Problem Set

Please check that you have 12 problems that are spanned across 24 pages in total (including this cover page).

The memory limits for the twelve problems are all the same, 512MB.

Problem A
Circuits
Time Limit: 1 Second

There are a number of electronic circuits, such as CPU's, ROM's, RAM's, to be printed in a single chip consisting of multiple layers. Due to some design restriction, there can be only two electrical wires that are horizontal segments. Your job is to find two horizontal wires that together connect as many circuits as possible so that the electric signals go through the circuits.

This problem can be stated formally as follows. There are n axis-aligned rectangles in the plane. Each of the rectangles represents a circuit to be printed in the chip. The rectangles may overlap each other. You are supposed to find two horizontal lines such that the total number of rectangles intersected by the two lines is maximized. We also say that a rectangle is intersected by a horizontal line if the line contains the top side or the bottom side of the rectangle. If a rectangle is intersected by both the lines, it is counted only once for the total number.

For example, let's consider 5 rectangles shown in Figure A.1. Figure A.1(c) shows two horizontal lines (red dashed lines) that intersect all 5 rectangles while the two horizontal lines (red dashed lines) in Figure A.1(b) intersect 4 rectangles.

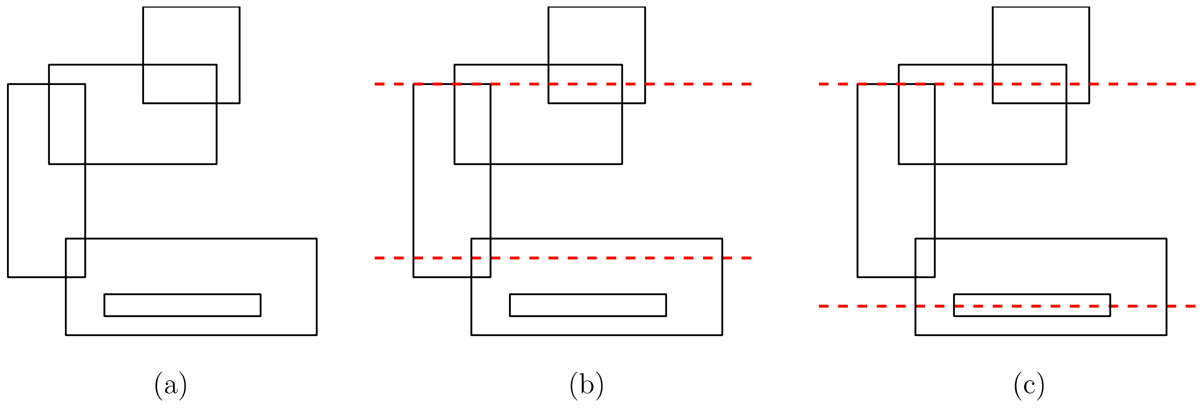


Figure A.1: (a) 5 axis-aligned rectangles. (b) Two horizontal lines that intersect 4 rectangles. (c) Two horizontal lines that intersect 5 rectangles.

Given a set of axis-aligned rectangles, write a program to find two horizontal lines such that the total number of rectangles intersected by the two lines is maximized.

Input
Your program is to read from standard input. The first line contains a positive integer n representing the number of axis-aligned rectangles in the plane, where $3 \leq n \leq 100{,}000$. It is followed by n lines, each contains four integers u_x, u_y, v_x, and v_y (with $u_x < v_x$ and $u_y > v_y$) representing the (x, y)-coordinates, (u_x, u_y), of the top-left corner and the (x, y)-coordinates, (v_x, v_y), of the bottom-right corner of an axis-aligned rectangle, where $-10{,}000{,}000 \leq u_x, u_y, v_x, v_y \leq 10{,}000{,}000$.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the maximum total number of rectangles that can be intersected by two horizontal lines.

The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1

5
0 13 4 4
2 14 11 9
7 17 12 12
3 5 16 0
5 2 13 1
	5

Sample Input 2	Output for the Sample Input 2

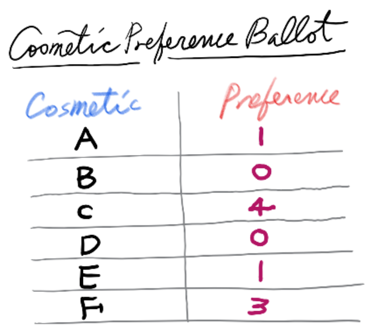
5
0 4 4 0
1 3 3 1
5 8 9 4
0 12 4 8
1 11 3 9
	4

Problem B
Cosmetic Survey
Time Limit: 1.5 Seconds

ICP(International Cosmetic Perfection) Company plans to survey the preferences of new m cosmetics in order to know which cosmetic is the most preferred. For this survey, ICPC selected n people as evaluators. Each evaluator must submit an ordered list of the preferences on those m cosmetics. Each evaluator can rank the cosmetics as follows:

- An evaluator can rank cosmetics with positive integer numbers, but the positive integers are not necessarily consecutive. The cosmetics with the smallest positive number are the most preferred ones, and the ones with the second smallest positive number are the second most preferred ones, and so on.
- An evaluator can give the same preference to more than one cosmetic. This indicates that this evaluator has no preference among those cosmetics.
- An evaluator may not rank some cosmetics. The unranked cosmetics are marked with number 0 in the ordered list. This indicates that this evaluator strictly prefers all ranked to all unranked ones, and has no preference among all unranked ones.

