Problem A. Aqua

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

In the famous ICPC race, n runners will participate. The course is m kilometers long, and for safety, it is divided into m ranges. Each range is one kilometer long, and range i ($1 \le i \le m$) is the interval $(i-1, i)$, which is the section between $(i-1)$ and i kilometer marks from the starting point. We will ignore the case where the distance between the starting point and a runner is an integer number of kilometers.

As the weather is quite hot, the organizers would like to put enough water. They will maintain a certain number of water bottles in each range. When a runner takes one bottle, they will put another immediately. They have found that the optimal number of water bottles in a range could be obtained by calculating the maximum number of runners in that range during the race. Based on the previous records of each runner, they have estimated how many seconds they will spend in each range.

Consider the following example. There are three runners, and the length of the course is six kilometers. The table shows the amount of time runners will spend in each range (in seconds).

Runner	Range 1	Range 2	Range 3	Range 4	Range 5	Range 6
1	350s	360s	370s	380s	390s	400s
2	240s	240s	240s	240s	240s	240s
3	480s	480s	520s	600s	600s	600s

Now we will check the number of runners in each range during the race. Intentionally, the table below is not complete. When you fill the whole table and compute the maximum number of runners for each range, you can see that you need to put three bottles of water in Range 1, two in Range 2 and Range 3, and one in Range 4, Range 5, and Range 6. Note that at 480 seconds, Runner 2 leaves Range 2 and Runner 3 arrives at Range 2, both of which will be ignored as their distance from the starting point is an integer. At 480 seconds, no runner is in Range 1 and in Range 3, while Runner 1 is in Range 2. Then, for example, at 481 seconds, Runner 1 and Runner 3 will be in Range 2.

Time Elapsed	Range 1	Range 2	Range 3	Range 4	Range 5	Range 6
(0s,240s)	3	0	0	0	0	0
(240s,350s)	2	1	0	0	0	0
(350s,480s)	1	2	0	0	0	0
(480s,710s)	0	2	1	0	0	0
(710s,720s)	0	1	2	0	0	0
...	...	...	...	...	...	...

Given the number of runners, the length of the course, and the amount of time each runner will spend in each range, write a program to compute the number of bottles to be put in each range.

Input

The input starts with a line containing two integers, n and m ($1 \le n \le 100$; $1 \le m \le 300$), where n is the number of runners and m is the length of the course. In the following n lines, the i-th line contains m positive integers that represent the amount of time Runner i will spend in each range. More precisely, the j-th number on the line is the time Runner i will spend in Range j. No runner will spend more than 10 000 seconds in any range.

Output

Print exactly one line. The line should contain the numbers of bottles in each range from Range 1 to Range m.

Examples

standard input	*standard output*
3 6	3 2 2 1 1 1
350 360 370 380 390 400	
240 240 240 240 240 240	
480 480 520 600 600 600	
4 5	4 4 4 4 4
1 1 1 1 1	
1 1 1 1 1	
1 1 1 1 1	
1 1 1 1 1	
3 5	3 1 1 1 1
1 1 1 1 1	
5 5 5 5 5	
25 25 25 25 25	

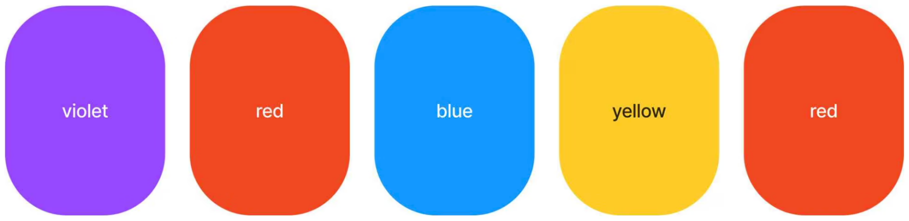
Problem B. Bottom and Top Colors

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 second
Memory limit:	1024 mebibytes

The magician B has n cards in a row on a desk. Each card has two sides with colors. The top side of a card is the side facing upwards. The bottom side of a card is the side facing downwards. Each side of a card has one color. We want to flip some cards (possibly none or all) so that the number of distinct colors on the top sides is the maximum possible.

