Problem A. Elections

Input file:	`elections.in`	Output file:	`elections.out`
Time limit:	10 seconds		

New parliament election in Berland is coming soon. Each of N political parties wants to be elected into the parliament. There is a law in Berland that allows only parties with more than a certain amount of votes be elected. Thus, some of the smaller parties are trying to use different technologies to collect necessary amount of votes.

The first technology is completely legal. Two or more parties can go to the elections together, forming so-called *election block*.

The second technology is not so legal. Some parties have specific information about their opponents. Those opponents don't want this information to become public.

If several parties join into one election block, their information is joined together as well. Thus, they become more powerful. However, the opponents of each party of the block know something discreditable about the whole block. The party or block might have some 'black' information on itself.

Let us now enumerate parties with the integers from 1 to N. Parties and blocks are allowed to join together into new blocks. Those blocks are numbered with the consecutive integers ($N+1$, $N+2$, etc.).

You are to write a program that processes queries of two different kinds.

- Query '`1 a b`' means that you have to answer the question: is there any discreditable information that party or block a knows about party or block b?
- Query '`2 a b`' means that you need to join party or block a with party or block b. The new block will have all the information a has, and all the information b has. All the information that was known by some parties or blocks about a and b now concerns the newly created block.

Input

The first line of the input file contains integers N and M ($1 \le N \le 100\,000$; $1 \le M \le 200\,000$). Next M lines contain pairs of integers a and b denoting that party a knows something discreditable about party b ($1 \le a, b \le N$). The next line contains single integer number Q ($1 \le Q \le 200\,000$). Each of the next Q lines contains a query. A query of the first kind looks like '`1 a b`'. A query of the second kind looks like '`2 a b`'. You should process queries in the order they are given. Each pair a, b references only existing parties or blocks. It is guaranteed that numbers a and b are different in any '`2 a b`' query, but they can be equal in a '`1 a b`' query.

Output

Write on the separate lines of the output file answer `YES` or `NO` for each query of the first kind.

Example

elections.in	elections.out

4 6
1 2
1 3
3 2
4 4
2 4
1 2
4
1 3 4
2 2 3
1 5 4
1 4 5
	NO
	YES
	NO

Problem B. Keven

Input file:	`keven.in`	Output file:	`keven.out`
Time limit:	2 seconds		

Consider a string of even length and integer K. The string is called K-*even* if and only if the first half of the string differs from the second half in no more than K positions.

For example, string `abac` is 1-even, 2-even, but not 0-even.

You are given integer K and the cyclic string with the odd length. You are to find its K-even substring of the maximal length. Note, input string is cyclic, so you can use any of its cyclic shifts.

Input

The first line of the input file contains integer K ($0 \le K \le 2000$). The second line contains string of small Latin letters. The length of the string is odd and it is less than 2000.

Output

Print single line containing K-even substring of the maximal length. If there are several such substrings, print the smallest in lexicographical order. If such substring does not exist, print one blank line.

Example

keven.in	keven.out
1 abacaba	abaaba
2 abacaba	aabaca
0 zzz	zz

Problem C. Casino

Input file:	`casino.in`	Output file:	`casino.out`
Time limit:	2 seconds		

Consider the following dice game. Player rolls a regular dice (with the numbers 1 through 6 on its sides) several times and calculates total amount of points each time. Player can finish his turn after any roll. Thus, player should make at least one roll. If the sum exceeds 21, player loses. When player finishes, croupier rolls a dice using the same scheme. Player wins, if he or she has more points than croupier. More formally, if player has S_p points and croupier has S_c points, than player wins if and only if ($S_p \le 21$ *and* ($S_c > 21$ *or* $S_c < S_p$)).

The optimal croupier's strategy allows casino to win in more than $\frac{2}{3}$ cases. But Andrew found the possibility to raise his chances to win! He will play together with Big Man. Big Man will bet X euros each game, while Andrew will bet 1 euro each game. In case of the win player gets his bet doubled. In case of the loss player gets nothing.

The scheme of the game is like following.

1. Big Man rolls dice several times and stops following the "best" strategy—the strategy that would be optimal if he played with croupier and without Andrew (he is really Big and doesn't care about little Andrew).

2. Than Andrew rolls dice several times. Andrew knows the amount of points Big Man has. Andrew is very smart boy and he realizes that croupier's strategy maximizes profit of the casino. So, Andrew uses optimal strategy using all those facts.

