Problem A. Axis-Aligned Area

Time limit: 2 seconds
Memory limit: 1024 megabytes

Alex has four sticks with positive integer lengths a_1, a_2, a_3, and a_4 ($a_1 \le a_2 \le a_3 \le a_4$).

She wants to place them on a plane in such a way that each stick is parallel to one of the two coordinate axes, and the area enclosed by these sticks is as large as possible.

Find this maximum enclosed area.

Input

The input contains four positive integers a_1, a_2, a_3, and a_4, each on a separate line, denoting the lengths of the sticks in non-decreasing order ($1 \le a_1 \le a_2 \le a_3 \le a_4 \le 100$).

Output

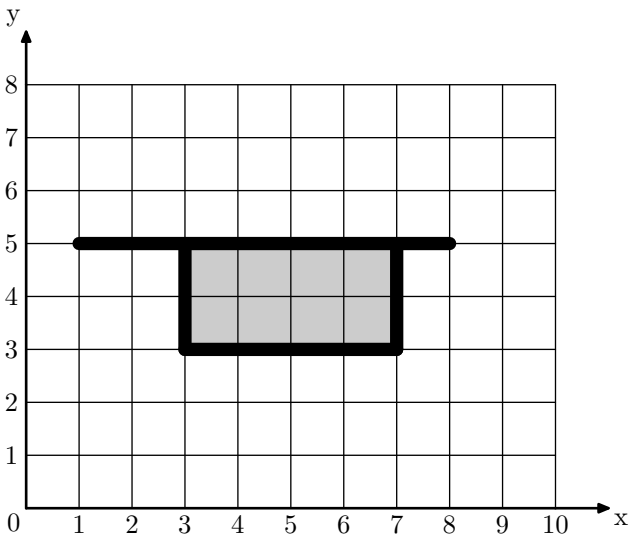
Print the maximum area that can be enclosed.

Examples

standard input	standard output
2 2 4 7	8
10 10 10 10	100

Note

Here is one optimal way to place the sticks on the plane for the first example:



The enclosed area is shaded in gray.

Problem B. Based Zeros

Time limit: 2 seconds
Memory limit: 1024 megabytes

Barbara has always known how to represent integers in the decimal numeral system (base ten), using digits $0, 1, 2, \ldots, 9$. Recently she has learned that for any integer base $b \geq 2$, she can also represent integers in base b, using symbols with values from 0 to $b - 1$, inclusive, as digits.

Barbara's favorite digit is 0. Luckily, it looks the same in all bases.

Today Barbara is playing with a positive integer n. Now she wonders: in what bases does the representation of n contain the biggest number of zeros? Help her to find all such bases.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 1000$). The description of the test cases follows.

The only line of each test case contains a single integer n ($2 \leq n \leq 10^{18}$).

Output

For each test case, in the first line, print two integers k and m, denoting the maximum number of zeros the representation of n can have in any integer base, and the number of such bases, respectively.

In the second line, print m integers $b_1, b_2, \ldots, b_m$, denoting all such bases in increasing order ($2 \leq b_1 < b_2 < \cdots < b_m \leq n$).

Example

standard input	standard output
3 11 1007 239	1 3 2 3 11 2 2 3 10 1 4 2 6 15 239

Note

Here are the representations with the maximum number of zeros for the example test cases:

- $11 = 1011_2 = 102_3 = 10_{11}$ (one zero);
- $1007 = 1101022_3 = 1007_{10}$ (two zeros);
- $239 = 11101111_2 = 1035_6 = 10\mathtt{E}_{15} = 10_{239}$ (one zero).

In the $239 = 10\mathtt{E}_{15}$ representation, `E` stands for a digit with the value of 14.

Problem C. Colorful Village

Time limit: 2 seconds
Memory limit: 1024 megabytes

Colorful Village is a popular tourist destination. It has $2n$ houses, numbered from 1 to $2n$. Every house has one of n colors, numbered from 1 to n. Coincidentally, for each of the n colors, exactly two houses are colored into it.

There are $2n - 1$ bidirectional roads in Colorful Village. Each road connects two different houses, and it is possible to reach any house from any other house using these roads.

Catherine is planning a trip to Colorful Village. Her time is limited, so she wants to choose a set S of n houses to visit, with exactly one house of each color. However, since Catherine also needs to move between houses, the set of houses she is going to visit must be connected. In other words, it must be possible to reach any house in S from any other house in S using the roads, and only visiting houses in S on the way.

