Problem A. Arrangement of Contest

Input file:	arrange.in
Output file:	arrange.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Little Dmitry and little Petr want to arrange a contest. Their little friends submitted several task proposals and now Dmitry and Petr want to select some of them for the contest. As they are just little boys, they cannot estimate quality of tasks, but they know for sure that in *good* contest title of the first problem starts with A, the title of the second one — with B, and so on.

Given titles of the proposed tasks, help little brothers to determine the maximal number of problems in a *good* contest they can arrange.

Input

The first line contains single integer n — the number of problem proposals received by the little brothers ($1 \le n \le 100$).

Next n lines contain titles of proposed problems, one per line. The length of each title does not exceed 30 characters. Each title starts with an uppercase letter and contains only English letters, digits and underscores.

Output

Output a single number — the maximal number of problems in a *good* contest. In case there is no *good* contest that may be arranged, output 0.

Examples

arrange.in	arrange.out
12 Arrangement_of_Contest Ballot_Analyzing_Device Correcting_Curiosity Dwarf_Tower Energy_Tycoon Flight_Boarding_Optimization Garage Heavy_Chain_Clusterization Intellectual_Property J Kids_in_a_Friendly_Class Lonely_Mountain	12
3 Snow_White_and_the_7_Dwarfs A_Problem Another_Problem	1
2 Good_Problem Better_Problem	0

Problem B. Ballot Analyzing Device

Input file:	bad.in
Output file:	bad.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Election committee of Flatland is preparing for presidential elections. To minimize human factor in ballot counting they decided to develop an automated Ballot Analyzing Device (BAD).

There are n candidates running for president. The ballot contains one square field for each candidate. The voter must mark exactly one of the fields. If no field is marked or there are two or more marked fields, the ballot is invalid. Each voter puts his/her ballot to a special scanner in BAD. The scanner analyzes marks on the ballot and creates a special *voting string* of n characters: '`X`' for marked field and '`.`' for unmarked one. Now voting strings must be analyzed to get the report. Your task is to develop a report generator for BAD.

Given voting strings for all ballots, your program must print the voting report. Candidates in the report must be arranged in order of decreasing number of votes. If two candidates have the same number of votes, they must have the same order as in a voting ballot. For each candidate calculate his/her result in percent (if the candidate received p votes, the result in percent is $100p/m$). The last line of the report must indicate the percentage of the invalid ballots.

Input

The first line contains two integers n and m — the number of candidates and the number of ballots ($2 \le n \le 10$; $1 \le m \le 1000$). The following n lines contain last names of candidates. Each name is a string of at most 100 English letters. There is no candidate named "`Invalid`".

Then m lines follow, each of them contains one voting string.

Output

Print $n+1$ lines. First print results for candidates in percent. For each candidate print his/her last name followed by a space and then his/her result in percent and a percent sign '`%`'. The last line must specify the percentage of invalid ballots: a word "`Invalid`" followed by a space, the percentage of invalid ballots and a percent sign.

Round all numbers to exactly two digits after the decimal point. If the number is exactly in the middle of two representable numbers, output the greater one (e.g. output "`12.35`" for 12.345).

Example

bad.in	bad.out
4 7 Loudy Apples Dogman Miller .X.. X... .... ..X. ..XX ..X. ..X.	Dogman 42.86% Loudy 14.29% Apples 14.29% Miller 0.00% Invalid 28.57%

Problem C. Correcting Curiosity

Input file:	curiosity.in
Output file:	curiosity.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Curiosity is the rover that explores the Gale Crater on Mars. Recently it found an evidence of water in Martian soil, which will make it easier to plan the future manned missions.

Curiosity can communicate with Earth directly at speeds up to 32 Kbit/s, but on average 14 minutes and 6 seconds will be required for signals to travel between Earth and Mars.

"You have just seen a stone and applied brakes, but you know that the rover is already passing that stone" — Matt Heverly, the rover's driver, explains. "So we just plan the route, then write down a list of simple textual commands: move one meter ahead, turn left, make a photo and so on".