For example, the right figure shows an ordered list of the preferences on six cosmetics A, B, C, D, E, F that an evaluator submitted. The cosmetics A, C, E, F are ranked with nonconsecutive integers of $1, 4, 1, 3$, respectively. The other two cosmetics B and D are not ranked, thus they are marked with 0. A and E have the same preference, which are the most preferred cosmetics because their preference number is the smallest positive integer. The unranked B and D are less preferred than the ranked ones. As a result, the preference order of this evaluator is $A = E \succ F \succ C \succ B = D$, where $X \succ Y$ means that X is strictly preferred to Y and $X = Y$ means the same preference.



Which cosmetics are the most preferred ones from n evaluators in this preference survey? We must now define which one is *preferred* to another. Let $d(X, Y)$ be the number of evaluators who strictly prefer X to Y. A path from X to Y is a sequence of distinct cosmetics $C_1, \dots, C_k$ such that $C_1 = X, C_k = Y$, and $d(C_t, C_{t+1}) > d(C_{t+1}, C_t)$ for every $t = 1, \dots, k-1$. The *preference index* of this path is defined as the minimum of $d(C_t, C_{t+1})$ for $1 \le t < k$. For two distinct cosmetics X and Y that are connected by a path from X to Y, the *preference strength* $S(X, Y)$ is the maximum preference index over all paths from X to Y. If there is no path from X to Y, then $S(X, Y)$ is defined as zero. Cosmetic X is one of the most preferred cosmetics from this survey if and only if $S(X, Y) \ge S(Y, X)$ for every Y other than X. You note that it has been known for this type of surveys that there always exists at least one cosmetic that is preferred the most.

Given preference lists of n evaluators for m cosmetics, write a program to output all the most preferred cosmetics.

Input
Your program is to read from standard input. The input starts with a line containing two integers, m and n ($1 \le m \le 500, 1 \le n \le 500$), where m is the number of cosmetics and n is the number of evaluators. The cosmetics are numbered from 1 to m, and the evaluators are numbered from 1 to n . In the following n lines, the i-th line contains m nonnegative integers that represent the preference values for the m cosmetics of the evaluator i. The preference values are ordered from the cosmetic 1 to the cosmetic m. The zero values in the

list mean that the evaluator unranked the corresponding cosmetics. The preference values are no more than 10^6.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the numbers of the cosmetics that are preferred the most. Such cosmetic numbers must appear in increasing order.

The following shows sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1

3 4
1 1 1
0 0 0
2 2 2
3 3 3
	1 2 3

Sample Input 2	Output for the Sample Input 2

4 5
1 0 1 1
1 1 5 2
2 1 3 6
0 1 0 1
1 2 2 2
	1 2

Sample Input 3	Output for the Sample Input 3

5 4
0 1 0 2 1
1 7 2 1 0
4 5 2 3 3
1 2 9 0 2
	5

Problem C
Disks Arrangement
Time Limit: 1 Second

The n disks shall be placed in a plane such that they touch the x-axis from above and such that no two disks overlap. In a valid placement, each disk touches the x-axis in its lowest point. The lowest point is called the bottom-point of the disk. The bottom-points induce a linear left-to-right order on the disks.

We will concentrate on a linear instance. The linear instance is a set of disks $\{D_1, D_2, \dots, D_n\}$ such that for any ordering $\sigma : \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}$ of disks, that is, $D_{\sigma(1)}\, D_{\sigma(2)} \cdots D_{\sigma(n)}$, there is a placement such that each disk $D_{\sigma(i)}$ touches only the two disks $D_{\sigma(i-1)}$ and $D_{\sigma(i+1)}$ except for $D_{\sigma(1)}$ and $D_{\sigma(n)}$, and $D_{\sigma(1)}$ and $D_{\sigma(n)}$ touch only the disk $D_{\sigma(2)}$ and $D_{\sigma(n-1)}$, respectively. See Figure C.1. An example which is not a linear instance is shown in Figure C.2.

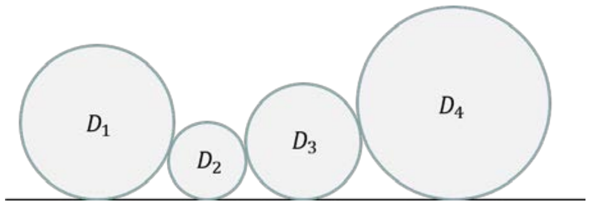


Figure C.1

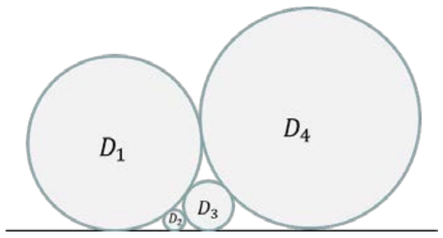


Figure C.2

It is known that if the ratio between the largest and smallest radius of the disks is less than four, then the disks induce the linear instance. So all the inputs of this problem shall satisfy this condition.

For a given linear instance of disks, find a valid placement to minimize the horizontal distance between the leftmost point and the rightmost point of the disks. See Figure C.3.

