The 2nd Universal Cup

Stage 26: Latin America

March 23-24, 2024

This problem set should contain 12 problems on 20 numbered pages.

Based on

International Collegiate Programming Contest (ICPC)

Problem A. Almost Aligned

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6.4 seconds
Memory limit:	1024 megabytes

A meteor shower is about to happen! As the enthusiastic astronomy photographer that you are, you want to take a single picture of all the meteors that will be part of the phenomenon. Not only that, you want to take the best possible picture. You know that the smaller the area of the photo, the better the picture. But how small can you make the picture to capture them all?

You can take a picture of any rectangular region of your camera's view, but you cannot rotate the camera. That is, your photo can be any axis-aligned rectangle. The challenge? The meteors are constantly moving. Think of time (t) as the number of seconds that have passed since the start of the meteor shower. Your goal is to find a non-negative value of t at which you can capture every single meteor with the smallest possible rectangle. A photo captures all the meteors within the rectangle, including those on the border.

Input

The first line contains an integer N ($1 \le N \le 10^6$) indicating the number of meteors.

Each of the next N lines describes a meteor with four integers X, Y, V_X and V_Y ($-10^9 \le X, Y, V_X, V_Y \le 10^9$), representing the location and velocity of the meteor, as seen by your camera. This means that at any time $t \ge 0$ the coordinates of the meteor are $(X + t \cdot V_X, Y + t \cdot V_Y)$. If $t < 0$, the location of the meteor is undefined.

Output

Output a single line with the minimum area of an axis-aligned rectangle containing all the meteors at a time $t \ge 0$. The output must have an absolute or relative error of at most 10^{-9}.

Examples

standard input	standard output

4
0 0 10 10
0 0 10 10
10 10 -10 -10
10 0 -20 0
	22.222222222222222

3
0 -1 0 2
1 1 1 1
-1 1 -1 1
	0.000000000000000

3
0 -1 0 -2
1 1 1 1
-1 1 -1 1
	4.000000000000000

1 0 0 0 0	0.000000000000000

Problem B. Beating the Record

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Speedy Greedy is a professional speedrunner: they play a videogame repeatedly, with the aim of finishing it as quickly as possible.

The game that Speedy Greedy is currently tackling consists of N levels, numbered sequentially from 1 to N. Although these levels must be completed in order, each of them is independent of the others in terms of gameplay. That is, no event from one level can have a relevant effect on the next levels (unlike other games where items, spells, points, lives, etc. are carried over from one level to the next).

The current world record for the fastest completion time of the game is T seconds. Speedy Greedy is determined to beat this world record, with no concern for the margin. Whether the game is finished 1 second or 1000 seconds faster than the current record is irrelevant: what matters to Speedy Greedy is improving the world record.

Speedrunners often choose and adjust their actions dynamically during a run, depending on various factors such as the situation of their character in the game. It is also common to restart the game, as for example once T seconds have passed there is no more hope to beat the world record. Any run of the game can be restarted at any point. When a restart command is issued by the speedrunner, the game instantly restarts from the beginning. Thus, to beat the world record, Speedy Greedy must complete the N levels sequentially, in a single run that takes less than T seconds.

How can Speedy Greedy achieve that? Well, most sections of the game are completely safe and easy to play for a speedrunner of such skill. Those sections take fixed amounts of time and have no risk of failing. However, there is a very hard section in each of the N levels, where the result of playing that section is not completely under Speedy Greedy's control, and also depends on the chosen strategy.

In each level, there are two possible strategies, and when reaching the hard section, Speedy Greedy can choose exactly one of them to attempt. Each strategy has its own probability of success, and the actual time that the level takes to complete depends on the chosen strategy and whether the attempt is successful or not.

For the purposes of this problem, we assume that Speedy Greedy can instantly detect whether the chosen strategy for a level succeeded or failed, right at the moment that the hard section of the level is reached. That is, reaching the hard section of a level, choosing a strategy, and knowing whether the attempted strategy succeeded or failed are all simultaneous events, occurring at exactly the same time.

