Problem A. Piece of Cake

Source file name:	Pieceof.c, Pieceof.cpp, Pieceof.java, Pieceof.py
Input:	`Standard`
Output:	`Standard`

Alice received a cake for her birthday! Her cake can be described by a convex polygon with n vertices. No three vertices are collinear.

Alice will now choose exactly k random vertices ($k \geq 3$) from her cake and cut a piece, the shape of which is the convex polygon defined by those vertices. Compute the expected area of this piece of cake.

Input

Each test case will begin with a line with two space-separated integers n and k ($3 \leq k \leq n \leq 2500$), where n is the number of vertices of the cake, and k is the number of vertices of the piece that Alice cuts.

Each of the next n lines will contain two space-separated real numbers x and y ($-10.0 \leq x,\ y \leq 10.0$), where $(x,\ y)$ is a vertex of the cake. The vertices will be listed in clockwise order. No three vertices will be collinear. All real numbers have at most 6 digits after the decimal point.

Output

Output a single real number, which is the expected area of the piece of cake that Alice cuts out. Your answer will be accepted if it is within an absolute error of 10^{-6}.

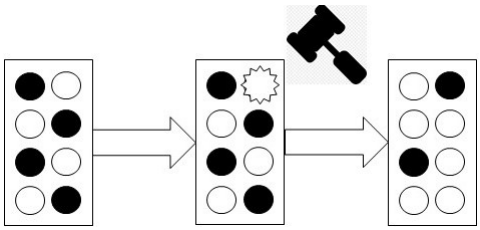
Example

Input	Output
4 3 0 0 1 1 2 1 1 0	0.50000000
5 5 0 4 4 2 4 1 3 -1 -2 4	12.50000000
5 3 -1.20 2.80 3.30 2.40 3.10 -0.80 2.00 -4.60 -4.40 -0.50	12.43300000

Problem B. Busy Board

Source file name:	Busy.c, Busy.cpp, Busy.java, Busy.py
Input:	`Standard`
Output:	`Standard`

Remember the busy boards for toddlers that have an array of holes into which to hammer pegs of various shapes? There's a new, electronic version. The board consists of a 2D grid of pegs. Each peg on the board can be either *up* or *down*, but not both simultaneously. You can pick any peg that is currently up, and "hammer" it down. This will push that peg down, and also raise all of the other pegs in its row, and its column, regardless of their current state. You cannot "hammer" a peg that is down (well, maybe you can, but it will have no effect). Those poor kids will never get all the pegs down at one time!



This example shows what happens when the top right peg is "hammered." ($\circ$ = up, $\bullet$ = down).

A substitute teacher wants to challenge her class. She uses the "Teacher Mode" to set up the board in a particular configuration, and then asks her students to see if they can get the board to a second configuration by hammering some (perhaps none) of the pegs.

That may be too tough of a problem for toddlers, but maybe you can handle it.

Input

Each test case will begin with a line with two space-separated integers r and c ($1 \leq r,\ c \leq 1000$), which are the dimensions of the board.

Each of the next r lines will have exactly c characters, consisting only of capital '`O`' (representing a peg that is up), capital '`X`' (representing a peg that is down), and no spaces or other characters. This is the starting configuration.

Following this, each of the next r lines will have exactly c characters, consisting only of capital '`O`' (representing a peg that is up), capital '`X`' (representing a peg that is down), and no spaces or other characters. This is the target configuration.

Output

Output a single integer, 1 if it is possible to reach the target configuration from the starting configuration, and 0 if it is impossible.

Example

Input	Output
4 2 XO OX XO OX OX OO XO OO	1
2 4 XXXX XXXX XOOO OOOO	0

Problem C. Cost of Living

Source file name:	Costof.c, Costof.cpp, Costof.java, Costof.py
Input:	`Standard`
Output:	`Standard`

A newspaper columnist recently wrote a column comparing Disneyland's price increases to other things in the economy; e.g., If gasoline were to have increased at the same rate as Disneyland's admission since 1990, it would cost $6.66 per gallon.