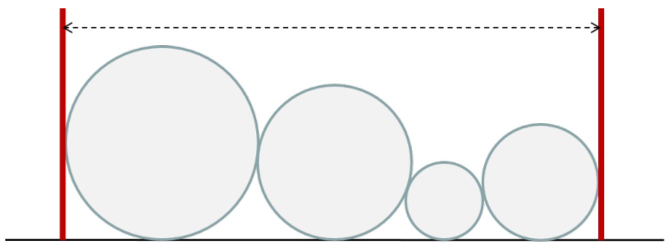


Figure C.3

Input

Your program is to read from standard input. The input starts with a line containing an integer n ($1 \le n \le 1{,}000$), where n is the number of disks. The next line contains n integer numbers each of which is a radius a of a disk ($1 \le a \le 1{,}000{,}000$). Note that the ratio between the largest and smallest radius of the disks is less than 4.

Output

Your program is to write to standard output. Print exactly one line which contains a real number z that represents the minimum horizontal distance OPT between the leftmost point and the rightmost point of the disks on any valid placement. The output z should be in the format that consists of its integer part, a decimal point, and its fractional part, and should satisfy the condition that $OPT - 10^{-5} < z < OPT + 10^{-5}$.

The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
4 4 2 7 6	34.99452

Sample Input 2	Output for the Sample Input 2
5 13 7 4 15 10	90.14124

Problem D
Go Latin
Time Limit: 0.5 Second

There are English words that you want to translate them into pseudo-Latin. To change an English word into pseudo-Latin word, you simply change the end of the English word like the following table.

English	pseudo-Latin
-a	-as
-i, -y	-ios
-l	-les
-n, -ne	-anes
-o	-os
-r	-res
-t	-tas
-u	-us
-v	-ves
-w	-was

If a word is not ended as it stated in the table, put '-us' at the end of the word. For example, a word "cup" is translated into "cupus" and a word "water" is translated into "wateres".

Write a program that translates English words into pseudo-Latin words.

Input
Your program is to read from standard input. The input starts with a line containing an integer, n ($1 \leq n \leq 20$), where n is the number of English words. In the following n lines, each line contains an English word. Words use only lowercase alphabet letters and each word contains at least 3 and at most 30 letters.

Output
Your program is to write to standard output. For an English word, print exactly one pseudo-Latin word in a line.

The following shows sample input and output for two test cases.

Sample Input 1
```
2
toy
engine
```

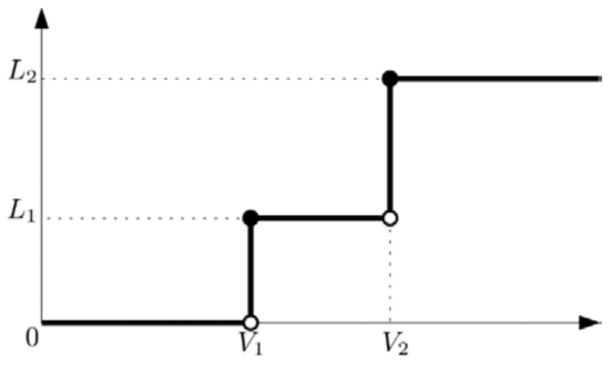
Output for the Sample Input 1
```
toios
engianes
```

Sample Input 2
```
3
cup
water
cappuccino
```

Output for the Sample Input 2
```
cupus
wateres
cappuccinos
```

Problem E
LED
Time Limit: 1.3 Seconds

A Light-Emitting Diode (LED) is a semiconductor light source, which emits light when an electric current of voltage higher than a threshhold is applied to its leads. ACM R&D recently reported that they have succesfully developed a new LED, namely, ACMOLED. An ACMOLED has a special behavior that the intensity of light emitted from it changes in two steps as the voltage of the electric current increases, as depicted in the graph below.



As shown, an ACMOLED is not activated in the voltage range from 0 to V_1, while it emits light with intensity $L_1 \geq 0$ when the voltage reaches the first threshold V_1 and light with intensity $L_2 \geq L_1$ when the voltage reaches the second threshold V_2. More specifically, if $F(v)$ is the function that maps voltage v to the intensity of light emitted from an ACMOLED, then for four real numbers L_1, L_2, V_1, and V_2 with $0 \leq L_1 \leq L_2$ and $0 < V_1 < V_2$, we have

$$F(v) = \begin{cases} 0 & \text{if } 0 \leq v < V_1 \\ L_1 & \text{if } V_1 \leq v < V_2 \\ L_2 & \text{if } v \geq V_2 \end{cases}.$$

The very issue now is that ACM R&D still does not know the exact values of two threshold voltage values V_1 and V_2 and the two intensity values L_1 and L_2 as well. Researchers in ACM R&D plan to estimate these four values for ACMOLEDs by repeated experiments.

Experiments are performed by applying current of a specific voltage and observing the intensity of light emitted from an ACMOLED. After n repeated experiments with different voltage values, obtained are the data of n tuples $(v_1, l_1), (v_2, l_2), \ldots, (v_n, l_n)$, where l_i is the observed intensity for voltage v_i. Due to the impreciseness of the observing device and other reasons, the experimental data are not accurate and may contain some error. Nonetheless, they want to find a best estimated intensity function $F(v)$ that minimizes the following error function:

$$\text{error}(F) = \max_{1 \leq i \leq n} |l_i - F(v_i)|$$

where $|x|$ denotes the absolute value of a real number x.

For a given data of n tuples, write a program that finds an estimated intensity function F that minimizes the above error function and outputs the value of $\text{error}(F)$.