Grinding the game by playing again and again and again becomes tiring and physically exhausting at such a high level of speedrunning competition. Speedy Greedy then decided to play the game so as to minimize the expected total play time until beating the world record. Your task is to compute this minimum.

Note that the total play time not only includes the time of the final successful run that takes less than T seconds (beating the world record), but also includes the time spent during all previous failed runs.

Input

The first line contains three integers N ($1 \le N \le 4$), T ($1 \le T \le 5000$) and S ($1 \le S \le 1000$), indicating respectively the number of levels, the time of the current world record, and the time from the start of level 1 until the instant that the hard section in level 1 is reached.

The i-th of the next N lines describes the two strategies for level i by means of six integers P_1, G_1, B_1, P_2, G_2 and B_2 ($1 \le P_j \le 99$ and $0 \le G_j \le B_j \le 1000$, for $j = 1, 2$).

P_j indicates the probability (as a percentage) that strategy j succeeds when used.

G_j represents the "good time" for strategy j, that is, the time from the moment the strategy succeeds, until the moment that the hard section of the next level is reached. For the last level, G_j represents the time from the moment the strategy succeeds, until the completion of the game.

Finally, B_j denotes the "bad time" for strategy j. It is analogous to G_j, but indicates the time corresponding to the case when the strategy fails.

All input times are given in seconds. It is guaranteed that the input is such that beating the world record is possible.

Output

Output a single line with the minimum expected total play time until Speedy Greedy beats the world record. The output must have an absolute or relative error of at most 10^{-9}.

Examples

standard input	standard output
1 100 50 50 48 49 1 1 50	98.500000000000000
1 100 50 50 48 49 52 1 50	97.153846153846154

Problem C. Clever Cell Choices

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	1024 megabytes

Two players play the following game on an $N \times M$ grid:

- Initially, each cell of the grid is either empty or occupied.
- Players take turns placing a stone on an empty cell, occupying the cell. Each new stone must be adjacent to the last placed stone, with the exception of the starting stone that can be placed on any empty cell. A stone is adjacent to another stone if they are located in two cells that share a side.
- The game ends whenever a player cannot place a stone according to the above rules. In that case, the player who cannot place a stone loses the game, and the other player wins.

A winning starting cell is a cell such that the first player wins the game if they place their starting stone there, assuming both players play optimally. Given a description of the initial grid, you must tell how many winning starting cells it has.

Input

The first line contains two integers N and M ($1 \leq N, M \leq 50$) indicating the dimensions of the grid.

Each of the next N lines contains a string of length M. In the i-th string, the j-th character describes the initial state of cell (i, j). The character is either "`.`" (dot) denoting an empty cell, or "`#`" (hash) representing an occupied cell.

Output

Output a single line with an integer indicating the number of winning starting cells.

Examples

standard input	standard output

`3 3`
`#.#`
`...`
`#.#`
	`4`

`3 3`
`..#`
`...`
`...`
	`0`

`1 4`
`...#`
	`2`

Problem D. DiviDuelo

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.3 seconds
Memory limit:	1024 megabytes

The Intense Challenges Players Club (ICPC) is hosting a DiviDuelo tournament.

DiviDuelo is a new two-player, turn-based game. In DiviDuelo, a number N is selected and the list of its divisors is written. For example, if $N = 10$ is selected, the list of numbers $1, 2, 5, 10$ is written. Players alternate turns picking one still unpicked divisor from the list each turn, until all divisors have been picked.

The winner is determined by the greatest common divisor (GCD) of the numbers picked by the starting player. If the GCD is not equal to 1, the starting player wins. Otherwise, if the GCD is equal to 1, the other player wins.

The ICPC needs your help to prepare some statistics about the games played in the tournament. Given the value of N, determine if the starting player can win the game assuming both players play optimally.