Consider the following example. We are given 5 cards in a row on a desk. The colors of the top sides of the cards are violet, red, violet, violet, and red from the left to the right as shown in the following figure. The colors of the bottom sides of the cards are red, violet, blue, yellow, and red from the left to the right.

If we flip a card, then the top side and the bottom side of the card are exchanged. If we flip the 3-rd and the 4-th cards from the left, then the colors of the cards on the top sides become the following.



The number of distinct colors on the top sides becomes 4, which is the maximum for the example.

Given n cards placed in a row on a desk and the colors on the sides of cards, write a program to output the maximum number of distinct colors on the top sides.

Input

The input starts with a line containing an integer n ($1 \le n \le 2 \cdot 10^5$), where n is the number of cards. The cards are numbered from 1 to n. Then follow two lines. The first of them contains the colors on the top sides of the cards. The second line contains the colors on the bottom sides of the cards. Each color is represented by a nonnegative integer not exceeding 10^6.

Output

Print exactly one line. The line should contain the maximum number of distinct colors on the top sides after we flip some of the cards (possibly none or all).

Examples

standard input	*standard output*

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

2
3 5
5 1
	2

3
0 1 0
1 0 2
	3

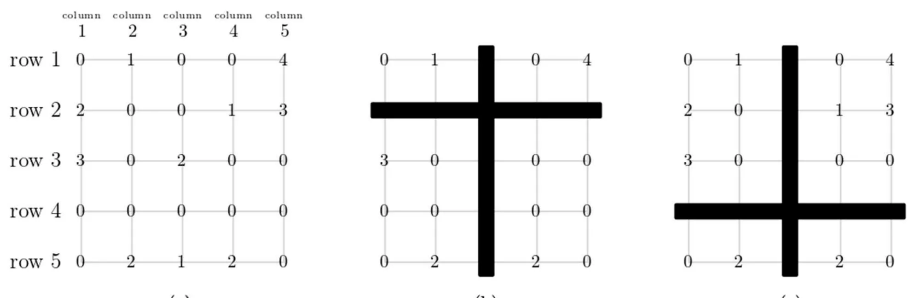
Problem C. Colorful Quadrants

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

You are given an $n \times n$ grid, and some of the grid points are colored by one of the k colors. The color of a point is represented by an integer from 0 to k, where 0 represents the uncolored case. Note that multiple points may be colored the same. The rows and columns of the grid are denoted by integers from 1 to n, and a point located at row i and column j is denoted by (i, j).

For an uncolored point (i, j) that satisfies $1 < i < n$ and $1 < j < n$, we define four sub-grids by removing row i and column j from the grid. Each of the four sub-grids is called NW (northwest), NE (northeast), SW (southwest), and SE (southeast) based on the position relative to (i, j). We say that (i, j) has colorful quadrants if we can select one point from each of the four sub-grids so that the chosen four points are all of different colors.

See Figure (a) as a 5×5 grid example. The point $(2, 3)$ has colorful quadrants because NW has color 1, NE has color 4, SW has color 3, and SE has color 2, as shown in Figure (b). However, the point $(4, 3)$ does not have colorful quadrants because both SW and SE have color 2 only, as shown in Figure (c).



(a) (b) (c)

Given an $n \times n$ grid containing at least four grid points colored in different colors, write a program to count the number of uncolored points that have colorful quadrants.

Input

The input starts with a line containing two integers, n and k ($3 \le n \le 2000$; $4 \le k \le 1000$), where n is the number of rows and columns of the grid and k is the number of colors. In the following n lines, the i-th line contains n integers that represent the colors of the points (i, j) for $1 \le j \le n$. The integer c that represents the color of a point is in the range $0 \le c \le k$, where 0 means that the point is uncolored.

Output

Print exactly one line. The line should contain the number of uncolored points that have colorful quadrants.

Examples

standard input	*standard output*
5 4	1
0 1 0 0 4	
2 0 0 1 3	
3 0 2 0 0	
0 0 0 0 0	
0 2 1 2 0	
3 4	0
1 2 3	
4 1 2	
3 4 1	
4 8	0
0 1 2 0	
8 0 0 3	
7 0 0 4	
0 6 5 0	

Problem D. Drawing Optimal Ladder

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

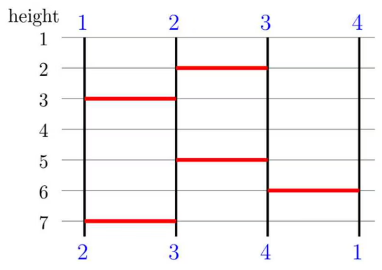
Ladder game is a popular game in Korea, as well as China and Japan. Wikipedia says that it is known in Korean as Sadaritagi (literally "ladder climbing"), in Japanese as Amidakuji ("Amida lottery"), and in Chinese as Guijiaotu (literally "ghost leg diagram").