Input

Your program is to read from standard input. The input starts with a line containing an integer n ($1 \leq n \leq 300{,}000$), where n is the number of tuples (v_i, l_i) in the experimental data. In the following n lines, each line contains two integers, which range inclusively from 0 to 10^9, representing v_i and l_i in each tuple (v_i, l_i) of the experimental data. Note that you may assume that there are no two tuples (v_i, l_i) and (v_j, l_j) in the input such that $1 \leq i < j \leq n$ and $v_i = v_j$.

Output

Your program is to write to standard output. Print exactly one line consisting of one real number, rounded to the first decimal place, which represents the minimum value of $\mathrm{error}(F)$.

The following shows sample input and output for two test cases.

Sample Input 1

```
5
0 0
2 1
3 5
6 7
7 11
```

Output for the Sample Input 1

```
1.0
```

Sample Input 2

```
10
5 9
8 9
0 0
23 18
26 18
2 0
3 0
13 9
18 9
21 18
```

Output for the Sample Input 2

```
0.0
```

Problem F
Parentheses
Time Limit: 1 Second

$$a + (b \times c)$$

The correspondence among the operators and the operands can be clarified using parentheses. In a C program, for example, the expression `a-b-c` can be clarified as `(a-b)-c` since the minus operator is left associative.

The parentheses can be used to override the default precedences and associativities of operators. For instance, in the expression `a-(b-c)`, the left associativity of the minus operator is overridden by the parentheses.

A novice C programmer Dennis has been stressed too much in remembering the operator precedences and associativities. Therefore, he made a new language namely ICPC (I can parenthesize C), in which the operator-operand correspondence should be clarified fully using parentheses; except for this, all the other features are same as C. For instance, one should write `(a-b)-c` instead of `a-b-c` and `a+(b*c)` rather than `a+b*c`.

However, the usage of the parentheses can be too much in some cases. For the expression `a-b-c`, it is enough to write

```
(a-b)-c
```

but one can write it as

$\texttt{(a-}\underline{\texttt{(}}\texttt{b}\underline{\texttt{)}}\texttt{)-c}$

or as

$\underline{\texttt{(}}\texttt{(a-b)-c}\underline{\texttt{)}}$

where the pairs of parentheses underlined are superfluous.

Dennis wants to convert the C expression into the ICPC expression, in which the pairs of parentheses should be used exactly as needed. You have to help Dennis. For simplicity, you can assume that the input C expression contains only five binary arithmetic operators (`+`, `-`, `*`, `/`, and `%`), left and right parentheses `(` and `)`, and single-lowercase alphabet operands. Given such a C expression, write a program to determine whether it is an ICPC expression or not.

If the expression is not an error in ICPC, then it should not be an error in C. Once it is not an error in C, the usage of parentheses should be checked to determine whether it is a proper expression in ICPC or not. If the expression is not properly parenthesized, i.e., the number of parentheses is not exact as needed, then it is considered improper.

Beware that some of the input C expressions may be erroneous originally. For instance, `a%/b` is an error since it requires one more operand between `%` and `/` to be valid. As another example, `a b + c` is also an error since it requires one more operator between `a` and `b`.

Input

Your program is to read from standard input. The input consists of a single line containing a C expression. The expression is a string of single-lowercase alphabets, special symbols including left and right parentheses and five binary arithmetic operators (`+`, `-`, `*`, `/`, and `%`), and spaces. The input line contains at least one operand. The length of the input line (the number of characters in it) is no more than 1000 including the spaces and the single newline character at the end.

Output

Your program is to write to standard output. Print exactly one line. The line should contain one of the following words: `error`, `proper`, and `improper`. Print `error` if the input C expression is erroneous. Once it is not an error, print `proper` or `improper` depending on the parenthesized status of the expression: print `proper` if it is parenthesized properly with the exact number of parentheses needed for ICPC, and print `improper` otherwise.

The following shows sample input and output for seven test cases.

Sample Input 1	Output for the Sample Input 1
`a + a`	`proper`

Sample Input 2	Output for the Sample Input 2
`(b+( a+c )) + b`	`proper`

Sample Input 3	Output for the Sample Input 3
`c + ((b) + a)`	`improper`

Sample Input 4	Output for the Sample Input 4
`c+(a%/b)`	`error`

Sample Input 5	Output for the Sample Input 5
`x + ((y + z)`	`error`

Sample Input 6	Output for the Sample Input 6
`a b + (c + b)`	`error`

Sample Input 7	Output for the Sample Input 7
`x + y + z`	`improper`

Problem G
Secret Code
Time Limit: 1 Second

Three secret agents A, B and C want to confirm if their secret codes are identical. To keep this secret, they did not set the meeting time. Instead, they decided to appear randomly at a café during the time interval $[0, S]$ of a day. Let t_A, t_B, t_C denote the arrival time of A, B, and C to the café, respectively. So, t_A, t_B, t_C are random numbers chosen uniformly from time interval $[0, S]$.

The code confirmation proceeds as follows. The agent who arrives earlier waits for the next agents to appear for a predetermined waiting time. If two agents encounter in the café, both check if their codes are identical. Then the agent who arrived earlier leaves the café immediately after the code confirmation. The second agent then waits till the third agent appears at the café. The agent leaves the café if the agent encounters the last agent in the agent's own waiting time.