Consider the prices of a number of commodities over a number of consecutive years. There is an inflation rate ($inflation(x) > 0$) for year x to the next year ($x + 1$). Also, each commodity A has a modifier ($modifier(A) > 0$) that is fixed for that commodity. So, for commodity A in year $x + 1$:

$$price(A,\ x + 1) = price(A,\ x) \cdot inflation(x) \cdot modifier(A)$$

Unfortunately, the modifiers are unknown, and some of the prices and inflation rates are unknown.

Given some inflation rates, the prices for a number of commodities over a number of consecutive years, and some queries for prices for certain commodities in certain years, answer the queries.

Input

Each test case will begin with a line with three space-separated integers y ($1 \leq y \leq 10$), c ($1 \leq c \leq 100$), and q ($1 \leq q \leq y \cdot c$), where y is the number of consecutive years, c is the number of commodities, and q is the number of queries to answer.

Each of the next $y - 1$ lines will contain a single real number r ($1.0 \leq r \leq 1.5$, or $r = -1.0$), which are the inflation rates. A value of -1.0 indicates that the inflation rate is unknown. The first inflation rate is the change from year 1 to year 2, the second from year 2 to year 3, and so on. Known inflation rates will conform to the limits specified; unknown but uniquely determinable inflation rates may not, and are only guaranteed to be strictly greater than zero.

Each of the next y lines will describe one year's prices. They will contain c space-separated real numbers p ($1.0 < p < 1000000.0$, or $p = -1.0$), which indicate the price of that commodity in that year. A value of -1.0 indicates that the price is unknown.

Each of the next q lines will contain two space-separated integers a ($1 \leq a \leq c$) and b ($1 \leq b \leq y$), which represent a query for the price of commodity a in year b. All queries will be distinct.

All prices that can be uniquely determined will be strictly greater than zero and strictly less than 1000000.0. Values for prices and inflation rates in the input may not be exact, but will be accurate to 10 decimal places. All real values contain no more than 10 digits after the decimal point.

Output

Produce q lines of output. Each line should contain a single real number, which is the price of the given commodity in the given year, or -1.0 if it cannot be determined. Answer the queries in the order that they appear in the input. Your answers will be accepted if they are within an absolute error of 10^{-4}.

Example

Input	Output
4 2 2 1.3333333333 1.2500000000 -1 3.00 -1 4.00 8.00 5.00 10.00 -1 11.00 2 1 1 4	6.0000000000 5.5000000000

Problem D. It's a Mod, Mod, Mod, Mod World

Source file name:	Itsamod.c, Itsamod.cpp, Itsamod.java, Itsamod.py
Input:	`Standard`
Output:	`Standard`

You are given multiple problems with three integers p, q, and n. Find $\sum_{i=1}^{n}[(p \cdot i) \bmod q]$. That is, the first n multiples of p, modulo q, summed. Note that the overall sum has no modulus.

Input

Each input will begin with a line with a single integer W ($1 \leq W \leq 10^5$), which is the number of cases you must solve.

Each of the next W lines will contain three space-separated integers p, q and n ($1 \leq p,\ q,\ n \leq 10^6$), which are the parameters of the problem as described above.

Output

Output W lines, each with the answer for a given instance of the problem, in the order that they appear in the input.

Example

Input	Output
3 2 7 2 1 4 5 3 8 10	6 7 37

Problem E. Monotony

Source file name:	Monotony.c, Monotony.cpp, Monotony.java, Monotony.py
Input:	`Standard`
Output:	`Standard`

You are given an $r \times c$ grid. Each cell of this grid is filled with a number between 1 and $r \cdot c$ inclusive, and each cell's number is distinct.

Define a grid of numbers to be monotonic if each row and column is either increasing or decreasing (this can be different for each row or column).

Define a subgrid of the grid as follows: First choose some nonempty subset of the rows and columns. Next, take elements that lie in both the chosen rows and columns in the same order.

There are $(2^r - 1)\ (2^c - 1)$ nonempty subgrids of the given grid. Of these subgrids, count how many are monotonic.