Help Catherine to find a connected set S of n houses, one of each color, or report that no such set exists.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^5$). The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 10^5$).

The second line contains $2n$ integers $c_1, c_2, \ldots, c_{2n}$, denoting the colors of the houses in Colorful Village ($1 \le c_i \le n$). Every integer from 1 to n appears exactly twice in this line.

The i-th of the following $2n - 1$ lines contains two integers u_i and v_i, denoting the houses connected by the i-th road ($1 \le u_i, v_i \le 2n$; $u_i \ne v_i$).

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, print a single integer -1 if the required set of houses does not exist.

Otherwise, print n distinct integers $s_1, s_2, \ldots, s_n$ in any order, denoting a connected set S of n houses, one of each color ($1 \le s_i \le 2n$). If there are multiple answers, print any of them.

Example

standard input	standard output
2 4 1 3 1 3 4 4 2 2 1 6 5 3 2 4 7 1 7 5 5 8 2 5 3 1 1 2 2 3 3 1 2 2 3 3 4 4 5 5 6	2 3 5 7 -1

Problem D. Divisibility Trick

Time limit: 2 seconds
Memory limit: 1024 megabytes

Dmitry has recently learned a simple rule to check if an integer is divisible by 3. An integer is divisible by 3 if the sum of its digits is divisible by 3.

Later he also learned that the same rule can be used to check if an integer is divisible by 9. An integer is divisible by 9 if the sum of its digits is divisible by 9.

Dmitry's elder sister Daria wants to trick him by showing that the same rule can be applied to any divisor d. To do this, she wants to show Dmitry an example of a positive integer n such that n is divisible by d, and the sum of the digits of n is also divisible by d. Help her to find such a number.

Input

The only line contains a single integer d ($1 \le d \le 1000$).

Output

Print a positive integer n divisible by d such that the sum of its digits is also divisible by d.

The value of n must consist of at most 10^6 digits and must not have leading zeroes. It can be shown that such an integer always exists. If there are multiple answers, print any of them.

Examples

standard input	standard output
3	3
13	1898
1	239

Note

In the first example, 3 is divisible by 3, and its sum of digits, 3, is also divisible by 3.

In the second example, 1898 is divisible by 13, and its sum of digits, $1 + 8 + 9 + 8 = 26$, is also divisible by 13.

In the third example, any positive integer satisfies the conditions.

Problem E. Every Queen

Time limit:	2 seconds
Memory limit:	1024 megabytes

There are n chess queens on an infinite grid. They are placed in squares with coordinates $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. Your task is to find a square that all queens attack, or report that no such square exists.

A queen in square (x_i, y_i) attacks square (x, y) if at least one of the following conditions is satisfied:

- $x_i = x$;
- $y_i = y$;
- $|x_i - x| = |y_i - y|$.

Note that in this problem, the queens do not block each other. For example, if there are queens in squares $(1, 1)$ and $(2, 2)$, both of them attack square $(3, 3)$. Moreover, you can choose a square that already contains a queen. For example, square $(1, 1)$ would be a valid answer in this case.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^5$). The description of the test cases follows.

The first line of each test case contains a single integer n, denoting the number of queens ($1 \le n \le 10^5$).

The i-th of the following n lines contains two integers x_i and y_i, denoting the coordinates of the square containing the i-th queen ($-10^8 \le x_i, y_i \le 10^8$). No two queens share the same square.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, if an answer exists, print "`YES`" in the first line. Then, in the second line, print two integers x and y, denoting the coordinates of a square attacked by every queen ($-10^9 \le x, y \le 10^9$).

If no such square exists, print a single line containing "`NO`" instead.

It can be shown that if an answer exists, there also exists an answer that satisfies $-10^9 \le x, y \le 10^9$. If there are multiple answers, print any of them.

Example

standard input	standard output
3 2 1 1 2 2 4 0 1 1 0 3 1 4 0 5 0 1 1 0 1 2 2 2 4 2	YES 1 1 NO YES -1 2

Problem F. First Solved, Last Coded

Time limit: 2 seconds
Memory limit: 1024 megabytes

In ICPC, teamwork is everything. That's why everyone on your team has a well-defined role: Sol the Solver can solve any problem in the problem set, Codie the Coder can implement any solution that Sol comes up with, and you... are the glue that holds everything together. Sol and Codie are very picky about the order of problems they would solve/code, and your job is to satisfy their preferences.

There will be n problems in the upcoming contest, and you know the general topic of each problem: greedy, geometry, graphs, etc. For simplicity, we will represent each topic with an integer from 1 to n. These integers don't have to be distinct, that is, multiple problems in the contest can have the same topic.