The waiting times of three agents A, B, C have already been determined as w_A, w_B, w_C. It is crucial that each of the arrival times t_A, t_B, t_C is a real number between 0 and S, not necessarily an integer, whereas each of the waiting times w_A, w_B, w_C is a positive integer satisfying $0 < w_A + w_B,\ w_B + w_C,\ w_A + w_C < S$. We assume it takes no time in code confirmation task. The agent immediately leaves the café if the agent confirms the code with the agent arrived later. Let us explain this procedure using Figure G.1.

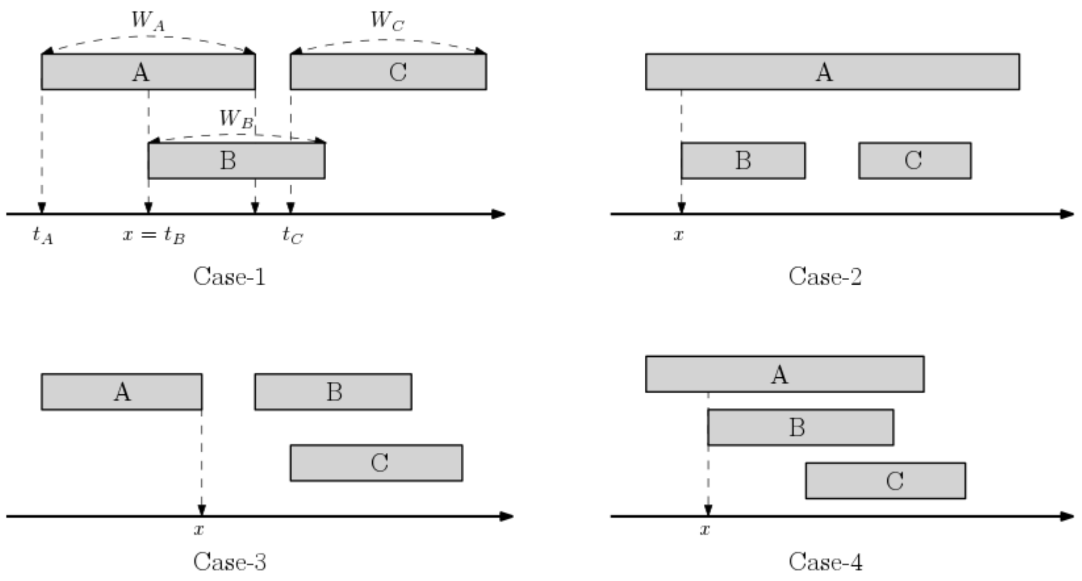


Figure G.1 Four cases for successful and unsuccessful confirmations.

Successful code confirmation needs at least two encounters among three agents. When the arrival order is A, B, C, both of Case-1 and Case-4 are examples for successful confirmation, but Case-2 and Case-3 are not. In Case-1, the agent A leaves the café at time $x = t_B$ immediately not waiting by time $t_A + W_A$. It is easy to see that Case-2 is an unsuccessful case. In Case-2, though the waiting time of A overlaps those of B and C, B cannot confirm the code with C since the agent A (who already confirmed the code with B) leaves the café at time x. So, there is no way to confirm the code between B and C. Note that A also leaves the café at time x in Case-3 and Case-4.

We know the probability of the successful code confirmation depends on four integers (S, w_A, w_B, w_C). We are given n different scenarios determined by four integers (S, w_A, w_B, w_C). We want to sort these n scenarios in terms of this confirmation probability.

Given n scenarios, write a program to print the scenario indices in the non-decreasing order of the probability.

Input

Your program is to read from standard input. The input starts with a line containing an integer n $(3 \le n \le 20)$, where n is the number of scenarios. In the following n lines, each line contains four positive integers S, w_A, w_B, w_C in this order, describing a scenario (S, w_A, w_B, w_C) where $0 < w_A + w_B, w_B + w_C, w_A + w_C < S \le 1{,}000$.

Output

Your program is to write to standard output. Print the indices of n scenarios in the non-decreasing order of the probability in one line. If scenarios i and j for $1 \le i < j \le n$ have the same probability, then print i before j.

The following shows sample input and output for two test cases.

Sample Input 1

```
3
100 12 13 14
110  8  9 15
200 23 30 40
```

Output for the Sample Input 1

```
2 1 3
```

Sample Input 2

```
6
201 15 16 16
375 30 32 27
900 75 73 67
203 16 17 16
373 31 32 27
895 73 75 66
```

Output for the Sample Input 2

```
1 2 3 6 5 4
```

Problem H
Simple Polygon
Time Limit: 2 Seconds

Let S be a set of n line segments. An endpoint of every line segment of S lies on the x-axis and the other lies above the x-axis in the plane. All endpoints are distinct, but the segments may intersect at their interiors. We want to draw a simple polygon P such that every line segment of S would be a part of the boundary of P, i.e., either an edge or a part of an edge of P. A polygon P is said to be *simple* if the angle at each vertex of P is not 180°, and no two edges meet except for the case where two adjacent edges meet at a vertex.

For example, Figure H.1 shows examples of line segments for which simple polygons can be drawn. In Figure, a black solid line segment represents a line segment in a given set S. When combined with red dotted line segments, a simple polygon is constructed. The left one in this figure is a simple poygon such that all endpoints of S become vertices of P. The right one shows a simple polygon only with 6 vertices; two endpoints are each contained in the interior of two edges of P. Note here that an endpoint of a line segment of S does not necessarily have to be a vertex of a simple polygon drawn for S.