Sometimes it is necessary to react very fast to some unexpected events. For example, if the cameras have seen something interesting, then you might want to change the route of the rover to make an additional photo. To do that, you send a substitution command of the form `s/`⟨*string*⟩`/`⟨*replacement*⟩`/g`. This replaces all occurrences of ⟨*string*⟩, starting with the leftmost one, to ⟨*replacement*⟩.

More formally, if A is a non-empty string and B is a string, then to apply the substitution command `s/`A`/`B`/g` to a string S, you should do the following:

1. Find the leftmost occurrence of A in S, such that $S = S_L + A + S_R$.
2. If there is no such occurrence, stop. Then, S is the answer.
3. Let R be the result of applying `s/`A`/`B`/g` to S_R.
4. The answer is $S_L + B + R$.

This means that:

1. If there are two overlapping occurrences of A in S, only the leftmost one is replaced. For example, applying "`s/aba/c/g`" to "`abababa`" yields "`cbc`": after replacing the first occurrence of "`aba`" the string turns to "`cbaba`", and only the last occurrence of "`aba`" can be replaced after that.
2. No substitution uses the results of previous substitutions. For example, applying "`s/a/ab/g`" to "`a`" yields "`ab`", applying "`s/a/ba/g`" to "`a`" yields "`ba`".

You know that the longer is the command, the bigger is the time necessary to transmit it. So, you have to write a program that finds shortest command that transforms the initial string to the final string.

Input

The first line contains the initial and the final strings. Both strings are non-empty and their lengths do not exceed 2000 characters. The strings contain only English letters, spaces and punctuation signs (commas, colons, semicolons and hyphens: '`,`', '`:`', '`;`', '`-`'). The given strings are not equal.

Output

Output the substitution command that transforms initial string to final string and has the minimum length. If there are several shortest substitution commands, output any of them.

Examples

curiosity.in	curiosity.out
move left, move right; move up move left, move down, move up	s/right;/down,/g
If not found: move x; else move -x If found: move x; else move -x	s/ not//g
abababa cbc	s/aba/c/g

Problem D. Dwarf Tower

Input file:	dwarf.in
Output file:	dwarf.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Little Vasya is playing a new game named "Dwarf Tower". In this game there are n different items, which you can put on your dwarf character. Items are numbered from 1 to n. Vasya wants to get the item with number 1.

There are two ways to obtain an item:

- You can buy an item. The i-th item costs c_i money.
- You can craft an item. This game supports only m types of crafting. To craft an item, you give two particular different items and get another one as a result.

Help Vasya to spend the least amount of money to get the item number 1.

Input

The first line of input contains two integers n and m ($1 \le n \le 10\,000; 0 \le m \le 100\,000$) — the number of different items and the number of crafting types.

The second line contains n integers c_i — values of the items ($0 \le c_i \le 10^9$).

The following m lines describe crafting types, each line contains three distinct integers a_i, x_i, y_i — a_i is the item that can be crafted from items x_i and y_i ($1 \le a_i, x_i, y_i \le n$; $a_i \ne x_i$; $x_i \ne y_i$; $y_i \ne a_i$).

Output

The output should contain a single integer — the least amount of money to spend.

Examples

dwarf.in	dwarf.out
5 3 5 0 1 2 5 5 2 3 4 2 3 1 4 5	2
3 1 2 2 1 1 2 3	2

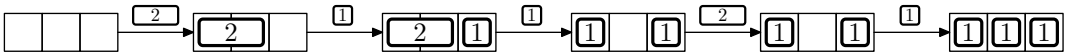
Problem E. Energy Tycoon

Input file:	`energy.in`
Output file:	`energy.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Little Vasya is playing a new computer game — turn-based strategy "Energy Tycoon".

The rules of the game are quite simple:

- The board contains n slots arranged in a line.
- There are power plants, one power plant occupies one or two consecutive slots, and produces one unit of energy.
- Each turn the game allows you to build one new power plant, you can put it on the board if you wish. If there is no place for the new power plant, you can remove some older power plants.
- After each turn, the computer counts the amount of energy produced by the power plants on the board and adds it to the total score.