3. Finally, croupier rolls dice several times. As mentioned above, croupier uses optimal strategy.

We call strategy *optimal* if it maximizes expected profit. If after the next roll any player or croupier have the same expected profit with the current value, he or she will prefer to roll the dice.

As Andrew calculated if X is big enough, he has almost 50% chance to win!

Input

The input contains one integer X ($0 \le X \le 1000$).

Output

Print one number—probability of Andrew's win. Your answer must be accurate up to 10^{-5}.

Example

casino.in	casino.out
0	0.331176
1	0.345634

Problem D. Segments

Input file:	`segments.in`	Output file:	`segments.out`
Time limit:	6 seconds		

Consider a real axis and segments on it. You are to write a program that processes queries of two kinds.

Query '`+ L R`' adds a segment from point L to point R. You program should print the number of segments that are (non-strictly) inside the new one.

Query '`- L R`' removes one segment from point L to point R. If there is no such segment, ignore this query.

You may assume that no more than 1000 segments will present on the axis simultaneously.

Input

Each line of the input file contains one query. Query can be in the form '`- L R`' or '`+ L R`', where L and R are integers ($-10^9 \le L < R \le 10^9$). There will be no more than 250 000 queries.

Output

Print to the output one line for each '`+ L R`' query, containing the number of segments inside the added segment.

Example

segments.in	segments.out
+ 1 2 + 1 2 + 0 3 - 1 2 + 1 2	0 1 2 1

Problem E. Circuits

Input file:	`circuits.in`	Output file:	`circuits.out`
Time limit:	15 seconds		

Consider a digital electric circuit operating with binary signals in a discrete time. It contains input signals, wires, junctions and logical gates. We will use six kinds of logical gates: NOT, AND, NAND, OR, NOR, DFF. Each of them returns one output signal. The amount of input signals and details about return value are showed in the following table.

Gate	Inputs	Return value
NOT	1	Opposite to the input signal value.
AND	1 or more	Conjunction of the input signal values.
NAND	1 or more	Negation of the conjunction of the input signal values.
OR	1 or more	Disjunction of the input signal values.
NOR	1 or more	Negation of the disjunction of the input signal values.
DFF	1	Same as the input signal. Returned with the delay of one tick of discrete time.

The scheme will operate in the following manner. Before the first tick of the discret time each DFF gate is initialized with zero value. Then during certain amount of ticks several junctions (*input* junctions) will be feeded with input values. You are to write a program that will calculate binary values on a certain set of junctions (*output* junctions).

Your program will be given the description of the logical scheme, the set of junctions in which we are interested and the input values for each consecutive tick of the discrete time.

Input

The input file consists of two blocks. First block contains several lines. If the first symbol of a line is a sharp (`#`), than the line contains a comment. You can ignore it. You can also ignore empty lines. A line describing the input signal looks like `INPUT(junction)`, where `junction` is the name of the junction. A line describing the interesting junction looks like `OUTPUT(junction)`, where `junction` is the name of the junction. A junction name consists of small and capital Latin letters and digits and the underline character (_). The length of each name is less than 64 characters.

All other lines of the first block contain the description of new junctions. Such line can look like `j1 = op(j2)`, where `j1` and `j2` are the names of junctions and `op` is either `NOT` or `DFF`. It can also look like `j1 = op(j2[, j3...])`, where j's are the junction names and `op` is `AND`, `NAND`, `OR` or `NOR`. There will be no more than 5 000 junctions.

The second block of the input file starts with the line `INPUT VALUES`. Each of the following lines contains a sequence of zeroes and ones. Amount of digits equals to the amount of input signals. You should match these binary values to the input signal values, in the order the input signals given. There are no more than 500 lines in this block. Please refer to the sample for the details.

The input file size is less than 320 KB.

You may assume that the input data is correct. You may assume that the given circuit will operate correctly.

Output

For each line of the input signal values, print one line containing zeroes and ones. The digits should correspond to the values of the output signals, preserving their order in the input file.