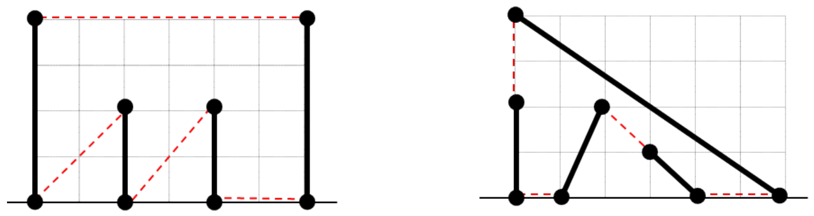


Figure H.1: Examples of line segments for which simple polygons can be drawn.

On the other hand, Figure H.2 shows examples of line segments for which any simple polygons cannot be drawn.

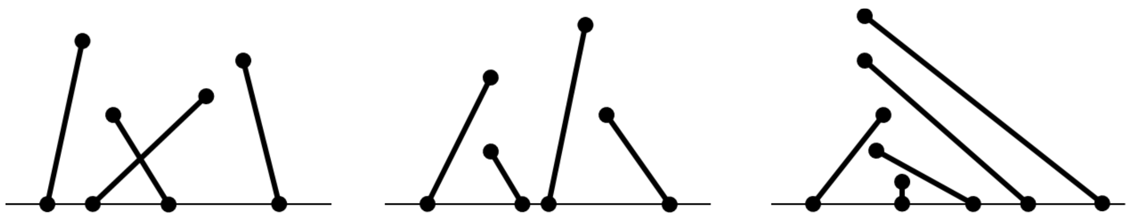


Figure X.2: Examples of line segments for which any simple polygons cannot be drawn.

There may be more than one simple polygon for a set of line segments. In that case, we want to find a simple polygon with the minimum boundary length. For example, the boundary length of the simple polygon shown in Figure H.3 is smaller than that of any other simple polygons for this set.

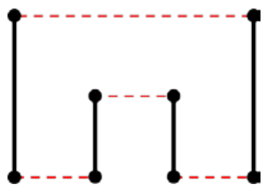


Figure H.3: A simple polygon with the minimum boundary length.

Given a set of line segments, write a program to determine whether there is a simple polygon for the set and if so, to find a simple polygon with the minimum boundary length.

Input

Your program is to read from standard input. The input starts with a line containing an integer, n ($3 \leq n \leq 20{,}000$), where n is the number of line segments. In the following n lines, each line contains three integers $x_1, x_2,$ and y_2 ($1 \leq x_1, x_2, y_2 \leq 10^6$) which represent the coordinates of the endpoints of a line segment, $(x_1, 0)$ and (x_2, y_2). Note that all endpoints of line segments are distinct, i.e., any two endpoints don't lie at the same position.

Output

Your program is to write to standard output. Print exactly one line. The line should contain a real number rounded to 1 decimal place which represents the boundary length of a simple polygon with the minimum boundary length if there is a simple polygon for a given set of line segments, otherwise, `-1`.

The following shows sample input and output for three test cases.

Sample Input 1	Output for the Sample Input 1

4
1 1 2
2 2 1
3 3 1
4 4 2
	12.0

Sample Input 2	Output for the Sample Input 2

4
1 1 2
2 3 2
5 4 1
7 1 4
	19.3

Sample Input 3	Output for the Sample Input 3

4
5 6 6
4 3 2
1 3 4
9 7 3
	-1

Problem I
Square Root
Time Limit: 3 Seconds

The *square* T^2 of a tree T is defined as a simple undirected graph with the same vertex set as T and the edge set that is augmented in such a way that two vertices of T^2 are adjacent if and only if there exists a path of length at most two in T joining them. That is, its vertex set is equal to that of T and its edge set is equal to $\{(u, v) : d_T(u, v) \leq 2\}$, where $d_T(u, v)$ denotes the distance between u and v in T. Figure X.1 shows a tree and its square.

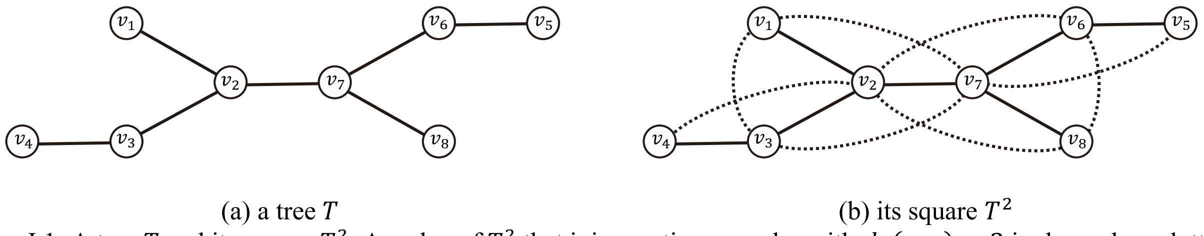


Figure I.1: A tree T and its square T^2. An edge of T^2 that joins vertices u and v with $d_T(u, v) = 2$ is shown by a dotted curve.

If a graph G is the square of some tree T, i.e. $G = T^2$, then T is said to be a *square root* of G. For a given tree T, computing the square T^2 is trivial; for a given graph G, however, deciding if there exists a tree T such that $T^2 = G$ is not trivial. Your job is to write an efficient running program for deciding whether or not there exists a tree that is a square root of an input graph G.