Vasya already knows the types of power plant he will be able to build each turn. Now he wants to know, what the maximum possible score he can get is. Can you help him?

Input

The first line of the input contains one integer n ($1 \le n \le 100\,000$) — the number of slots on the board. The second line contains the string s. The i-th character of the string is 1 if you can build one-slot power plant at the i-th turn and the character is 2 if you can build two-slot power plant at the i-th turn. The number of turns does not exceed 100 000.

Output

The output should contain a single integer — the maximal score that can be achieved.

Examples

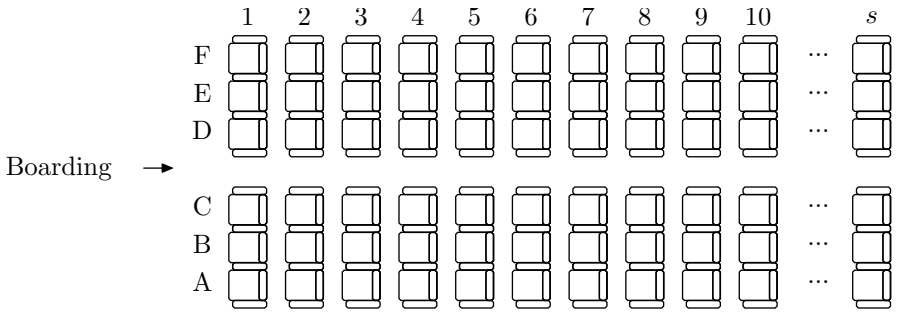
energy.in	energy.out
3 21121	10
2 12	2
2 211	4

The picture shows the optimal sequence of moves for the first example.

Problem F. Flight Boarding Optimization

Input file:	`flight.in`
Output file:	`flight.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Peter is an executive boarding manager in Byteland airport. His job is to optimize the boarding process. The planes in Byteland have s rows, numbered from 1 to s. Every row has six seats, labeled A to F.



There are n passengers, they form a queue and board the plane one by one. If the i-th passenger sits in a row r_i then the difficulty of boarding for him is equal to the number of passengers boarded before him and sit in rows $1 \dots r_i - 1$. The total difficulty of the boarding is the sum of difficulties for all passengers. For example, if there are ten passengers, and their seats are 6A, 4B, 2E, 5F, 2A, 3F, 1C, 10E, 8B, 5A, in the queue order, then the difficulties of their boarding are 0, 0, 0, 2, 0, 2, 0, 7, 7, 5, and the total difficulty is 23.

To optimize the boarding, Peter wants to divide the plane into k zones. Every zone must be a continuous range of rows. Than the boarding process is performed in k phases. On every phase, one zone is selected and passengers whose seats are in this zone are boarding in the order they were in the initial queue.

In the example above, if we divide the plane into two zones: rows 5–10 and rows 1–4, then during the first phase the passengers will take seats 6A, 5F, 10E, 8B, 5A, and during the second phase the passengers will take seats 4B, 2E, 2A, 3F, 1C, in this order. The total difficulty of the boarding will be 6.

Help Peter to find the division of the plane into k zones which minimizes the total difficulty of the boarding, given a specific queue of passengers.

Input

The first line contains three integers n ($1 \le n \le 1000$), s ($1 \le s \le 1000$), and k ($1 \le k \le 50$; $k \le s$). The next line contains n integers r_i ($1 \le r_i \le s$).

Each row is occupied by at most 6 passengers.

Output

Output one number, the minimal possible difficulty of the boarding.

Example

flight.in	flight.out
10 12 2 6 4 2 5 2 3 1 11 8 5	6

Problem G. Garage

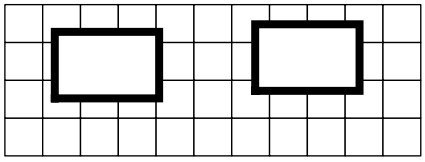
Input file:	garage.in
Output file:	garage.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Wow! What a lucky day! Your company has just won a social contract for building a garage complex. Almost all formalities are done: contract payment is already transferred to your account.