Example

circuits.in	circuits.out
`INPUT(a)`	`00`
`INPUT(b)`	`11`
`x = DFF(a)`	`10`
`t = OR(a, b)`	
`y = AND(x, t)`	
`OUTPUT(x)`	
`OUTPUT(y)`	
`INPUT VALUES`	
`10`	
`11`	
`00`	
`# 4 inputs`	`1`
`# 1 output`	`0`
`# 3 D-type flipflops`	`0`
`# 2 inverters`	`0`
`# 8 gates (1 AND+1 NAND+2 ORs+4 NORs)`	`1`
	`1`
`INPUT(G0)`	
`INPUT(G1)`	
`INPUT(G2)`	
`INPUT(G3)`	
`OUTPUT(G17)`	
`G5 = DFF(G10)`	
`G6 = DFF(G11)`	
`G7 = DFF(G13)`	
`G14 = NOT(G0)`	
`G17 = NOT(G11)`	
`G8 = AND(G14, G6)`	
`G15 = OR(G12, G8)`	
`G16 = OR(G3, G8)`	
`G9 = NAND(G16, G15)`	
`G10 = NOR(G14, G11)`	
`G11 = NOR(G5, G9)`	
`G12 = NOR(G1, G7)`	
`G13 = NOR(G2, G12)`	
`INPUT VALUES`	
`0000`	
`0001`	
`0010`	
`0100`	
`1000`	
`1010`	

Problem F. TEX2HTML

Input file:	`tex2html.in`	Output file:	`tex2html.out`
Time limit:	2 seconds		

Andrew just finished writing his thesis. That was a big work and now Andrew wants some remarks from his friends. He thinks the best way to share the paper is to post it to his blog. Andrew wrote his thesis in the TEX format and now he needs to convert them to HTML. Please help Andrew to implement the most essential part of the converter—the mathematical formulas converter.

Andrew's formulas are always enclosed in the dollar signs (`$`). They contain Latin letters, digits, parentheses, white spaces, binary operators (`+-*/`), superscripts and subscripts. You should ignore all white spaces.

Superscripts are written using the cap character (`^`) and braces (`{}`). The part of the formula inside the braces is the superscript itself. You may assume that only non-whitespace character will follow the cap character. If the superscript contains only one character, the braces can be omitted. The superscript will not be followed by another superscript or subscript. E.g. `a^2` means a^2; `2^{2 + 2}` means 2^{2+2}.

Subscripts are written using the underline character (`_`) and braces (`{}`). The part of the formula inside the braces is the subscript itself. You may assume that only non-whitespace character will follow the underline character. If the subscript contains only one character, the braces can be omitted. The subscript will not be followed by another subscript or superscript. E.g. `x_i` means x_i; `P_{n+1-i}` means P_{n+1-i}.

Your program should produce HTML-like output using the following rules. Each letter should be italicized using `<i>` open tag and `</i>` close tag. These tags should enclose each maximal sequence of letters (don't forget to ignore white spaces). Each binary operator should be surrounded by the non-breaking spaces (`&nbsp;`). Superscripts should be enclosed in `<sup>` and `</sup>` tags. Subscripts should be enclosed in `<sub>` and `</sub>` tags.

You may assume that input will contain correct mathematical formula with only binary operators.

Input

The input file will contain several test cases. Each test case is a line with two dollar signs with the formula between them. No extra characters will appear on the line.

The input file size is less than 128 KB.

Output

Print the HTML version of each formula on a separate line.

Example

tex2html.in

```
$  a   $
$  a   -   (v-b)   $
$a_i+b_2$
$B^{k_i}$
```

tex2html.out

```
<i>a</i>
<i>a</i>&nbsp;-&nbsp;(<i>v</i>&nbsp;-&nbsp;<i>b</i>)
<i>a</i><sub><i>i</i></sub>&nbsp;+&nbsp;<i>b</i><sub>2</sub>
<i>B</i><sup><i>k</i><sub><i>i</i></sub></sup>
```

Problem G. Reihenfolge

Input file:	`reihenfolge.in`	Output file:	`reihenfolge.out`
Time limit:	2 seconds		

Consider positive integers A and B. Your task is to represent A as the algebraic sum of integer powers of B with the minimal possible number of terms. In other words,

$$A = \sum_{i=1}^{n} s_i B^{k_i},$$

where $s_i = -1$ or $s_i = 1$, k_i are integers and n should be minimized.

Input

The first line of the input contains positive integer A written without leading zeroes. A contains no more than 3000 digits. Second line contains integer B ($1 \le B \le 10^6$).

Output

Print one integer number n.

Example

`reihenfolge.in`	`reihenfolge.out`

`1120`
`10`
	`4`

Problem H. Weed

Input file:	`weed.in`	Output file:	`weed.out`
Time limit:	2 seconds		

Andrew has visited his garden for the last time many years ago. Today's property taxes are so high, so Andrew decided to sell his garden. The land was not cultivated for a long time and now it is probably a lot of weed on it. Andrew wants to remove everything from the ground before selling. Now he wants to estimate the amount of work.