The diagram where the game is played consists of n vertical lines with horizontal line segments connecting two adjacent vertical lines. The horizontal line segments are called legs. Each vertical line has a starting point at the top and an end point at the bottom. The basic rule of this game is simple as follows:

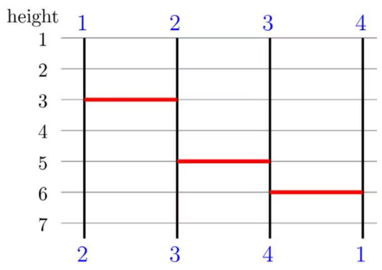
Start from the starting point of each vertical line and move downward along the vertical line. Whenever you encounter a leg, move along the leg to the adjacent vertical line, and continue moving downwards along that line. Stop when you reach the bottom of some vertical line.

The vertical lines are numbered from 1 to n from left to right. When we start moving from the top of line i, we move to the end of some vertical line and write the number i at the bottom of that line. The result of the game is the numbers written under the lines from 1 to n from left to right.

It is well known that the number at the bottom of each line is unique, and the game result is a permutation of $[1, 2, \ldots, n]$. For example, given a diagram with 4 vertical lines and 5 legs shown below, the game result is $[2, 3, 4, 1]$ from left to right.



However, some legs are redundant, meaning that the same result $[2, 3, 4, 1]$ can be achieved with fewer legs; as in the figure below, one can obtain the same result only with three legs excluding the topmost and bottommost ones. We want to determine the minimum number of horizontal line segments (legs) needed to achieve the same result. Note that it is possible to draw new legs than the given ones if necessary.



You are given q queries, where each query either adds a new leg or deletes an existing one. Write a program to output the minimum number of legs required to achieve the same game result of the ladder structure obtained after the query is processed. Note that each query is cumulative, meaning each subsequent query is applied to the ladder structure resulting from previous queries.

Input

The input starts with a line containing three integers: n, the number of vertical lines, m, the initial number of legs, and q, the number of queries ($2 \leq n \leq 10^5$; $1 \leq m, q \leq 10^5$).

In the following m lines, each line contains two positive integers h and a, representing a leg at height h connecting the a-th and $(a+1)$-th vertical lines $(1 \le a \le n-1)$. The vertical lines are ordered from left to right, and the height is measured **from top to bottom** starting with 1. Each height is no more than 10^9. You can assume that all vertical lines start above and end below all the given heights.

The next q lines contain the query information. Each query is in the form of either "`A` h a" or "`D` h a", where $1 \le h \le 10^9$ and $1 \le a \le n-1$.

- "`A` h a": add a leg at height h between the a-th and $(a+1)$-st vertical lines.
- "`D` h a": delete the leg at height h between the a-th and $(a+1)$-st vertical lines.

You can assume that there are no contradictory operations, that is, existing legs will not be added, and non-existing legs will not be deleted. Also, you can assume that no two legs are positioned such that they share the endpoint of the same height and the same vertical line.

Output

The output consists of q lines. Each line contains the minimum number of legs required to achieve the same result for a query in the input order.

Examples

standard input	*standard output*

`4 4 3`
`3 1`
`2 2`
`5 2`
`6 3`
`A 7 1`
`A 4 3`
`D 3 1`
	`3`
	`6`
	`3`

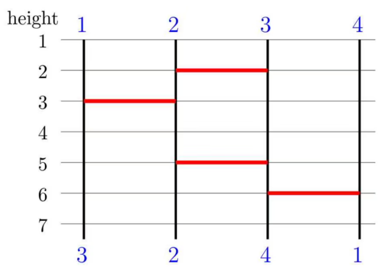
`4 5 5`
`3 1`
`2 2`
`5 2`
`6 3`
`7 1`
`D 6 3`
`D 7 1`
`D 5 2`
`D 3 1`
`D 2 2`
	`2`
	`3`
	`2`
	`1`
	`0`