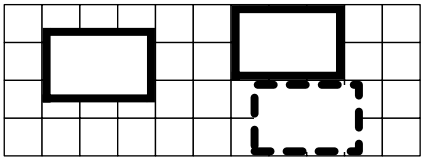
So now it is the right time to read the contract. Okay, there is a sandlot in the form of $W \times H$ rectangle and you have to place some garages there. Garages are $w \times h$ rectangles and their edges must be parallel to the corresponding edges of the sandlot (you may not rotate garages, even by 90°). The coordinates of garages may be non-integer.

You know that the economy must be economical, so you decided to place as *few* garages as possible. Unfortunately, there is an opposite requirement in the contract: placing maximum possible number of garages.

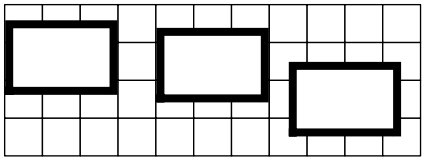
Now let's see how these requirements are checked... The plan is accepted if it is impossible to add a new garage without moving the other garages (the new garage must also have edges parallel to corresponding sandlot edges).



Accepted optimal plan



Rejected plan



Accepted, but non-optimal plan

Time is money, find the minimal number of garages that must be ordered, so that you can place them on the sandlot and there is no place for an extra garage.

Input

The only line contains four integers: W, H, w, h — dimensions of sandlot and garage in meters. You may assume that $1 \le w \le W \le 30\,000$ and $1 \le h \le H \le 30\,000$.

Output

Output the optimal number of garages.

Examples

garage.in	garage.out
11 4 3 2	2
10 8 3 4	2
15 7 4 2	4

The plan on the first picture is accepted and optimal for the first example. Note that a rotated (2×3) garage could be placed on the sandlot, but it is prohibited by the contract.

Problem H. Heavy Chain Clusterization

Input file:	`heavy.in`
Output file:	`heavy.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

A group of biologists is trying to find a cure for a viral disease. They have tried many antibodies of various origins that could potentially fight the viral antigens, and have selected n antibodies that seem to work best during experiments.

Each antibody is identified by its *heavy chain* — a sequence of amino acids.

The set of antibodies form a *similarity cluster*, if at least one of the following holds:

- k-prefixes (first k amino acids) of all their heavy chains are equal;
- k-suffixes (last k amino acids) of all their heavy chains are equal.

In order to simplify the future research, biologists want to group antibodies to similarity clusters.

You need to split the given antibodies to a minimum number of similarity clusters.

Input

The first line contains two integers n and k — the number of heavy chains and the length of sequence of amino acids to coincide ($1 \le n \le 5\,000$, $1 \le k \le 550$).

The following n lines contain sequences of amino acids that form heavy chains of antibodies. Each amino acid described with an uppercase English letter. Each heavy chain contains at least k and no more than 550 amino acids.

Output

The first line of output must contain a single integer — the minimum number of similarity clusters. The following lines must contain descriptions of clusters, one per line.

Each description starts with m_i — the number of antibodies in the cluster and is followed by m_i integers — numbers of these antibodies. Antibodies are numbered in the order of appearance in the input starting from one.

Each antibody must be present in exactly one cluster.

Examples

heavy.in	heavy.out
4 1 AA AB BB BA	2 2 1 2 2 3 4
3 2 ABA BAB XY	3 1 1 1 2 1 3

Problem I. Intellectual Property

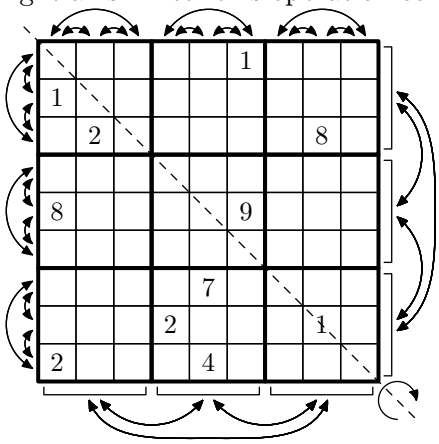
Input file:	`intellectual.in`
Output file:	`intellectual.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Erast Kopi is famous Sudoku puzzle designer. Resounding success of his puzzle compilations caused a number of imitations and plagiarisms. Prior to sending a lawsuit he decided to get more evidence.