Input
Your program is to read from standard input. The first line contains two positive integers n and m, respectively, representing the numbers of vertices and edges of the input graph G, where $2 \leq n \leq 100{,}000$ and $m \leq 1{,}000{,}000$. It is followed by m lines, each contains two positive integers u and v representing an edge between the vertices u and v of G. It is assumed that G is a simple undirected graph whose vertices are indexed from 1 to n.

Output
Your program is to write to standard output. The first line must contain an integer indicating whether there exists a tree that is a square root of the input graph. If yes, the integer must be `1`; otherwise `-1`. When and only when the first line is `1`, it must be followed by the description of an arbitrary tree that is a square root of the input graph. A tree is described by a single line containing an integer n, representing the number of vertices, followed by $n - 1$ lines, each contains two positive integers u and v representing an edge between the vertices u and v of the tree.

The following shows sample input and output for four test cases.

Sample Input 1

```
8 15
1 2
1 3
1 7
2 3
2 4
2 6
2 7
2 8
3 4
3 7
5 6
5 7
6 7
6 8
7 8
```

Output for the Sample Input 1

```
1
8
1 2
2 3
3 4
7 2
5 6
6 7
7 8
```

Sample Input 2

```
8 14
1 2
1 3
1 7
2 3
2 4
2 6
2 7
2 8
3 4
3 7
5 6
5 7
6 7
6 8
```

Output for the Sample Input 2

```
-1
```

Sample Input 3

```
5 7
1 2
2 3
3 4
4 5
5 3
4 2
3 1
```

Output for the Sample Input 3

```
1
5
1 2
2 3
3 4
4 5
```

Sample Input 4

```
4 6
1 2
2 3
3 4
4 1
4 2
3 1
```

Output for the Sample Input 4

```
1
4
2 1
3 1
4 1
```

Problem J
Starwars
Time Limit: 1 Second

In the future, humans have colonized the universe and are at war with an alien race. In space, there are many solar systems and we can separate these solar systems into two categories: 1) human-controlled solar systems and 2) non-human-controlled solar systems. Humans have also installed their military bases in several solar systems, where some of these solar systems are human-controlled and the others are non-human-controlled. Solar systems are very far apart, so the only way to travel between two systems is by wormhole. However, not all pairs of solar systems are connected by a wormhole and wormholes only go in one direction. When a human ship, starting at a human-controlled solar system, wants to travel to a military base in some solar system, it has to receive a series of certificates to prove that it is a human ship instead of an alien spy ship. A ship travels between two solar systems via a wormhole and receives the corresponding certificate from the wormhole. There may be many different wormholes between two solar systems, each with different certificates. There can also be many wormholes from a solar system that lead to all different solar systems with the same certificate. A solar system can even have wormholes that travel to itself for time travel. When a ship arrives at a military base, its collected certificates are examined in collected order to confirm whether or not the ship originated from a human-controlled solar system. However, humans are lazy and only check if the sequence of certificates matches a route from any human-controlled system to any military base.

The aliens immediately realize that humans do not check if there is a route from a non-human-controlled solar system to a military base that has the same sequence of certificates due to their laziness. Alien spies want to sneak into human military bases. Therefore, they try to find a route to a military base that produces the same sequence of certificates that a ship travelling from a human-controlled system to a military base would have. However, aliens cannot start from human-controlled solar systems but they can visit human-controlled solar systems after their initial departure.

As an alien spy, your job is to determine if there is a route from a non-human-controlled system to a military base B_i such that the route produces a sequence of certificates that would allow you to pass as a human; in other words, this sequence of certificates is same as a sequence of certificates that a human can collect when travelling from a human-controlled system to a military base B_j. Note that B_i and B_j can be different.

The universe is made up of N solar systems. Some of them are specially marked as human-controlled and some are specially marked as having a military base (solar systems can be both human controlled and have a military base, one or the other, or neither). Some of the solar systems are connected by one-way wormholes. When a ship travels through a wormhole, it receives a special certificate (there are many different types of certificates).

Input
Your program is to read from standard input. The first line of the input contains five integers N, W, C, H, M separated by spaces. N $(1 \le N \le 1{,}000)$ is the number of solar systems. The solar systems are labeled with distinct integers from 0 to $N-1$. W $(1 \le W \le 8000)$ is the number of wormholes. C $(1 \le C \le 20)$ is the number of distinct certificates. The next line contains H $(1 \le H \le N)$ integers, separated by spaces, which correspond to the human controlled solar systems. The next line contains M $(1 \le M \le N)$ integers, separated by spaces, which mark the solar systems with military bases. The remaining W lines contain three integers per line s, c, t separated by spaces, corresponding to a wormhole. For each wormhole, s is the source solar system for the wormhole, c is the certificate the wormhole awards, and t is the target solar system of the wormhole with $0 \le s, t \le N-1$ and $1 \le c \le C$. Note that wormholes can have $s = t$.

Output

Your program is to write to standard output. Print `YES` if there is a way for an alien to sneak into a military base and `NO` if there is no way for an alien to sneak into a military base.

The following shows sample input and output for three test cases.