Consider this grid:

$$\begin{matrix} 1 & 2 & 5 \\ 7 & 6 & 4 \\ 9 & 8 & 3 \end{matrix}$$

There are nine 1×1 subgrids, nine 1×2's, three 1×3's, nine 2×1's, nine 2×2's, three 2×3's, three 3×1's, three 3×2's, and one 3×3. They are all monotonic, for $9 + 9 + 3 + 9 + 9 + 3 + 3 + 3 + 1 = 49$ monotonic subgrids.

Input

Each test case will begin with a line with two space-separated integers r and c ($1 \leq r,\ c \leq 20$), which are the dimensions of the grid.

Each of the next r lines will contain c space-separated integers x ($1 \leq x \leq r \cdot c$, all x's are unique). This is the grid.

Output

Output a single integer, which is the number of monotonic subgrids in the given grid.

Example

Input	Output
3 3 1 2 5 7 6 4 9 8 3	49
4 3 8 2 5 12 9 6 3 1 10 11 7 4	64

Problem F. Heaps of Fun

Source file name:	Heapsof.c, Heapsof.cpp, Heapsof.java, Heapsof.py
Input:	`Standard`
Output:	`Standard`

Consider a rooted tree with n nodes, numbered $1 \dots n$. Each node will have a fixed integer b, and for each, a uniform random real number is chosen in the interval $[0 \dots b]$.

What is the probability that the random numbers chosen cause the tree to form a Heap (i.e., the random value in each node is less than the random values in its children)?

This probability can always be expressed as a rational number $\frac{P}{Q}$, with $Q \not\equiv 0 (\text{mod } 10^9 + 7)$. You are to output the probability as $P \cdot Q^{-1} \text{ mod } 10^9 + 7$, where Q^{-1} is an integer, which is the multiplicative inverse of Q modulo $10^9 + 7$ ($Q \cdot Q^{-1} \equiv 1 (\text{mod } 10^9 + 7)$). (Note: $P \cdot Q^{-1} \text{ mod } 10^9 + 7$ does not depend on whether P and Q are relatively prime, only on their ratio $\frac{P}{Q}$.)

Input

Each test case will begin with a line with a single integer n ($1 \leq n \leq 300$), which is the number of nodes in the tree.

Each of the next n lines will contain a pair of space-separated integers b ($1 \leq b \leq 10^9$) and p ($0 \leq p \leq n$) describing a node of the tree, where b is the fixed integer value in the node and p is the node number of its parent. The nodes are listed in order; node 1 is first, then node 2, and so on. A single node will have a parent $p = 0$. This is the root of the tree.

Output

Output a single integer, which is the probability expressed as $\left(P \cdot Q^{-1}\right) \text{ mod } (10^9 + 7)$.

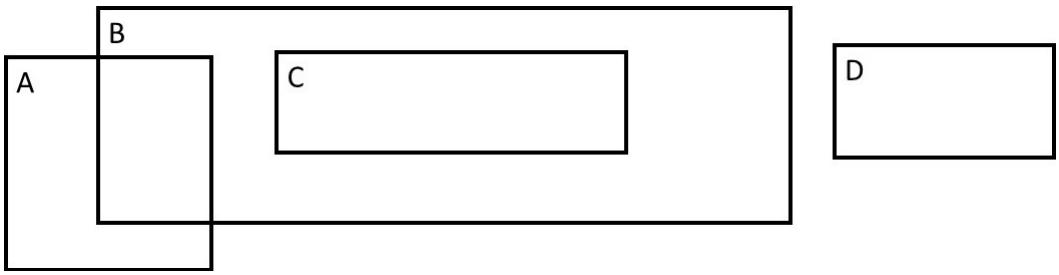
Example

Input	Output
2 1000000000 0 1000000000 1	500000004
5 2 3 2 3 1 0 2 3 2 3	87500001

Problem G. Intersecting Rectangles

Source file name:	Intersecting.c, Intersecting.cpp, Intersecting.java, Intersecting.py
Input:	`Standard`
Output:	`Standard`

You are given a set of n axis-aligned rectangles in a 2D plane. For this problem, two rectangles are considered to intersect if their boundaries contain any common points (in particular, two nesting rectangles don't count as intersecting). Determine if some pair of rectangles intersect.