Input

The input consists of a single line that contains an integer N ($1 \le N \le 10^{12}$) indicating the number selected for the game.

Output

Output a single line with the uppercase letter "`Y`" if the starting player can win the game and the uppercase letter "`N`" otherwise, assuming both players play optimally.

Examples

standard input	standard output
10	Y
9	N
1	N

Problem E. Expanding STACKS!

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.3 seconds
Memory limit:	1024 megabytes

Tired of always waiting in lines, you invented a revolutionary restaurant concept: "STACKS! Where the last customer is served first".

The restaurant operates as follows:

- There is a single line inside the restaurant.
- When a customer enters, they immediately join the back of the line.
- Whenever a stack of glazed pancakes (the only dish at STACKS!) is ready, it's served to the person at the back of the line, who then immediately devours the pancakes and leaves the restaurant.

This business model has been incredibly successful, so much so that STACKS! is beginning to expand.

In fact, you just opened the very first STACKS!+, offering two types of pancakes: glazed and savory. The new restaurant works as follows:

- There are two lines, one for each type of pancake. Each customer joins the back of the line corresponding to the type of pancake they want.
- Whenever a stack of glazed pancakes is ready, it is served to the customer at the back of the glazed pancake line, who immediately devours it and leaves the restaurant.
- Whenever a stack of savory pancakes is ready, it is served to the customer at the back of the savory pancake line, who instantly gobbles it and leaves the restaurant.

As the boss, you want to ensure your employees follow the concept and maintain your vision. Given the order in which customers come in and out of the restaurant, you need to determine whether there is an assignment of customers to lines such that the STACKS!+ concept is followed.

You can assume that whenever a customer enters the restaurant, they immediately join the back of a line, and that they leave as soon as they are served. Also, each customer visits the restaurant exactly once.

Input

The first line contains an integer N ($1 \le N \le 1000$) indicating the number of customers who visited STACKS!+. Each customer is identified by a distinct integer from 1 to N.

The second line contains $2N$ signed integers $X_1, X_2, \ldots, X_{2N}$ ($1 \le |X_i| \le N$ for $i = 1, 2, \ldots, 2N$) indicating, in chronological order, the entrance and departure of the customers. The value $X_i = +c$ denotes the entrance of customer c into the restaurant, while $X_i = -c$ represents their departure. It is guaranteed that each customer enters and leaves the restaurant exactly once, and that they do not leave before entering.

Output

Output a single line with a string of length N if there is an assignment of customers to lines such that the STACKS!+ concept can be honored. In this case the i-th character of the string must be the uppercase letter "`G`" if customer i is assigned to the glazed pancake line, and the uppercase letter "`S`" if they are assigned to the savory line. If there are multiple solutions, output any of them.

If the STACKS!+ concept cannot be honored with the given input, output the character "`*`" (asterisk) instead.

Examples

standard input	standard output
2 +2 +1 -1 -2	GG
2 +1 +2 -1 -2	GS
3 +1 +2 +3 -1 -2 -3	*

Problem F. Fair Distribution

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.35 seconds
Memory limit:	1024 megabytes

An entrepreneur has N blueprints, each describing a type of building. Each blueprint specifies the building's height through two integers G and R.

- G: Height of the ground floor. It can be zero, indicating that the building has no ground floor.
- R: Height of each residential floor. Each building has at least one residential floor.

The entrepreneur wants to distribute all of these blueprints among their two children Alice and Bob. Each child will build exactly one building from each blueprint allocated to them, choosing the number of residential floors for each building.

The entrepreneur wants to avoid showing favoritism towards either child, so they are looking for a fair distribution of the blueprints. They decided that a fair distribution is one in which it is possible to construct the buildings in such a way that the sum of the heights of the buildings constructed by each child is the same. Can you tell whether a fair distribution exists?