Sample Input 1

```
4 6 2 1 2
0
1 3
0 1 0
0 2 1
0 1 2
1 2 0
2 2 1
2 1 3
```

Output for the Sample Input 1

```
YES
```

Sample Input 2

```
5 6 2 1 1
0
4
0 1 1
1 1 2
2 2 3
2 2 1
3 1 4
4 1 4
```

Output for the Sample Input 2

```
NO
```

Sample Input 3

```
5 4 2 2 2
1 3
2 4
0 2 1
1 1 2
3 2 3
3 1 4
```

Output for the Sample Input 3

```
YES
```

Problem K
TV Show Game
Time Limit: 1 Second

Mr. Dajuda, who is famous for a TV show program, occasionally suggests an interesting game for the audience and gives them some gifts as a prize. The game he suggested this week can be explained as follows.

The $k(> 3)$ lamps on the stage are all turned off at the beginning of the game. For convenience, lamps are numbered from 1 to k. Each lamp has a color, either red or blue. However, the color of a lamp cannot be identified until it is turned on. Game participants are asked to select three lamps at random and to guess the colors of them. Then each participant submits a paper on which the predicted colors of selected lamps are recorded to Mr. Dajuda, the game host. When all the lamps are turned on, each participant checks how many predicted colors match the actual colors of the lamps. If two or more colors match, he/she will receive a nice gift as a prize.

Mr. Dajuda prepared a special gift today. That is, after reviewing all the papers received from the game participants he tries to adjust the color of each lamp so that every participant can receive a prize if possible.

Given information about the predicted colors as explained above, write a program that determines whether the colors of all the lamps can be adjusted so that all the participants can receive prizes.

Input
Your program is to read from standard input. The input starts with a line containing two integers, k and n $(3 < k \leq 5{,}000,\ 1 \leq n \leq 10{,}000)$, where k is the number of lamps and n the number of game participants. Each of the following n lines contains three pairs of (l, c), where l is the lamp number he/she selected and c is a character, either `B` for blue or `R` for red, which denotes the color he/she guessed for the lamp. There is a blank between l and c and each pair of (l, c) is separated by a blank as well as shown in following samples.

Output
Your program is to write to standard output. If it is possible that all the colors can be adjusted so that every participant can receive a prize, print k characters in a line. The i^{th} character, either `B` for blue or `R` for red represents the color of the i^{th} lamp. If impossible, print `-1`. If there are more than one answer, you can print out any of them.

The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
7 5 3 R 5 R 6 B 1 B 2 B 3 R 4 R 5 B 6 B 5 R 6 B 7 B 1 R 2 R 4 R	BRRRBBB

Sample Input 2

```
5 6
1 B 3 R 4 B
2 B 3 R 4 R
1 B 2 R 3 R
3 R 4 B 5 B
3 B 4 B 5 B
1 R 2 R 4 R
```

Output for the Sample Input 2

```
-1
```

Problem L
Working Plan
Time Limit: 2 Seconds

ICPC manager plans a new project which is to be carried out for n days. In this project, m persons numbered from 1 to m are supposed to work. Each day j $(1 \le j \le n)$ requires d_j persons, and each person i $(1 \le i \le m)$ wants to work w_i days.

To increase the efficiency in performing the project, the following two conditions should be satisfied:

(1) each person works for only consecutive w days when he/she works, and
(2) each person can work again after he/she has a rest for at least h days.

ICPC manager wants to find a working plan to assign the working days for all persons such that the number of working days of each person i $(1 \le i \le m)$ is equal to w_i and the number of persons who work for each day j $(1 \le j \le n)$ is equal to d_j, and above two conditions are also satisfied.

For example, assume the project is carried out for $n = 9$ days, and $m = 4$ persons participate in the project. Let $w = 2$ and $h = 1$. Also, assume $(w_1, w_2, w_3, w_4) = (4, 4, 6, 2)$ and $(d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9) = (1, 3, 2, 1, 2, 1, 1, 3, 2)$. The table below shows a feasible solution where the i-th row corresponds to person i, and the j-th column corresponds to day j. If person i works or has a rest in day j, the value of the table element with row i and column j is 1 or 0, respectively.

		Day								
		1	2	3	4	5	6	7	8	9
Person	1	1	1	0	0	0	0	0	1	1
	2	0	1	1	0	0	0	1	1	0
	3	0	1	1	0	1	1	0	1	1
	4	0	0	0	1	1	0	0	0	0

Given m, n, w, h, w_i $(1 \le i \le m)$ which is a multiple of w, and d_j $(1 \le j \le n)$, write a program to find a feasible solution as a working plan.

Input
Your program is to read from standard input. The input starts with a line containing four integers, m, n, w, h $(1 \le m \le 2{,}000,\ 1 \le n \le 2{,}000,\ 1 \le w, h \le n)$. The following line contains m integers where the i-th $(1 \le i \le m)$ integer represents w_i $(1 \le w_i \le n)$ which is a multiple of w. The next line contains n integers where the j-th $(1 \le j \le n)$ integer represents d_j $(0 \le d_j \le m)$.

Output
Your program is to write to standard output. If there is a feasible working plan, print `1` in the first line followed by m lines, each i-th $(1 \le i \le m)$ line should contain w_i/w integers. These integers form an increasing sequence of first days that person i works in the feasible plan. If there is no feasible working plan, print only `-1` in the first line. The first sample below corresponds to the example given in the table above.

The following shows sample input and output for two test cases.

Sample Input 1

```
4 9 2 1
4 4 6 2
1 3 2 1 2 1 1 3 2
```

Output for the Sample Input 1

```
1
1 8
2 7
2 5 8
4
```

Sample Input 2

```
4 7 2 2
4 4 4 2
1 3 2 1 3 3 1
```

Output for the Sample Input 2

```
-1
```