Sol wants to solve problems in a specific order of topics: first, the problem with the topic a_1, after that, the problem with the topic a_2, ..., and finally, the problem with the topic a_n. Codie also has a preference list: $b_1, b_2, \ldots, b_n$, only willing to code problems in that order of topics.

Your job during the contest will be to take solution sheets from Sol and hand them to Codie in the correct order. As your team only has one table to work with, you don't have enough space to arrange all the solutions neatly. Thus, you came up with the following workflow: you will ask Sol for solutions (who will hand them to you in order $a_1, a_2, \ldots, a_n$), store them in a stack on your part of the table, and hand them to Codie to code (in order $b_1, b_2, \ldots, b_n$).

More formally, at any moment during the contest, you have (at most) two actions you can make:

- If there are still any unsolved problems remaining, ask Sol for another solution and put it on top of your stack of solution sheets. This action is denoted by the character '`S`'.
- If your stack is not empty, take the solution sheet from the top of your stack and give it to Codie to implement. This action is denoted by the character '`C`'.

For the given lists of Sol's and Codie's preferences, find a sequence of actions that ensures that all problems are solved and coded in the correct order. Consider all solving and coding times insignificant — managing solution sheets is a much harder and more important job anyway.

Input

The first line contains a single integer n, denoting the number of problems in the contest ($1 \le n \le 100$).

The second line contains n integers $a_1, a_2, \ldots, a_n$, denoting Sol's preferred order of topics ($1 \le a_i \le n$).

The third line contains n integers $b_1, b_2, \ldots, b_n$, denoting Codie's preferred order of topics ($1 \le b_i \le n$).

The given lists are equal as multisets: every integer occurs the same number of times in A and in B.

Output

If your task is impossible, print "`NO`". Otherwise, print "`YES`" on the first line, followed by the sequence of actions on the second line: a string consisting of $2n$ characters '`S`' or '`C`' (n of each), describing your actions in order.

You are not allowed to ask Sol for more solutions if all n problems have already been solved, or give Codie a solution with the wrong topic. If there are multiple answers, print any of them.

Examples

standard input	standard output
4 4 1 2 2 1 2 4 2	YES SSCSCCSC
3 2 3 1 1 2 3	NO

Problem G. Game of Nim

Time limit: 2 seconds
Memory limit: 1024 megabytes

Georgiy and Gennady are playing a new game they have just invented after learning about the classical game of Nim. This game is played with n stones and consists of two stages.

In the setup stage, Georgiy chooses a positive integer $p < n$ and puts a pile of p stones on a game field. After that, Gennady forms an arbitrary number of piles, each containing an arbitrary number of stones, using all $(n - p)$ stones not used by Georgiy.

For example, if $n = 10$ and $p = 2$, Gennady can form:

- 8 piles of 1 stone each,
- or one pile of 5 stones and one pile of 3 stones,
- or 2 piles of 2 stones and 4 piles of 1 stone,
- or one pile of 8 stones,
- etc.

After the setup stage, the Nim stage comes. At this stage, the game of Nim is played. Players take turns, starting from Georgiy. On each turn, the player must remove at least one stone and may remove any number of stones, provided they all come from the same pile. The player who takes the last stone wins at Nim and, consequently, wins the entire game.

Georgiy and Gennady have just started the game, and now it is the middle of the setup stage: Georgiy has already made his pile of p stones, but Gennady has not divided the remaining $(n-p)$ stones into piles yet. Now Gennady wants to know what his chances are to win the game.

You are to calculate the number of ways Gennady can divide $(n - p)$ stones into piles so that he will win the game, assuming that both players will play Nim optimally.

As you may know, according to the Sprague-Grundy theory, Gennady will win if and only if the bitwise exclusive or (XOR) of all pile sizes (the pile of p stones and all piles made from the remaining $(n - p)$ stones) is equal to zero.

Since the answer can be large, please calculate it modulo m. Two ways are considered to be different if the corresponding multisets of pile sizes are different — that is, the order of piles does not matter.

Input

The only line contains three integers n, p, and m, denoting the total number of stones in the game, the size of the pile chosen by Georgiy, and the value of the modulus ($1 \le p < n \le 500$; $2 \le m \le 10^9$).

Output

Print the number of ways Gennady can divide the remaining $(n - p)$ stones into piles so that he will win the game, modulo m.