Consider the following example for $N = 3$ blueprints:

1. $G = 1$ and $R = 1$ (possible heights are $2, 3, 4, \ldots$);
2. $G = 0$ and $R = 3$ (no ground floor, possible heights are $3, 6, 9, \ldots$);
3. $G = 2$ and $R = 1$ (possible heights are $3, 4, 5, \ldots$).

In this case a possible fair distribution is assigning the second blueprint to Alice and the rest to Bob. Even though Alice receives a single blueprint while Bob receives two, they may construct two residential floors on the first building type (height 3), two residential floors on the second (height 6), and one residential floor on the third (height 3). In this way, the sum of the heights of the buildings constructed by each child would be 6.

Input

The first line contains an integer N ($1 \le N \le 2 \cdot 10^5$) indicating the number of blueprints.

Each of the next N lines contains two integers G ($0 \le G \le 2 \cdot 10^5$) and R ($1 \le R \le 10^9$) denoting respectively the height of the ground floor and the height of each residential floor specified by the corresponding blueprint. The sum of the heights of the ground floors of all blueprints is at most $2 \cdot 10^5$.

Output

Output a single line with the uppercase letter "`Y`" if there exists a fair distribution of the blueprints, and the uppercase letter "`N`" otherwise.

Examples

standard input	standard output

3
1 1
0 3
2 1
	Y

3
3 2
2 1
3 2
	Y

3
1 10
2 20
4 30
	N

1 1 1	N

Problem G. Greek Casino

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.7 seconds
Memory limit:	1024 megabytes

Since the early civilizations, humankind has enjoyed games of chance. Even the ingenious Greeks, known for their groundbreaking concept of the least common multiple (LCM), couldn't resist a good gamble.

Inspired by this mathematical marvel, folks in Athens devised a unique betting system: after purchasing a ticket, a participant would receive a random number of coins. To determine this number, there are $N \geq 3$ ordered slots numbered from 1 to N. A token is initially placed at slot 1, and the following steps are repeated:

- Let x be the number of the slot where the token is currently located.
- Generate a random integer y between 1 and N, and compute z, the LCM of x and y.
- If $z > N$, the procedure ends.
- Otherwise, the token is moved to slot z, and the participant receives one coin.

As it is well known, the house always wins: the casino employs a particular probability distribution for generating random integers, so as to ensure a profitable outcome.

The casino owner is constantly seeking to optimize the betting system's profitability. You, an AI designed to aid in such tasks, are given N and the probability distribution. Determine the expected total number of coins awarded to a participant.

Input

The first line contains an integer N ($3 \leq N \leq 10^5$) indicating the number of slots.

The second line contains N integers $W_1, W_2, \ldots, W_N$ ($1 \leq W_i \leq 1000$ for $i = 1, 2, \ldots, N$), representing that the probability of generating i is $W_i / \left(\sum_j W_j\right)$, that is, the probability of generating i is the relative weight of W_i with respect to the sum of the whole list $W_1, W_2, \ldots, W_N$.

Output

Output a single line with the expected total number of coins awarded to a participant. The output must have an absolute or relative error of at most 10^{-9}. It can be proven that the procedure described in the statement ends within a finite number of iterations with probability 1, and that the expected total number of coins is indeed finite.

Examples

standard input	standard output
3 1 1 1	3.5000000000
3 1 1 2	3.6666666667

Problem H. Harmonic Operations

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.4 seconds
Memory limit:	1024 megabytes

In this problem, we consider three types of operations that can be applied to any 0-based string t of length $|t| \geq 2$.

- $I(t)$: Reverses t.
- $R(t, D)$: Rotates t to the right D positions, for some positive integer $D < |t|$. That is, for each $0 \leq i < |t|$, the character at position $(i + D) \bmod |t|$ in $R(t, D)$ is the character at position i in t.
- $L(t, D)$: Analogous to $R(t, D)$, but rotates t to the left instead of to the right.