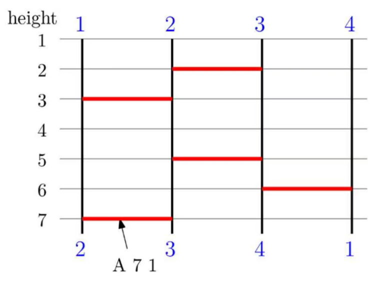
Note

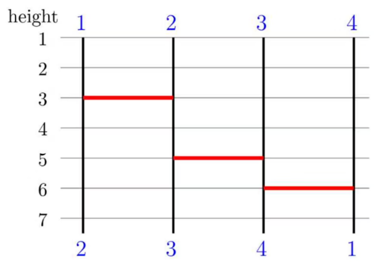
The sample input 1 gives the initial ladder structure on the left picture below. The game result is $[3, 2, 4, 1]$.

After applying the first query "`A 7 1`" in the middle picture below, the ladder structure is changed, then the game result becomes $[2, 3, 4, 1]$.

Among the five legs, only three legs (without topmost and bottommost legs) are enough to achieve the same game result $[2, 3, 4, 1]$ as shown in the right picture below, so the answer for the first query is 3.



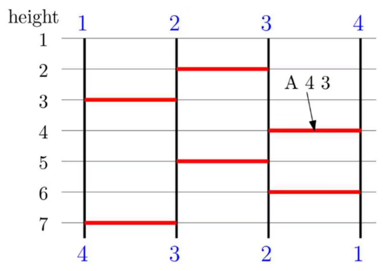


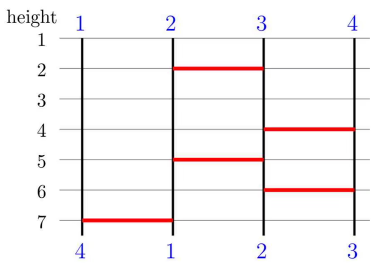


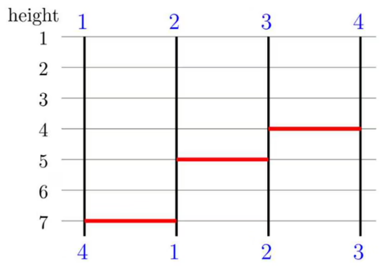
After processing the second query "`A 4 3`", the ladder structure is changed as shown on the left picture below. The number of legs cannot be further reduced. The answer for the second query is 6.

After applying the third query "`D 3 1`", the ladder structure is changed as shown on the middle picture below.

The ladder structure with three legs as shown on the right picture below guarantees the same game result, so the answer for the third query is 3.



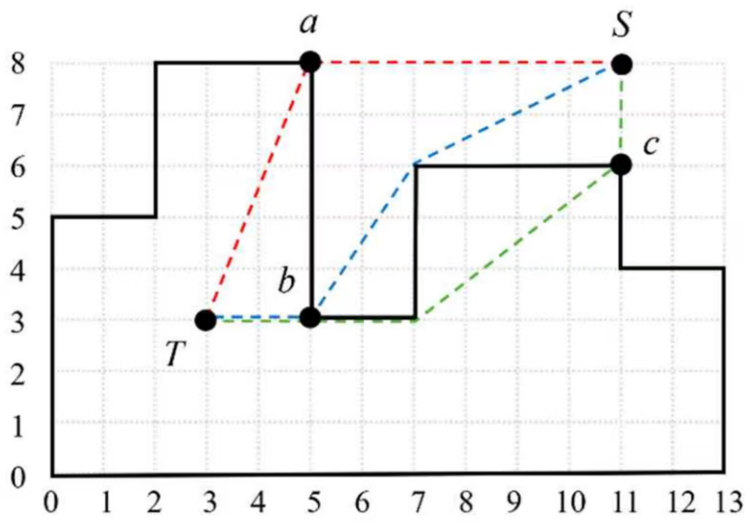




Problem E. Expedition For Treasures

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

The Mausoleum of King Geo III is a huge stone structure in the shape of a histogram. A histogram is a simple rectilinear polygon whose boundary consists of two chains: an upper chain that is monotone with respect to the horizontal axis, and a lower chain that is a horizontal line segment, called the base segment. On the figure below, a mausoleum and some paths between S and T are depicted.