Examples

standard input	standard output
8 3 1000	2
5 2 1000	0

Note

In the first example, the only two winning ways for Gennady to divide the remaining 5 stones are:

- one pile of 3 stones and 2 piles of 1 stone,
- or one pile of 2 stones and 3 piles of 1 stone.

In the second example, no matter how Gennady divides the remaining 3 stones, he is bound to lose.

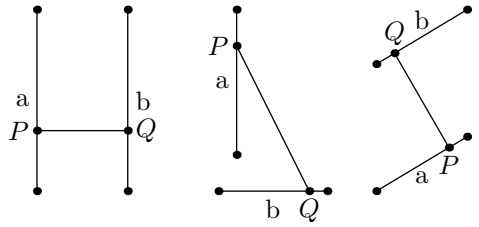
Problem H. H-Shaped Figures

Time limit:	2 seconds
Memory limit:	1024 megabytes

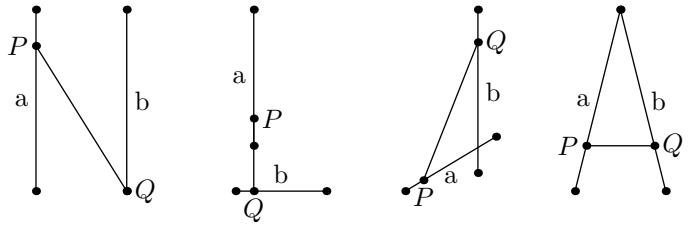
After a huge success of the last year's "K-Shaped Figures" problem, we've come up with an innovative "H-Shaped Figures" problem for this year. And we have some plans for the next 24 years.

Let's say that three segments PQ, a, and b on a plane form an *H-shaped figure* if:

- point P lies strictly inside segment a, and segments PQ and a are not collinear;
- point Q lies strictly inside segment b, and segments PQ and b are not collinear;
- segments a and b do not have common points.



Valid H-shaped figures



Invalid H-shaped figures

You are given the coordinates of points P and Q, along with n candidate segments for a and b. Note that some of the given segments may coincide, but they should still be treated as different segments.

Find the number of possible ways to choose one of the given n segments as a and another one as b to form an H-shaped figure along with the given segment PQ.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^5$). The description of the test cases follows.

The first line of each test case contains four integers x_P, y_P, x_Q, y_Q, denoting the coordinates of points P and Q ($-10^9 \le x_P, y_P, x_Q, y_Q \le 10^9$). Points P and Q do not coincide.

The second line contains a single integer n, denoting the number of candidate segments ($2 \le n \le 2 \cdot 10^5$).

The i-th of the following n lines contains four integers $x_{i,1}, y_{i,1}, x_{i,2}, y_{i,2}$, denoting the coordinates of the endpoints of the i-th segment ($-10^9 \le x_{i,1}, y_{i,1}, x_{i,2}, y_{i,2} \le 10^9$). All segments have positive lengths.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print the number of ways to form an H-shaped figure using the given segment PQ and two of the candidate segments.

Example

standard input	standard output
1 0 0 4 0 8 0 0 2 1 -1 -1 2 2 3 3 5 -3 0 2 6 -1 2 -2 5 1 -1 1 3 -3 -1 0 2 0 -1 -1 2 2	6

Problem I. Intersegment Activation

Time limit: 2 seconds
Memory limit: 1024 megabytes

This is an interactive problem.

There is an array of n cells, numbered from 1 to n. For each pair of integers (i, j), where $1 \le i \le j \le n$, there is a barrier covering all cells from i to j, inclusive. Each barrier is either *active* or *inactive*. A cell is *visible* if there are no active barriers that cover it. Otherwise, the cell is *invisible*.

The state of each barrier is unknown to you. All you can observe is the number of visible cells. But you can flip the state of any barrier: if it's active, it turns inactive, and the other way around. Your task is to make all barriers inactive, so that all cells become visible.

Interaction Protocol

First, read an integer n, denoting the number of cells ($1 \le n \le 10$).

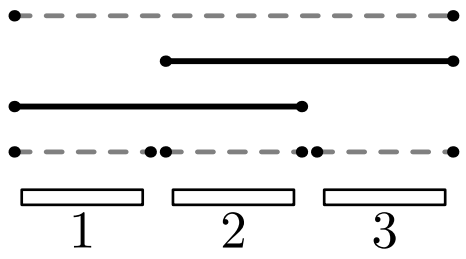
The following interaction will proceed in rounds. Your program should start each round by reading an integer k, denoting the number of currently visible cells ($0 \le k \le n$).