In this example, only rectangles A and B intersect.

Input

Each test case will begin with a line with a single integer n ($1 \leq n \leq 10^5$), which is the number of rectangles.

Each of the next n lines will contain four space-separated integers:

$$x_1 \ y_1 \ x_2 \ y_2$$

($-10^9 \leq x_1,\ y_1,\ x_2,\ y_2 \leq 10^9,\ x_1 < x_2,\ y_1 < y_2$), which describe a rectangle, where $(x_1,\ y_1)$ is the lower left corner and $(x_2,\ y_2)$ is the upper right corner. All x values will be distinct. All y values will be distinct.

Output

Output a single integer, which is 1 if some pair of rectangles intersect, 0 if no pair of rectangles intersect.

Example

Input	Output
3 0 0 2 2 1 1 3 4 5 7 6 8	1
4 0 0 20 20 1 1 3 4 2 10 9 12 11 3 19 18	0

Problem H. Rocket Powered Hovercraft

Source file name:	Hovercraft.c, Hovercraft.cpp, Hovercraft.java, Hovercraft.py
Input:	`Standard`
Output:	`Standard`

You are programming an autonomous rocket powered hovercraft. The vehicle can travel very, very fast, but turning is difficult. Since it's hovering, it turns by firing directional thrusters, which will turn the vehicle whether it is moving forward or is stationary.

The salt flats on which you're testing have been mapped with a 2D Cartesian grid. The hovercraft starts at location $(0,\ 0)$ on this grid, facing in the positive X direction. Your job is to get the vehicle to another location $(x,\ y)$ on the flats.

The hovercraft has a fixed ground speed v miles per second and a fixed rate of rotation w radians per second. Due to the power of the rockets both for acceleration and deceleration, it can attain its maximum speed virtually instantaneously, and come to a stop from maximum speed just as quickly. Likewise, it can begin rotating at its fixed rate instantaneously, and stop just as quickly. It can rotate either clockwise or counter-clockwise.

You must figure out the minimum amount of time to get the vehicle to its target. The program which controls the vehicle can start forward motion, stop forward motion, start rotating, and stop rotating, each exactly once. Note that starting/stopping forward movement can be independent of starting/stopping rotation.

Input

Each test case will consist of exactly two lines.

The first line will contain two space-separated integers x and y ($-1000 \leq x,\ y \leq 1000,\ (x,\ y) \neq (0,\ 0)$), which indicate the location on the grid mapped onto the flats that you are trying to reach, in units of miles.

The second line of input will contain two space-separated real numbers with exactly two decimal places: v ($0.01 \leq v \leq 10.00$) and w ($0.01 \leq w \leq 10.00$), where v is the fixed speed of travel in miles per second, and w is the fixed rate of rotation in either direction in radians per second.

Output

Output a single real number, which is the minimum amount of time (in seconds) it would take get the hovercraft from $(0,\ 0)$ to $(x,\ y)$ subject to the constraints. Your answer will be accepted if it is within an absolute error of 10^{-3}.

Example

Input	Output
20 0 1.00 0.10	20.00000000
-10 10 10.00 1.00	3.14159265
0 20 1.00 0.10	28.26445910
-997 -3 5.64 2.15	177.76915187

Problem I. Cutting Strings

Source file name:	Cutting.c, Cutting.cpp, Cutting.java, Cutting.py
Input:	`Standard`
Output:	`Standard`

You are given a string s and an integer k. You can remove at most k non-intersecting substrings from s. Your task is to find the alphabetically (i.e., dictionary order) largest resulting string.

For example, with string `abcdcada` and $k = 2$, you can choose the substrings `[abc]d[ca]da` and remove them to get `dda`.

Input

Each input will begin with a line with a single integer c ($1 \leq c \leq 2 \cdot 10^5$), which is the number of cases you must solve.

Each of the next c lines will contain an integer k and a string s ($1 \leq k \leq |s| \leq 10^5,\ s \in [a-z]^*$), separated by a space.