Sudoku puzzle is a table 9×9, divided into 3×3 subtables of 3×3 cells each. Each cell may contain a digit from 1 to 9. The task is to fill empty cells with digits in a way that each row, each column and each of the 9 subtables 3×3 contains each digit from 1 to 9 exactly once.

Kopi has a database of Sudoku puzzles and he wants to check if it contains similar puzzles. The puzzle P is similar to the puzzle Q, if it is possible to transform the puzzle P into the puzzle Q using a sequence of the following operations:

- choose two digits x and y and replace all digits x with y and vice versa;
- swap two triples of rows: $(1, 2, 3)$, $(4, 5, 6)$, $(7, 8, 9)$;
- swap two rows in one triple of rows;
- swap two triples of columns: $(1, 2, 3)$, $(4, 5, 6)$, $(7, 8, 9)$;
- swap two columns in one triple of columns;
- flip along top-left — bottom-right axis. After this operation columns become rows and vice versa.



Help Kopi to find similar puzzles in his database.

Input

The first line of the input contains single integer n — the number of puzzles in the database ($1 \le n \le 20$).

The rest of the input contains description of n puzzles: $P_1, P_2, \ldots, P_n$. Each puzzle is described by nine lines that contain nine characters each. Each character is either a digit from 1 to 9, or a dot (‘`.`’) denoting an empty cell. An empty line separates consecutive puzzles in the database.

There are no spaces in the input file.

The puzzles are *not* guaranteed to be solvable.

Output

Check if the puzzle P_1 is similar to puzzles $P_2, P_3, \ldots, P_n$ (in this order), than check if the puzzle P_2 is similar to puzzles $P_3, P_4, \ldots, P_n$ (in this order) and so on.

If the puzzle P_i is similar to the puzzle P_j ($1 \le i < j \le n$) output “`Yes`”, otherwise output “`No`”. If the answer is positive, the next line should contain an integer q_{ij} — the number of operations required to transform the puzzle P_i to the puzzle P_j. The number of operations is not required to be minimal, however it must not exceed 1000. In the following q_{ij} lines write the operations that transform the puzzle P_i to the puzzle P_j, one per line.

Operations are encoded in the following way:

- "`D` x y" for swapping digits x and y;
- "`R` a b" for swapping triples of rows $(3a-2, 3a-1, 3a)$ and $(3b-2, 3b-1, 3b)$;
- "`r` a b" for swapping rows a and b, rows must belong to same triple of rows;
- "`C` a b" for swapping triples of columns $(3a-2, 3a-1, 3a)$ and $(3b-2, 3b-1, 3b)$;
- "`c` a b" for swapping columns a and b, columns must belong to same triple of columns;
- "`F`" for flipping along top-left — bottom-right axis.

The columns are numbered from left to right and the rows are numbered from top to bottom as they are given in the input file, starting from one.

Example

intellectual.in	intellectual.out
4 .....1... 1....... .2.....8. .... 8....9... ....7.... ...2...1. 2...4.... ....2.... ...7.4... 8.......9 .8...2..1 ..2...... ....1.8.... .... 1....... ....1... 1....... .2.....8. .... 8....9... ....7.... ...2...1. 2...4....	Yes 7 C 1 2 D 5 3 F r 7 9 c 6 5 C 2 3 D 1 8 No Yes 0 No Yes 8 R 1 2 C 2 3 c 4 5 F r 5 6 c 7 9 D 1 8 D 3 5 No

Problem J. J

Input file:	`j.in`
Output file:	`j.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

> The J programming language, developed in the early 1990s by Kenneth E. Iverson and Roger Hui, is a synthesis of APL (also by Iverson) and the FP and FL function-level languages created by John Backus.
>
> Wikipedia. J (programming language)

APL language family is famous for its support of advanced operations on vectors and arrays, and J is not an exception. For example, all unary and binary numeric operations in these languages by default are applicable to vectors and arrays of different dimensions. Plus operation (‘`+`’) can add not only scalars, like in other languages, but also scalars and vectors (the scalar is added to each component of the vector), or vectors and vectors (the vectors are added component-wise).