- If $k = n$, then the task is done and your program must exit.
- If $k < n$, you can flip the state of any barrier. On a separate line, print two integers i and j to flip the state of the (i, j) barrier ($1 \le i \le j \le n$). After your query, the next round begins, and your program should read a new value of k.

Your solution must make all cells visible using at most 2500 flips. In the beginning, not all cells are visible ($k < n$ in the first round).

The interactor is not adaptive: in every test, the state of all barriers is chosen before the program execution.

Example

standard input	standard output	Initial state
3 0 0 1 2 3	 2 2 2 3 1 2 2 2	

Note

In the example, initially, only two barriers, $(1, 2)$ and $(2, 3)$, are active. These two barriers cover all three cells, so k is equal to 0 in the first round.

- After flipping the $(2, 2)$ barrier, there are now three active barriers, and still $k = 0$ visible cells.
- After flipping the $(1, 2)$ barrier, cell 1 becomes visible, so now there is $k = 1$ visible cell.
- After flipping the $(2, 3)$ barrier, cell 3 also becomes visible. The only invisible cell now is 2, covered by the only active barrier, $(2, 2)$, and there are $k = 2$ visible cells.
- After flipping the $(2, 2)$ barrier, all barriers are now inactive, and all cells are visible. After reading $k = 3$, the program terminates.

Problem J. Jumping Frogs

Time limit: 2 seconds
Memory limit: 1024 megabytes

Julia is a fan of wild nature photos. Yesterday, she took two photos of a beautiful river with water lilies and some frogs sitting on them.

There are many water lilies on the river, numbered with consecutive positive integers from left to right, starting from 1. Both photos were taken from exactly the same spot, and both photos have the same n frogs sitting on water lilies. Each water lily can hold at most one frog.

After comparing the photos, Julia found out that all the frogs moved between the photos, since no water lily had a frog sitting on it in both photos. However, Julia couldn't understand which frog from the first photo moved to which water lily in the second photo, as all frogs looked exactly the same!

One thing is for sure: each frog jumped to a different water lily. Some frogs moved *to the left*, to a water lily with a smaller number, while the other frogs moved *to the right*, to a water lily with a larger number.

To investigate the movement of frogs, Julia wants to answer the following question: how many frogs moved to the left between the photos? As it may not be possible to find a unique answer to this question, you need to help Julia to find all possible answers.

Input

The first line contains a single integer n, denoting the number of frogs ($1 \le n \le 200\,000$).

The second line contains n integers $a_1, a_2, \ldots, a_n$, denoting the water lilies with frogs on them in the first photo, in increasing order ($1 \le a_1 < a_2 < \cdots < a_n \le 10^9$).

The third line contains n integers $b_1, b_2, \ldots, b_n$, denoting the water lilies with frogs on them in the second photo, in increasing order ($1 \le b_1 < b_2 < \cdots < b_n \le 10^9$).

All $2n$ given integers are distinct: no water lily has a frog sitting on it in both photos.

Output

In the first line, print a single integer k, denoting the number of possible answers to Julia's question.

In the second line, print k integers $c_1, c_2, \ldots, c_k$, denoting all possible answers in increasing order ($0 \le c_1 < c_2 < \cdots < c_k \le n$).

Examples

standard input	standard output
4 10 20 30 40 1 2 51 52	1 2
4 10 20 30 40 5 15 25 35	4 1 2 3 4
1 100 200	1 0

Note

In the first example, frogs that ended up on water lilies 1 and 2 must have moved to the left, while frogs that ended up on water lilies 51 and 52 must have moved to the right. Thus, we know for sure that exactly 2 frogs moved to the left between the photos.

Problem K. Kitchen Timer

Time limit: 2 seconds
Memory limit: 1024 megabytes

Kenny has a microwave in his kitchen. The microwave has a pretty weird one-button timer interface.

When you have put some food into the microwave and want it to start heating, you should press the button one or multiple times. When you press the button for the first time, the timer is set for 1 minute. If you immediately press the button again, 2 minutes are added to the timer, for a total of 3 minutes. If you immediately press the button yet again, 4 more minutes are added to the timer, and so on. If you press the button for the k-th time without a pause, it adds 2^k minutes to the timer.

It seems impossible to set the timer for some periods of time by using the button: for example, how to set the timer for 2 minutes? Fortunately, you can reset the button counter by making a pause for one second. So, for example, if you press the button, make a pause for one second, and then press the button again, the timer is set for 2 minutes. Another example: if you press, press, pause, press, press, press, the total time on the timer is $1 + 2 + 1 + 2 + 4 = 10$ minutes.