For example, $I(\text{“}\texttt{pda}\text{''}) = \text{“}\texttt{adp}\text{''}$, $R(\text{“}\texttt{pda}\text{''}, 2) = \text{“}\texttt{dap}\text{''}$, and $L(\text{“}\texttt{pda}\text{''}, 2) = \text{“}\texttt{apd}\text{''}$. Note that for any t and any D, it holds that $|I(t)| = |R(t, D)| = |L(t, D)| = |t|$.

When a list of the above operations is applied to a string, it is done sequentially in list order. That is, the first operation of the list is applied to the original string, the second operation is applied to the result after having applied the first operation, the third operation is applied to the result after having applied the first two operations, and so on.

You are given a string S consisting of lowercase letters, and a list of K operations $F_1, F_2, \ldots, F_K$. Your task is to find out how many pairs of indices (i, j) there are such that $1 \leq i \leq j \leq K$, and applying the sublist of operations $F_i, F_{i+1}, \ldots, F_j$ to S yields S as the final result.

Consider, for instance, $S = \text{“}\texttt{pda}\text{''}$, $K = 2$, $F_1 = R(t, 2)$ and $F_2 = L(t, 2)$. The result of applying the sublist F_1 to S is $R(\text{“}\texttt{pda}\text{''}, 2) = \text{“}\texttt{dap}\text{''}$, which is different from S. The result of applying the sublist F_1, F_2 to S is $L(R(\text{“}\texttt{pda}\text{''}, 2), 2) = L(\text{“}\texttt{dap}\text{''}, 2) = \text{“}\texttt{pda}\text{''} = S$. Finally, the result of applying the sublist F_2 to S is $L(\text{“}\texttt{pda}\text{''}, 2) = \text{“}\texttt{apd}\text{''}$, which is different from S. Thus, in this example, the answer is 1.

Input

The first line contains a string S ($2 \leq |S| \leq 2 \cdot 10^5$) which is made up of lowercase letters.

The second line contains an integer K ($1 \leq K \leq 2 \cdot 10^5$) indicating the number of operations in the list of operations that is being considered.

Operations are described in the next K lines, in the order they appear in the list, one operation per line. If the operation is $I(t)$, the line contains the uppercase letter “`I`”. If the operation is $R(t, D)$, the line contains the uppercase letter “`R`” and the integer D ($1 \leq D < |S|$). Finally, if the operation is $L(t, D)$, the line contains the uppercase letter “`L`” and the integer D ($1 \leq D < |S|$).

Output

Output a single line with an integer indicating the requested number of pairs.

Examples

standard input	standard output
pda 2 R 2 L 2	1
aaa 4 R 1 I I R 1	10
caso 6 L 1 I I R 1 I I	4

Problem I. Insects, Mathematics, Accuracy, and Efficiency

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	1024 megabytes

Bog has three passions in life: insects, mathematics, accuracy, and efficiency. His last passion led him to wanting to merge the first two, so Bog decided to adopt a differential grasshopper, an adorable little green guy that he named Dydx.

In order to keep Dydx happy, Bog made him a little lair. He bought a sprinkler that could water any plant within a circle with radius R, and planted N crops within this circle.

Dydx really liked his new home! But Bog realized that the grasshopper would only stay within the area defined by the smallest convex polygon that encloses all crops. Now he regrets not having spread the crops out more. Fortunately, Bog managed to find one last crop he hadn't planted yet. Bog wants your help to maximize the area Dydx can inhabit by planting his last crop within the sprinkler's range.

Input

The first line contains two integers N and R ($1 \le N, R \le 10^4$), indicating respectively the number of planted crops and the radius of the circle defined by the sprinkler. Crops are represented as points in the two-dimensional plane, where $(0, 0)$ are the coordinates of the sprinkler.

Each of the next N lines describes a crop with two integers X and Y, indicating that the crop is planted at coordinates (X, Y). The Euclidean distance from (X, Y) to $(0, 0)$ is at most R. No two crops have the same location.