The total length of all strings in the input will be at most 10^6.

Output

Output the largest string, alphabetically, that you can get by removing k or fewer non-intersecting substrings from s.

Example

Input	Output
4 2 abcdcada 1 ababb 2 ababb 1 dadbdcdbdad	dda bb bbb ddcdbdad

Problem J. Subsequences in Substrings

Source file name:	Subsequences.c, Subsequences.cpp, Subsequences.java, Subsequences.py
Input:	`Standard`
Output:	`Standard`

You are given two strings s, and t. Count the number of substrings of s that contain t as a subsequence at least once.

Note that a substring and a subsequence both consist of characters from the original string, in order. In a substring, the characters must be contiguous in the original string, but in a subsequence, they are not required to be contiguous. In the string `abcde`, `ace` is a subsequence but not a substring.

If s is `aa` and t is `a`, then the answer is 3: `[a]a`, `[aa]`, and `a[a]`.

Input

Each test case will consist of exactly two lines.

The first line will contain string s ($1 \leq |s| \leq 10^5,\ s \in [a-z]^*$), with no other characters.

The second line will contain string t ($1 \leq |t| \leq 100,\ |t| \leq |s|,\ t \in [a-z]^*$), with no other characters.

Output

Output a single integer, which is the number of substrings of s that contain t as a subsequence at least once.

Example

Input	Output
abcdefghijklmnopqrstuvwxyz a	26
abcdefghijklmnopqrstuvwxyz m	182
penpineappleapplepen ppap	68

Problem K. Knight of the Tarot Cards

Source file name:	Knightof.c, Knightof.cpp, Knightof.java, Knightof.py
Input:	`Standard`
Output:	`Standard`

Consider an infinite chessboard and a special knight whose move types can change given power-ups. The knight is trying to reach square $(0,\ 0)$.

Some of the squares of the infinite chessboard have tarot cards on them. If the knight lands on some position $(r,\ c)$ on the infinite chessboard with a tarot card on it, then the knight has the option of buying that card at that position. Each tarot card will have a price, and will have two integer values written on it. The tarot card with values a and b written on it allow the knight to make relative jumps:

$$(-a,\ -b) \quad (a,\ -b) \quad (-a,\ b) \quad (a,\ b) \quad (b,\ a) \quad (-b,\ a) \quad (b,\ -a) \quad (-b,\ -a)$$

The knight retains all cards he purchases and can make relative moves from any of the cards he owns any number of times. The knight must only pay when obtaining cards and can perform jumps at no additional cost. For example, if he buys a card with 3 and 2 and another card with 8 and 4, he may jump by $(-2,\ 3)$, and from there jump by $(8,\ 4)$, and later jump by $(-3,\ 2)$. Of course, he cannot make a jump $(a,\ b)$ until after he arrives at a square with a tarot card with a and b on it, and purchases that card.

Given positions of the tarot cards on the board and their prices, find the least amount that the knight must pay to reach square $(0,\ 0)$.

Input

Each test case will begin with a line containing a single integer n $(1 \leq n \leq 1000)$, which is the number of tarot cards on the chessboard.

Each of the next n lines contains a description of a tarot card in five space-separated integers:

$$r\ c\ a\ b\ p$$

$(-10^9 \leq r,\ c \leq 10^9,\ 1 \leq a,\ b,\ p \leq 10^9)$, where $(r,\ c)$ is the location of the tarot card, a and b are the offset values, and p is the price of the card.

The first tarot card is the initial position of the knight. Multiple tarot cards may exist at the same location. The knight must have a tarot card to move.

Output

Output a single integer, which is the minimum cost for the knight to reach the goal at $(0,\ 0)$. Output -1 if it is not possible to reach the goal.

Example

Input	Output
2 3 3 2 2 100 1 1 1 1 500	600
2 2 0 2 1 100 6 0 8 1 1	100
3 1 0 100 50 100 1 50 50 25 100 26 0 20 30 123	-1

Problem L. Planes, Trains, but not Automobiles

Source file name:	Planes.c, Planes.cpp, Planes.java, Planes.py
Input:	`Standard`
Output:	`Standard`

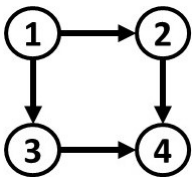
Being a traveling salesman is tough work. Per is one such salesman and would like to find an efficient way to visit all the cities in a foreign country exactly once.