The expressive power of J is amazing (as well as its cryptic syntax), but for this problem we need just a small subset of the language. We consider a single expression, where we may use one vector variable X, one scalar variable N — the length of the vector X, and the following operations:

- We can add (‘`+`’), subtract (‘`-`’) or multiply (‘`*`’) two vectors, vector and scalar, or two scalars.
- We can use unary minus (‘`-`’) and unary squaring operations (‘`*:`’) for scalars and vectors (component-wise).
- We can *fold* a vector with plus operation (‘`+/`’) — that is, compute the sum of a vector (unary operation).

Operations are evaluated right-to-left, natural precedence of operations is ignored in J. The order of evaluation can be altered by parentheses. More precisely the syntax is specified in the following BNF.

$$
\begin{aligned}
\langle \text{expression} \rangle &::= \langle \text{term} \rangle \mid \langle \text{term} \rangle\ (\text{`+'} \mid \text{`-'} \mid \text{`*'})\ \langle \text{expression} \rangle \mid (\text{`-'} \mid \text{`*:'} \mid \text{`+/'})\ \langle \text{expression} \rangle \\
\langle \text{term} \rangle &::= \text{`('}\ \langle \text{expression} \rangle\ \text{`)'} \mid \text{`X'} \mid \text{`N'} \mid \langle \text{number} \rangle \\
\langle \text{number} \rangle &::= (\text{`0'} \mid \text{`1'} \mid \ldots \mid \text{`9'})^{+}
\end{aligned}
$$

To correctly impose one more limitation on expression syntax, let us define *complexity* of an expression:

- complexity of scalars (numbers, ‘`N`’, and result of fold) is zero;
- complexity of ‘`X`’ is one;
- complexity of addition and subtraction is the maximum of their operands’ complexities;
- complexity of multiplication is the sum of its operands’ complexities;
- complexity of unary squaring is twice the complexity of its operand.

For example, the complexity of expression “`(3-+/*:*:X)-X**:X`” is 3, while the complexity of its subexpression “`*:*:X`” is 4.

Your program is given a scalar-valued expression and a value of the vector X, and it should compute the expression result modulo 10^9. The complexity of each subexpression in the given expression does not exceed 10.

Input

The first line contains one integer number N ($1 \le N \le 10^5$) — the length of the vector X.

The second line contains N integers — components of the vector X ($0 \le X_i < 10^9$).

The third line contains the expression to be computed, a non-empty string of not more than 10^5 symbols. Each number in the expression is less than 10^9. The fold is never applied to a scalar.

Output

Output a single integer number — the expression result modulo 10^9.

Examples

j.in	j.out
5 1 2 3 4 5 +/*:X	55
5 1 2 3 4 5 N++/X-X+1	0
3 11 56 37 +/(3-+/*:*:X)-X**:X	964602515

The first expression computes squared length of the vector X in Euclidean metrics.

The second expression result does not depend on X and always equals to zero.

The actual result of the third expression is -35397485.

Problem K. Kids in a Friendly Class

Input file:	`kids.in`
Output file:	`kids.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Kevin resembles his class in primary school. There were girls and boys in his class. Some of them were friends, some were not. But if one person considered another person a friend, the opposite was also true.

Interestingly, every girl had exactly a friends among girls and exactly b friends among boys, whereas every boy had exactly c friends among girls and exactly d friends among boys.

Kevin does not remember the size of his class. Help him reconstruct the class with minimal possible number of kids, such that the above conditions are satisfied.

Input

The only line contains four integers a, b, c, and d ($1 \le a, b, c, d \le 50$).

Output

Output an example of a class of minimal possible size satisfying the above conditions.

The first line should contains two positive integers: m — the number of girls, and n — the number of boys.