Output

Output a single line with the maximum area of the region Dydx can inhabit after planting one more crop within the range of the sprinkler. The output must have an absolute or relative error of at most 10^{-9}. Notice that the added crop does not need to have integer coordinates.

Examples

standard input	standard output

`4 1000`
`-1000 0`
`0 0`
`1000 0`
`0 -1000`
	`2000000.000000000000000`

`2 100`
`17 7`
`19 90`
	`4849.704644437563740`

`1 100`
`13 37`
	`0.0`

Problem J. Joys of Trading

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	1024 megabytes

Apolyanka and B ü delsdorf are two small Neolithic villages that have recently come into contact. There are N resources, numbered from 1 to N, and each village is capable of independently producing any of them, albeit with different efficiencies. In order to produce one unit of resource i, Apolyanka needs A_i person-hours, while B ü delsdorf needs B_i person-hours. Currently, Apolyanka is producing U_i units of resource i in each given time period, while B ü delsdorf is producing W_i units.

Each village is currently working at maximum capacity, that is, there is no way they can put more person-hours to work than they are employing now. However, through the recently discovered benefits of trade, it is possible for both villages to produce all the resources they need while reducing the total person-hours worked, and thus becoming able to spend those freed person-hours resting and playing some games. All that is needed is that the villages cooperate, coordinate work, and exchange resources among them.

For example, suppose $N = 2$, resource 1 is wood, resource 2 is food, $A_1 = 1$, $U_1 = 2$, $B_1 = 4$, $W_1 = 1$, $A_2 = 2$, $U_2 = 1$, $B_2 = 3$, and $W_2 = 4$. Then Apolyanka is doing 4 person-hours of work: $A_1 \cdot U_1 =2$ for producing $U_1 =2$ units of wood, and $A_2 \cdot U_2 =2$ for producing $U_2 =1$ unit of food. Similarly, B ü delsdorf is doing 16 person-hours of work: $B_1 \cdot W_1 =4$ for producing $W_1 =1$ unit of wood, and $B_2 \cdot W_2 =12$ for producing $W_2 =4$ units of food. Thus, the total production is $U_1 + W_1 =3$ units of wood and $U_2 + W_2 =5$ units of food, requiring $4 + 16 = 20$ person-hours.

However, a better organization is possible: Apolyanka could produce 3 units of wood and 0.5 units of food, while B ü delsdorf could produce no wood and 4.5 units of food. The total production of each resource would be the same, but requiring only $3A_1 + 0.5A_2 + 0B_1 + 4.5B_2 =3 + 1 + 13.5 = 17.5$ person-hours.

Another example with $N = 3$ is $A_1 = 1$, $B_1 = 2$, $A_2 = 2$, $B_2 = 1$, $A_3 = 1$, $B_3 = 1$, and $U_i = W_i = 1$ for $i = 1, 2, 3$. In this case, each village is currently working 4 person-hours. With a slight reorganization, however, they can each work 3 person-hours while producing the exact same total resources! All that is required is for Apolyanka to produce one less unit of resource 2 and one more of resource 1, while B ü delsdorf does the opposite.

Given all of these values, can you compute what is the minimum total number of person-hours that the villages have to work, in order to produce exactly the same total resources? Note that the number of person-hours invested in producing a resource is not required to be an integer.

Input

The first line contains an integer N ($1 \le N \le 10^5$) indicating the number of resources. Each resource is identified by a distinct integer from 1 to N.

The i-th of the next N lines describes resource i with four integers A_i, U_i, B_i, and W_i ($1 \le A_i, U_i, B_i, W_i \le 1000$ for $I = 1, 2, \ldots, N$), as explained in the statement.

Output

Output a single line with the minimum total number of person-hours required to produce the resources. The output must have an absolute or relative error of at most 10^{-9}.