Per defines efficiency in a peculiar way as Per hates flying on planes. Even worse, he absolutely refuses to use automobiles. Per's favorite mode of transportation is trains. He will gladly take trains for as long as possible.

The train system in this country is very peculiar, and very limited. Train lines are all one-way, and once anyone takes a train out of a city, there is no sequence of train lines that return to that city. This is because the country is trying to make money off of the more costly planes. In this country, every city has exactly one airport, so you can travel by plane from any city to any other city.

Per doesn't just want to know the minimum number of flights he needs. He also wants to know in which cities he can visit the airport during some trip with fewest flights. Per likes airport restaurants, you see, and would like to know which restaurants he can visit, so he can choose his route to visit his favorites. He can visit the airport if he flies in or out of the city. Note that Per can start in any city.

Consider this country with four cities, with the arrows representing one-way train routes:



There are several possible trips Per could take, but he's going to need to fly at least once. Here are some (but not all) possible routes with fewest flights, with $\rightarrow$ indicating a train trip and $\Rightarrow$ indicating a flight:

$$1 \rightarrow 2 \rightarrow 4 \Rightarrow 3$$

$$2 \rightarrow 4 \Rightarrow 1 \rightarrow 3$$

$$1 \rightarrow 3 \rightarrow 4 \Rightarrow 2$$

In this example, every airport is visited on at least one of the routes. Per has the option to choose his route so he can visit any airport restaurant he wishes.

Input

Each test case will begin with a line with two space-separated integers n ($1 \leq n \leq 10^5$) and m ($0 \leq m \leq 10^5$), where n is the number of cities and m is the number of train lines. The cities are numbered $1 \ldots n$.

Each of the next m lines contains two space separated integers a and b ($1 \leq a,\ b \leq n,\ a \neq b$), which indicates that there is a train line from city a to city b (but not back). All train lines will be distinct.

Output

Produce exactly two lines of output.

On the first line, output a single integer, which is the minimum number of flights Per must take to visit all of the cities.

On the second line, output a list of space-separated integers, which are the cities with airports he can visit. If he can visit an airport on any one of the routes with the minimum number of flights, it should be listed. Output these numbers in increasing order. If no airports are to be visited, output a blank line.

Example

Input	Output
4 4 1 2 1 3 2 4 3 4	1 1 2 3 4
4 3 1 2 2 3 3 4	0

Problem M. XOR Sequences

Source file name:	Sequences.c, Sequences.cpp, Sequences.java, Sequences.py
Input:	`Standard`
Output:	`Standard`

Suppose you are given two integers, m and n. You are also given a list of n distinct integers x_1, x_2, ..., x_n, with $0 \leq x_i \leq 2^m - 1$. For each number y from 0 to $2^m - 1$, you've found the number p_y such that x_{p_y} has a maximum bitwise-XOR with y. That is, $y \oplus x_{p_y} > y \oplus x_i$ for all $i = 1 \ldots n$, $i \neq p_y$ ($\oplus$ means bitwise-XOR).

Now, consider the reverse problem. Given m, n, and the sequence p_0, p_1, ..., p_{2^m-1}, count the number of sequences of distinct integers x_1, x_2, ..., x_n that could have generated that p sequence from the above algorithm. Two x sequences are different if there is some i such that x_i in one sequence is different from x_i in the other sequence. Output this count modulo $10^9 + 7$.

Input

Each test case will begin with a line with two space-separated integers m ($0 \leq m \leq 16$) and n ($1 \leq n \leq 2^m$), where 2^m is the length of the p sequence, and n is the length of the x sequences.

Each of the next 2^m lines will contain a single integer p ($1 \leq p \leq n$). These are the values of the sequence p_0, p_1, ..., p_{2^m-1}, in order. Every value from 1 to n inclusive will appear at least once.