Kenny needs to heat his food for exactly x minutes. Help him to find the minimum number of one-second pauses he needs to set the timer. Let us assume that only pauses take time, while time to press the button is ignored.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The only line of each test case contains a single integer x, denoting the number of minutes Kenny needs to heat the food for ($1 \le x \le 10^{18}$).

Output

For each test case, print a single integer, denoting the minimum number of one-second pauses Kenny needs to make when setting the microwave timer for x minutes.

Example

standard input	standard output
7 1 2 3 4 10 239 123456789012	0 1 0 1 1 4 19

Note

In the first example test case, no pauses are needed: Kenny can just press the button once.

In the second example test case, Kenny can press, pause, press to set the timer for 2 minutes.

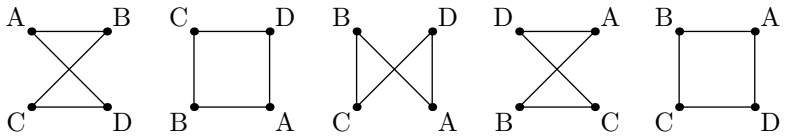
In the third example test case, Kenny can just press the button twice to set the timer for 3 minutes.

In the fourth example test case, Kenny can press, press, pause, press to set the timer for $1 + 2 + 1 = 4$ minutes.

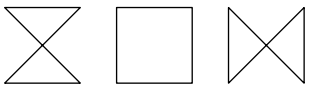
Problem L. Loops

Time limit: 2 seconds
Memory limit: 1024 megabytes

Consider four integers A, B, C, and D, such that $A < B < C < D$. Let's put them in the corners of a square in some order and draw a loop $A - B - C - D - A$. Depending on the arrangement of the integers, we can get different loop shapes, but some arrangements produce the same shape:

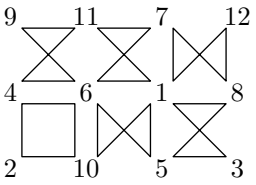


There are three possible loop shapes we can get:



type 1 type 2 type 3

Now consider an $n \times m$ matrix filled with distinct integers from 1 to nm, inclusive. Each 2×2 square in this matrix can be seen as a square with integers in its corners. Let's build a loop for each of these squares like we did before:



Your task is to perform the inverse operation. You are given the shape types for all $(n-1)(m-1)$ loops, and you need to build an $n \times m$ matrix filled with distinct integers from 1 to nm, inclusive, that produces these shapes.

Input

The first line contains two integers n and m ($2 \le n, m \le 500$).

Each of the next $n-1$ lines contains a string of $m-1$ characters without spaces. Each character is a digit from 1 to 3, denoting the type of the shape of the corresponding loop.

Output

Print an $n \times m$ matrix filled with distinct integers from 1 to nm, inclusive, that produces the shapes of the loops in the input.

It can be shown that such a matrix always exists. If there are multiple answers, print any of them.

Example

standard input	standard output
3 4 113 231	9 11 7 12 4 6 1 8 2 10 5 3

Problem M. Missing Vowels

Time limit: 2 seconds
Memory limit: 1024 megabytes

There are many ways to write a word on paper. For example, some writing systems, like Arabic and Hebrew, omit most vowels, although they write some of them.

In this problem, we will only consider strings consisting of English letters and hyphens. Letters '`a`', '`e`', '`i`', '`o`', '`u`', and '`y`' are considered to be vowels, while hyphens and all other letters are considered to be consonants. All comparisons are case-insensitive: uppercase and lowercase versions of the same letter are considered equal.

You are given two strings s and f, called the *short* name and the *full* name, respectively. Your task is to check whether the short name s can be obtained from the full name f by omitting some vowels (possibly none).

Input

The first line contains a single string s, denoting the short name.

The second line contains a single string f, denoting the full name.

Each string is non-empty and consists of at most 1000 English letters and hyphens.

Output

Print "`Same`" if the short name s can be obtained from the long name f by omitting some vowels, and "`Different`" otherwise.

Examples

standard input	standard output
Shrm-el-Shikh Sharm-el-Sheikh	Same
Eilot Eilat	Different
Saint-Petersburg Saint-Petersburg	Same
Bcdfghjklmnpqrstvwxz Abcdefghijklmnopqrstuvwxzyy	Same
Aa aaaA	Same
Etis-Atis-Amatis Etis-Atis-Animatis	Different
will-the-wisp will-o-the-wisp	Different
--a-very-short-name-- long-name	Different