Examples

standard input	standard output

2
1 2 4 1
2 1 3 4
	17.500000000000000

3
1 1 2 1
2 1 1 1
1 1 1 1
	6.000000000000000

Problem K. KMOP

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.5 seconds
Memory limit:	1024 megabytes

You probably know the KMP algorithm. You may also know that "`KMP`" is an acronym that stands for "`Knuth Morris Pratt`", who jointly published the algorithm in 1977. How do you pronounce "`KMP`"? Of course, you can just say "`Knuth Morris Pratt`", but what about pronouncing the acronym itself? Since "`KMP`" is not a pronounceable word, you are forced to say the letters one by one. In this problem we are interested in pronounceable acronyms.

We need a few definitions to formalize the requirement. A phrase is a list of words and a word is a sequence of letters. Each letter is either a vowel or a consonant. Deciding whether a letter is a vowel or a consonant depends on the language and other elements. For simplicity, we say that the six letters "`A`", "`E`", "`I`", "`O`", "`U`" and "`Y`" are vowels, while all the rest are consonants. Although it is debatable whether a given word is pronounceable, we say that a word is pronounceable when it does not contain more than two contiguous consonants. For instance, "`LEMPEL`" is a pronounceable word, while "`DIJKSTRA`" is not.

Given a phrase composed of N words, an acronym for the phrase is the concatenation of N prefixes, one prefix for each word, in the order they appear in the phrase. Each prefix must have at least one and at most three letters. Your task is to determine the minimum length a pronounceable acronym can have.

As an example with $N = 3$ consider the phrase "`KNUTH MORRIS PRATT`". There are 27 possible acronyms for this phrase, such as "`KMP`", "`KMPR`", "`KMPRA`", "`KMOP`", "`KMOPR`" and "`KNUMORPRA`", among others. Some of these acronyms are pronounceable ("`KMOP`" and "`KMOPR`"), while some others not ("`KMP`", "`KMPR`", "`KMPRA`" and "`KNUMORPRA`"). Since the only three-letter acronym "`KMP`" is not pronounceable, it follows that "`KMOP`" is a minimum-length pronounceable acronym for the phrase.

Input

The first line contains a positive integer N indicating the number of words in the phrase.

Each of the next N lines contains a non-empty string made of uppercase letters representing a word in the phrase. Words are given in the order they appear in the phrase. The sum of the lengths of all the strings is at most 10^6.

Output

Output a single line with an integer indicating the minimum length a pronounceable acronym can have, or the character "`*`" (asterisk) if no pronounceable acronym exists for the phrase.

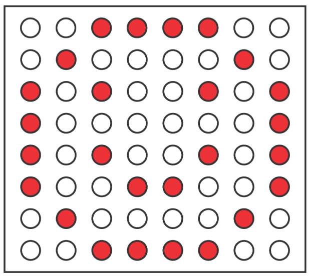
Examples

standard input	standard output
3 KNUTH MORRIS PRATT	4
3 KNUTH M PRATT	5
3 K M P	*
2 K M	2
4 YOU SHOULD BE DANCING	5