Output

Output a single integer, which is the number of sequences x_1, x_2, ..., x_n which could have generated the sequence p_0, p_1, ..., p_{2^m-1} from the above algorithm, modulo $10^9 + 7$.

Example

Input	Output
3 6 1 1 2 2 3 4 5 6	4
2 3 1 2 1 3	0
3 8 1 2 3 4 5 6 7 8	1

Problem N. Nice Raises

Source file name:	Nice.c, Nice.cpp, Nice.java, Nice.py
Input:	`Standard`
Output:	`Standard`

Everyone knows that the UCF Programming Team coaches are the best trainers in the world and turn out some of the best problem solvers and programmers. The UCF team members are heavily sought after by different companies. FAAMGInc has arranged a nice raise package to attract the UCF team members. When it comes to giving raises, FAAMGInc gives each team member two options: a fixed raise or double their salary. Each team member can then decide which option they'd like. For example, if the fixed raise offered by FAAMGInc is `$5000`, then a team member earning `$3000` would choose the fixed raise option (`$5000`) but a team member making `$8000` would pick the double the salary option since that will be a better raise (`$8000` raise instead of `$5000` raise).

Given the fixed raise offered by FAAMGInc and the salary of each team member, determine which option is picked the most.

Input

The first input line contains two integers: n ($1 \leq n \leq 30$), indicating the number of team members and r ($1 \leq r \leq 50000$), indicating the fixed raise offered by FAAMGInc. Each of the next n input lines contains an integer (between 1 and 200000, inclusive) indicating the current salary for a team member.

Output

Output will be a single integer: 1, 2, or 0. Print 1 if the majority of the team members choose the fixed raise option, 2 if the majority pick the double the salary option, and 0 if the two options are selected by the same number of team members.

If a team member will end up with the same raise choosing either option, that team member doesn't count towards either group. For example, if the fixed raise is $2000 and a team member's current salary is $2000, the fixed raise and double the salary end up with the same raise for this team member so the team member doesn't count towards either group.

Example

Input	Output
6 1000 200 3000 500 2000 800 700	1
4 600 10 9000 20 8000	0
3 400 10000 9000 20	2
6 2500 100 100000 2500 100 2500 200000	0

Problem O. Changing Strings

Source file name:	Changingstr.c, Changingstr.cpp, Changingstr.java, Changingstr.py
Input:	`Standard`
Output:	`Standard`

We love our UCF and we are going to change everything to UCF!

Given a string, change the string to UCF as follows:

- Characters before the leftmost "`U`" in the string are changed to hyphen ("-").
- Characters after the rightmost "`F`" in the string are changed to hyphen.
- Characters between the leftmost `U` and the rightmost `F` are changed to "`C`".

Assume the string will contain at least one "`U`", at least one "`F`" after that U, and at least one character between the `U` and `F`.

Input

There is only one input line; it contains a string of uppercase letters. The string will have at least 3 and at most 50 characters.

Output

Print the input string converted to UCF.

Example

Input	Output
ABUDEGHFXYZ	--UCCCCF---
CCUZF	--UCF
ABFABCUABABFABUABFABUAB	------UCCCCCCCCCCF-----
UABCFABCDE	UCCCF-----

Problem P. Heavy Numbers

Source file name:	Heavy.c, Heavy.cpp, Heavy.java, Heavy.py
Input:	`Standard`
Output:	`Standard`

Consider a positive integer a. We define weight of a as:

$$(\text{number of digits in } a) \cdot (\text{sum of the digits in } a)$$

For example, if $a = 5767$, then weight of a is:

$$(4) \cdot (5 + 7 + 6 + 7) = 100$$

Given two positive integers, determine which one weighs more, i.e., it is heavier.

Input

There is only one input line; it contains two integers separated by exactly one space (blank). Assume each integer is between 1 and 10^6 (inclusive).

Output

Print 1 (one) if the first number is heavier, 2 (two) if the second number is heavier, and 0 (zero) if the two numbers weigh the same.

Example

Input	Output
59 1001	1
8 567	2
123 90	0