takes the biggest number of man-hours to be completed, among several hardest tasks with same difficulty level he chooses arbitrary one) and complete it on his own. So, if tasks with numbers $[l, r]$ are chosen then the developers are left with $r - l$ tasks to be done by themselves.

Every developer can spend any integer amount of hours over any task, but when they are done with the whole assignment there should be exactly a_i man-hours spent over the i -th task.

The last, but not the least problem with developers is that one gets angry if he works more than another developer. A set of tasks $[l, r]$ is considered *good* if it is possible to find such a distribution of work that allows to complete all the tasks and to have every developer working for the same amount of time (amount of work performed by Yura doesn't matter for other workers as well as for him).

For example, let's suppose that Yura have chosen tasks with following difficulties: $a = [1, 2, 3, 4]$, and he has three developers in his disposal. He takes the hardest fourth task to finish by himself, and the developers are left with tasks with difficulties $[1, 2, 3]$. If the first one spends an hour on the first task and an hour on the third one, the second developer spends two hours on the second task and the third developer spends two hours on the third task, then they are done, since every developer worked exactly for two hours and every task has been worked over for the required amount of time. As another example, if the first task required two hours instead of one to be completed then it would be impossible to assign the tasks in a way described above.

Besides work, Yura is fond of problem solving. He wonders how many pairs (l, r) ($1 \leq l < r \leq n$) exists such that a segment $[l, r]$ is *good*? Yura has already solved this problem, but he has no time to write the code. Please, help Yura and implement the solution for this problem.

Input

The first line of input contains two positive integers: n and k ($1 \leq n \leq 300\,000$, $1 \leq k \leq 1\,000\,000$), the number of tasks in the list and the number of developers in Yura's disposal.

The second line contains n integers a_i ($1 \leq a_i \leq 10^9$).

Output

Output a single integer — the number of pairs (l, r) satisfying the conditions from the statement.

Examples

| input |
|----------------|
| 4 3 1 2 3 4 |
| output |
| 3 |

| input |
|----------------|
| 4 2 4 4 7 4 |
| output |
| 6 |

Note

In the first sample there are three good segments:

- $[1;3]$ — the hardest task requires 3 man-hours, so there are tasks left that require 1 and 2 man-hours. A solution is to make first developer work on the first task for an hour, while second and third developers work on the second task. Each developer works exactly one hour.
- $[1;4]$ — the hardest task requires 4 man-hours, so there are tasks left that require 1, 2 and 3 man-hours. If the first developer spends an hour on the first task and an hour on the third one, the second developer spends two hours on the second task and the third developer spends two hours on the third task, then they are done, since every developer worked exactly for two hours.
- $[3;4]$ — the hardest task requires 4 man-hours, so there is only one task left that requires 3 man-hours. A solution is to make

each developer work for an hour.

G. Happy Line

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

Do you like summer? Residents of Berland do. They especially love eating ice cream in the hot summer. So this summer day a large queue of n Berland residents lined up in front of the ice cream stall. We know that each of them has a certain amount of berland dollars with them. The residents of Berland are nice people, so each person agrees to swap places with the person right behind him for just 1 dollar. More formally, if person a stands just behind person b , then person a can pay person b 1 dollar, then a and b get swapped. Of course, if person a has zero dollars, he can not swap places with person b .

Residents of Berland are strange people. In particular, they get upset when there is someone with a *strictly smaller* sum of money in the line in front of them.

Can you help the residents of Berland form such order in the line so that they were all *happy*? A *happy* resident is the one who stands first in the line or the one in front of who another resident stands with *not less* number of dollars. Note that the people of Berland are people of honor and they agree to swap places only in the manner described above.

Input

The first line contains integer n ($1 \leq n \leq 200\,000$) — the number of residents who stand in the line.

The second line contains n space-separated integers a_i ($0 \leq a_i \leq 10^9$), where a_i is the number of Berland dollars of a man standing on the i -th position in the line. The positions are numbered starting from the *end* of the line.

Output

If it is impossible to make all the residents happy, print ": (" without the quotes. Otherwise, print in the single line n space-separated integers, the i -th of them must be equal to the number of money of the person on position i in the new line. If there are multiple answers, print any of them.

Examples

| |
|---------------|
| input |
| 2 11 8 |
| output |
| 9 10 |

| |
|------------------|
| input |
| 5 10 9 7 10 6 |
| output |
| : (|

| |
|---------------|
| input |
| 3 12 3 3 |
| output |
| 4 4 10 |

Note

In the first sample two residents should swap places, after that the first resident has 10 dollars and he is at the head of the line and the second resident will have 9 coins and he will be at the end of the line.

In the second sample it is impossible to achieve the desired result.

In the third sample the first person can swap with the second one, then they will have the following numbers of dollars: 4 11 3, then the second person (in the new line) swaps with the third one, and the resulting numbers of dollars will equal to: 4 4 10. In this line everybody will be happy.

H. Degenerate Matrix

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

The *determinant* of a matrix 2×2 is defined as follows:

$$\det \begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$$

A matrix is called *degenerate* if its determinant is equal to zero.

The *norm* $||A||$ of a matrix A is defined as a maximum of absolute values of its elements.

You are given a matrix $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Consider any degenerate matrix B such that norm $||A - B||$ is minimum possible. Determine $||A - B||$.

Input

The first line contains two integers a and b ($|a|, |b| \leq 10^9$), the elements of the first row of matrix A .

The second line contains two integers c and d ($|c|, |d| \leq 10^9$) the elements of the second row of matrix A .

Output

Output a single real number, the minimum possible value of $||A - B||$. Your answer is considered to be correct if its absolute or relative error does not exceed 10^{-9} .

Examples

| |
|---------------|
| input |
| 1 2 3 4 |
| output |
| 0.2000000000 |

| |
|---------------|
| input |
| 1 0 0 1 |
| output |
| 0.5000000000 |

Note

In the first sample matrix B is $\begin{pmatrix} 1.2 & 1.8 \\ 2.8 & 4.2 \end{pmatrix}$

In the second sample matrix B is $\begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix}$