Problem L. LED Matrix

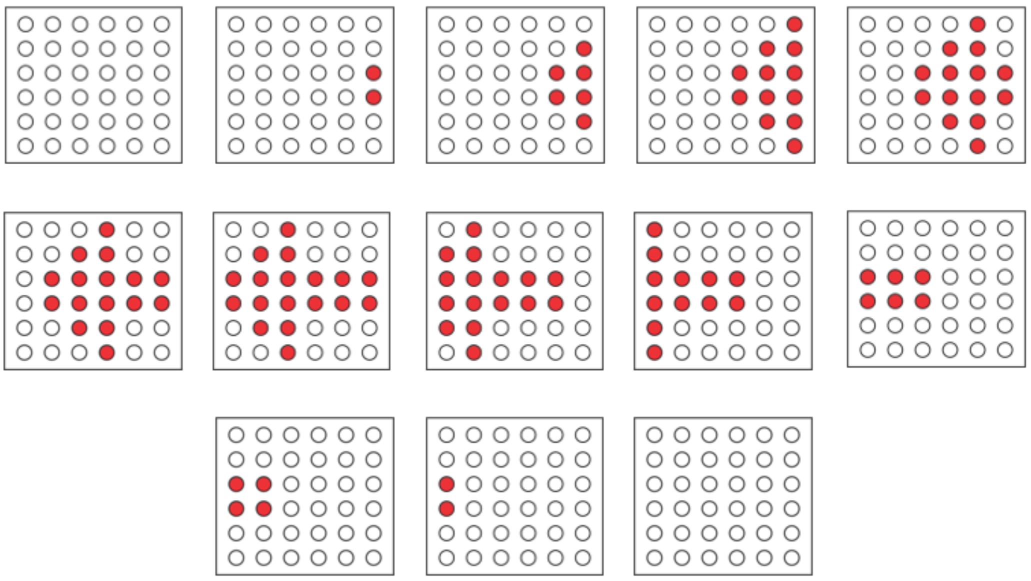
Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.3 seconds
Memory limit:	1024 megabytes

An LED matrix is a two-dimensional array of LEDs that is used to display information. This is achieved by turning on the LEDs that form a desired pattern. The figure below represents an LED matrix displaying a smiling-face pattern. LEDs that are turned off are shown in white, while LEDs that are turned on appear colored.



Some LED matrices scroll the pattern from right to left across the matrix, turning on just the appropriate LEDs at each step. Thus, any pattern with the same height as the matrix can be displayed, even patterns that are wider than the matrix. The pattern scrolling works as follows: Initially, all the LEDs in the matrix are turned off. At the next step, the last column of the matrix displays the first column of the pattern. At each new step, the pattern is moved one column to the left across the matrix, until the first column of the matrix displays the last column of the pattern. Finally, all the LEDs in the matrix are turned off again. If an LED matrix is equipped with pattern scrolling, the scrolling occurs even if the pattern is not wider than the matrix.

The picture below shows all the steps required to display a pattern of an arrow that is pointing to the left.



Astrid has just received an old LED matrix with pattern scrolling, and she thinks that some LEDs might be broken. Since broken LEDs cannot be turned on, she is worried that some patterns will not display properly. Given the description of the state of each LED, and the pattern to display, you must tell whether the appropriate LEDs can be turned on at every step of the pattern scrolling.

Input

The first line contains three integers R, C, and K ($1 \leq R, C, K \leq 1000$), indicating respectively the number of rows of both the LED matrix and the pattern, the number of columns of the matrix, and the number of columns of the pattern.

The next R lines describe the matrix and the pattern from top to bottom. Each of these lines contains a string M of length C and a string P of length K, describing respectively a row of the matrix and a row of the pattern. Each character of both M and P is either "`*`" (asterisk) or "`-`" (hyphen). For M, the character "`*`" indicates a good LED while the character "`-`" represents a broken LED. For P, the character "`*`" indicates an LED that must be turned on while the character "`-`" represents an LED that must be turned off.

Output

Output a single line with the uppercase letter "`Y`" if the appropriate LEDs can be turned on at every step of the pattern scrolling, and the uppercase letter "`N`" otherwise.

Examples

standard input	standard output
`6 6 6`	`N`
`****** --*---`	
`****** -**---`	
`****** ******`	
`****** ******`	
`****** -**---`	
`*****- --*---`	
`2 4 6`	`N`
`**** ------`	
`***- *-----`	
`2 6 4`	`Y`
`****** ****`	
`*-**-* ----`	
`1 1 1`	`Y`
`* *`	
`1 1 1`	`Y`
`* -`	
`1 1 1`	`N`
`- *`	
`1 1 1`	`Y`
`- -`	