The garden has the rectangular form and is divided into $N \times M$ equal squares. Andrew's memory is phenomenal. He remembers which squares were occupied by the weed. For the purpose of simplicity, Andrew thinks that each square is either fully occupied by the weed or completely free from it. Andrew likes botany and he knows that if some square is free from the weed but at least two of its adjacent squares are occupied by the weed (two squares are adjacent if they have common side), that square will be also occupied by the weed soon. Andrew is pretty sure that during last years weed occupied every square possible. Please help Andrew to estimate how many squares is occupied by the weed.

Input

The first line of the input contains integers N and M ($1 \le N, M \le 1000$). Next N lines contain M characters each. Character `X` denotes that the corresponding square is occupied by the weed. A period character (`.`) denotes an empty square.

Output

Print one integer denoting the number of squares occupied by the weed after so many years.

Example

weed.in	weed.out

`3 3`
`X..`
`.X.`
`.X.`
	`6`

`3 4`
`X..X`
`.X..`
`.X..`
	`12`

Problem I. VaR

Input file:	`var.in`	Output file:	`var.out`
Time limit:	2 seconds		

In economics and finance, *value at risk* (VaR) is a measure of how the market value of an asset or of a portfolio of assets is likely to decrease over a certain time period (usually over 1 day or 10 days) under usual conditions. It is typically used by security houses or investment banks to measure the market risk of their asset portfolios (market value at risk), but is actually a very general concept that has broad application. *From Wikipedia, the free encyclopedia.*

VaR has three parameters.

- The time horizon (period) to be analyzed (i. e. the period of time over which one plans to hold the assets in the portfolio—the "holding period"). The typical holding period is 1 day, although 10 days are used, for example, to compute capital requirements under the European Capital Adequacy Directive (CAD). For some problems, even a holding period of 1 year is appropriate.
- The confidence level at which the estimate is made. Popular confidence levels are usually 99% and 95%.
- The unit of the currency which will be used to denominate the value at risk (VaR).

In the purposes of this problem we will use the fixed confidence level value of 95% and Berland dollar as a currency unit.

The VaR is the maximum amount at risk to be lost from an investment (under 'normal' market conditions) over a given holding period, at a particular confidence level. As such, it is the converse of shortfall probability, in that it represents the amount to be lost with a given probability, rather than the probability of a given amount to be lost.

Consider an example of trading portfolio. Its market value in Berland dollars today is known, but its market value tomorrow is not known. The investment bank holding that portfolio might report that its portfolio has a 1-day VaR of $4 million at the 95% confidence level. This implies that (provided usual conditions will prevail over the 1 day) the bank can expect that, with a probability of 95%, the value of its portfolio will decrease by at most $4 million during 1 day, or, in other words, that, with a probability of 5%, the value of its portfolio will decrease by $4 million or more during 1 day.

The key thing to note is that the target confidence level (95% in the above example) is the given parameter here; the output from the calculation ($4 million in the above example) is the maximum amount at risk (the *value at risk*) for that confidence level.

In the following, *return* means percentage change in value.

A variety of models exist for estimating VaR. Each model has its own set of assumptions, but the most common assumption is that historical market data is our best estimator for future changes. The important assumption of "variance-covariance" model (VCV) is that risk factor returns are always (jointly) normally distributed and that the change in portfolio value is linearly dependent on all risk factor returns. There are also "historical simulation" and "Monte Carlo simulation" models, but here we will touch only the VCV model.

In the following, we will take the simple case, where the only risk factor for the portfolio is the value of the assets themselves. The following two assumptions enable to translate the VaR estimation problem into a linear algebraic problem:

(1) the portfolio is composed of the assets whose deltas are linear, more exactly: the change in the value of the portfolio is linearly dependent on (i.e. is a linear combination of) all the changes in the values of the assets, so that also the portfolio return is linearly dependent on all the asset returns;

(2) the asset returns are jointly normally distributed.

The implication of (1) and (2) is that the portfolio return is normally distributed because it always holds that a linear combination of jointly normally distributed variables is itself normally distributed.

We will use the following notation:

- there are N assets;
- μ_i = expected value of the return on asset i;
- μ_P = expected value of the return on the portfolio;
- σ_i = standard deviation of the return on asset i;
- σ_P = standard deviation of the the return on the portfolio;
- V_i = initial value of asset i (in currency units);
- V_P = initial value of the portfolio (in currency units);
- $\omega_i = V_i/V_P$;
- $\boldsymbol{\omega}$ = vector of all ω_i ($\boldsymbol{\omega}^T$ means transposed);
- $\mathbf{\Sigma}$ = covariance matrix, i. e. matrix of covariances between all N assets, i. e. an $N \times N$ matrix.