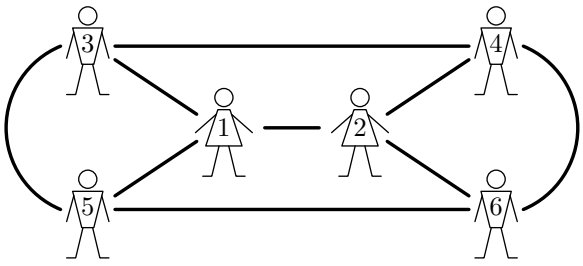
Let's assign numbers 1 through m to the girls and $m + 1$ through $m + n$ to the boys.

Each of the next lines should contain a pair of distinct integers describing a pair of friends by their numbers. Each pair of friends should appear exactly once in this list.

Example

kids.in	kids.out
1 2 1 2	2 4 1 2 1 3 1 5 2 4 2 6 3 4 3 5 4 6 5 6

The class from the example output is shown below:

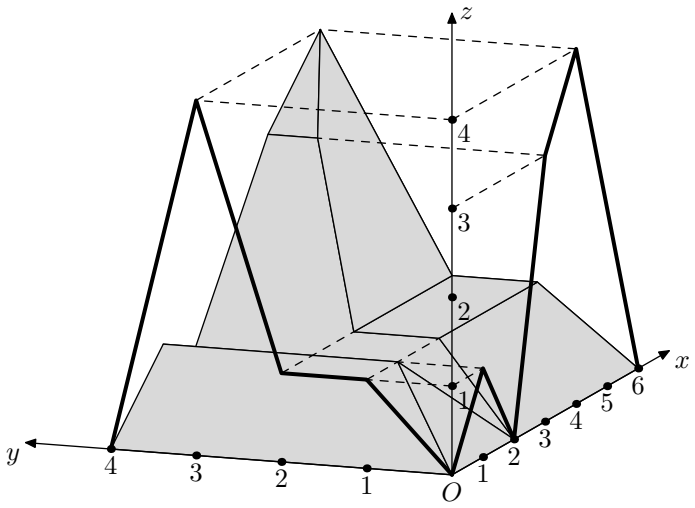


Problem L. Lonely Mountain

Input file:	`lonely.in`
Output file:	`lonely.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

> "This was made by Thror, your grandfather, Thorin", he said in answer to the dwarves' excited questions. "It is a plan of the Mountain."
>
> J. R. R. Tolkien. The Hobbit, or There and Back Again

The plan of the Lonely Mountain consists of two parallel projections of the mountain to two projection planes. Both planes are perpendicular to the ground and each other. Each projection has a mountain-like shape.



Since Bilbo Baggins has never seen the mountain, he tries to imagine it. Is it really the *Lonely* Mountain or some ridges and other mountains surround it? In any case, it must be tremendous to hold the whole dwarves' kingdom!

Bilbo decided to estimate the maximum possible volume of the Lonely Mountain and nearby mountains (if any) based on the plan provided by Gandalf.

Input

The first line contains single integer number n_x — the number of points in the parallel projection of the mountain to the plane Oxz ($2 \le n_x \le 100\,000$). The second line contains n_x pairs of integer numbers x_i, z_i — the coordinates of the polygonal chain, representing the projection ($-10^9 \le x_1 < x_2 < x_3 < \cdots < x_{n_x} \le 10^9$; $0 \le z_i \le 10^9$; $z_1 = z_{n_x} = 0$).

The following two lines contain projection to the Oyz plane in the same format.

Output

The only line of the output file must contain a single number V — the maximum possible volume of the Lonely Mountain.

The absolute or relative precision of you answer should be at least 10^{-6}. E.g. if V' is the actual maximum possible volume, the following must hold: $\min(|V - V'|, \frac{|V-V'|}{V'}) \le 10^{-6}$.

If there are no mountains corresponding to the given projections, output a single line "`Invalid plan`".

Examples

lonely.in	lonely.out
6 0 0 1 1 2 0 3 3 4 4 6 0 5 0 0 1 1 2 1 3 4 4 0	21.824074074074073
3 -1 0 0 1 2 0 4 0 0 1 1 2 2 3 0	Invalid plan