The covariance between two real-valued random variables X and Y, with expected values $\mathrm{E}(X) = \mu$ and $\mathrm{E}(Y) = \nu$ is defined as: $\mathrm{cov}(X, Y) = \mathrm{E}((X - \mu)(Y - \nu))$, where E is the expected value operator.

The calculation goes as follows.

$$\mu_P = \sum_{i=1}^{N} \omega_i \mu_i,$$

$$\sigma_P = \sqrt{\boldsymbol{\omega}^T \mathbf{\Sigma} \boldsymbol{\omega}}.$$

The normality assumption allows us to z-scale the calculated portfolio standard deviation to the appropriate confidence level. So for the 95% confidence level VaR we get:

$$VaR = -V_P(\mu_P - 1.644854\sigma_P).$$

You will be given historical data of the assets prices and the quantity of each asset in the portfolio. You should calculate the value at risk for that portfolio.

Input

The first line of the input contains integers T and N ($1 \le T \le 10\,000; 1 \le N \le 10$). Second line contains N integers denoting the quantity of each asset in the portfolio. These integers are positive and not exceed 1000.

Each of the next $T + 1$ lines contains N positive numbers $price_t^{(i)}$ written with 2 digits after the decimal point ($0 \le t \le T$; $1 \le i \le N$). The first of those lines contains today's market prices (day 0) for one unit of the corresponding asset. Next line corresponds to the previous working day (day 1), and so on. Today's prices should be used to calculate the initial value of assets. The *return* of asset i on day t equals to $(price_{t-1}^{(i)} - price_t^{(i)})/price_t^{(i)}$. Prices are positive and do not exceed 100 000.00.

Output

Print one number, the VaR for the given data. Your answer must be accurate up to 10^{-2}.

Example

var.in	var.out

5 2
200 10
31.11 489.75
31.04 488.04
31.10 497.28
31.15 504.28
31.22 505.00
30.69 501.02
	150.89

Problem J. Revolution

Input file:	`revolution.in`	Output file:	`revolution.out`
Time limit:	6 seconds		

During the last parliament elections in Berland two political parties collected the same amount of votes. Now the first party hired you to help them with their plan of a revolution.

During the revolution they want to split the country into two parts. The party is very democratic, so each member has its own separation plan.

The border of Berland can be considered as a convex polygon on the plane. The polygon can have three or more consecutive vertices lay on one line. Each separation plan is represented by the line of the cut.

Your program should calculate the area of the smallest part of the Berland for each separation plan.

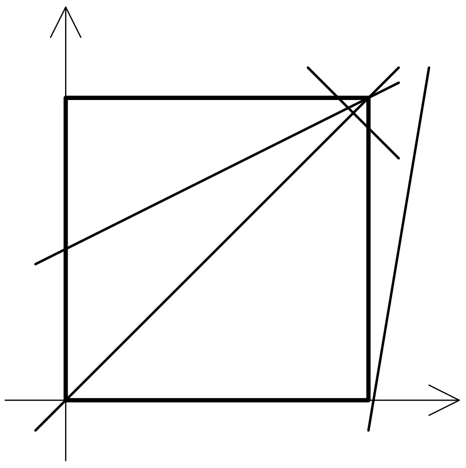
Input

The first line of the input contains integer N ($3 \le N \le 50\,000$). Next N lines contain coordinates x_i, y_i of the Berland border in the clockwise or counterclockwise order. The polygon is non-degenerate. The next line contains integer P ($1 \le P \le 50\,000$). Next P lines contain coordinates $x_{1,j}$, $y_{1,j}$, $x_{2,j}$, $y_{2,j}$ of two distinct points on the separation line. All coordinates are real and do not exceed 10 000 by the absolute value. They are given with no more than 4 significant digits after decimal point.

Output

Print P lines. Each line should contain the area of the smallest part after separating Berland with the corresponding line. Print value with at least 6 digits after decimal point. Absolute or relative error must be less than 10^{-5}.

Example



revolution.in	revolution.out

5
0 0
0 5
0 10
10 10
10 0
4
0 0 10 10
9 10 10 9
10 -1 12 11
10 10 0 5
	50.000000
	0.500000
	0.000000
	25.000000