It is rumored that a hidden treasure lies somewhere within this mausoleum. Metry, a renowned treasure hunter, has uncovered the treasure's location at point T. Metry's plan is to break through the mausoleum's walls, enter, and retrieve the treasure. She will start at a specific location S outside the mausoleum. Using her equipment, Metry can drill through only one point, which corresponds to a vertex on the boundary of the mausoleum. Since the time required to drill through the walls is the same at all vertices, the key to minimizing the time spent is to find the shortest path from S to T.

Consider the figure above. It illustrates a mausoleum along with several possible paths from S to T, where the vertices are pierced only once. The path through vertex a has a total length of $11.385165 = 6 + \sqrt{29}$, the path through vertex b has a length of $10.077687 = \sqrt{20} + \sqrt{13} + 2$, and the path through vertex c has a length of $11.0 = 2 + \sqrt{25} + 4$. Among these, the shortest path is through vertex b.

Given the boundary of the mausoleum and the positions of S and T, write a program to find the length of the shortest path from S to T with a single vertex piercing.

Input

The input starts with a line containing an integer n, the number of vertices of a histogram representing the mausoleum ($4 \le n \le 10^5$; n is even). In the second line, n integers $v_1, v_2, \ldots, v_n$ are given which represent, in alternating order, the x-coordinates of the vertical edges and the y-coordinates of the horizontal edges ($v_1 = v_n = 0$; $1 \le v_i \le 10^6$ for $1 < i < n$). The vertical and horizontal edges alternate as you traverse the upper chain of the histogram, from the left end to the right end of the base segment. The length of each edge is at least 1, and the x-coordinates are given in strictly increasing order. The last line contains four integers s_x, s_y, t_x, and t_y ($-10^6 \le s_x, s_y \le 2 \cdot 10^6$; $0 < t_x, t_y < 10^6$), where (s_x, s_y) and (t_x, t_y) correspond to the points S and T, respectively. Notice that S is a point outside the histogram and T is a point inside the histogram. Neither S nor T lies on the boundary.

Output

Print exactly one line. The line should contain exactly one real value which is the length of the shortest path between S and T. Your output q should be in the format that consists of its integer part, a decimal point, and its fractional part, and will be decided to be correct if it holds that $a - 10^{-3} < q < a + 10^{-3}$, where a denotes the jury's answer. The Euclidean distance between two points $p = (x_1, y_1)$ and $q = (x_2, y_2)$ is $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Examples

standard input	*standard output*

12
0 5 2 8 5 3 7 6 11 4 13 0
11 8 3 3
	10.077687230463569

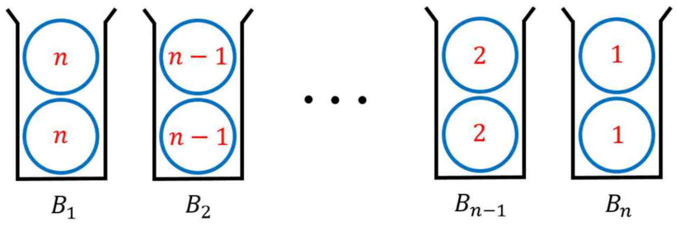
8
0 7 2 2 5 7 7 0
-2 4 6 4
	11.767828935632370

4
0 5 8 0
8 6 4 2
	6.000000000000000

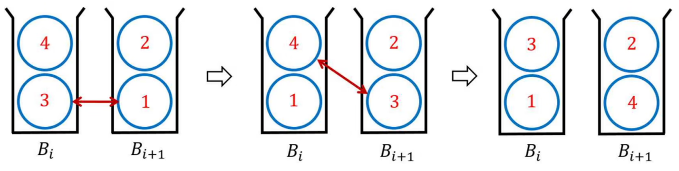
Problem F. Fix The Balls Order

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 mebibytes

There are n bins arranged in a row and $2n$ balls on the ground. The balls are labeled by integers: for each i from 1 to n, there are exactly two balls labeled by i. The bins are denoted as $B_1, \ldots, B_n$ from left to right. Each bin can contain at most two balls. Initially, bin B_i contains two balls with the number $n+1-i$ on them. See the figure below for the initial configuration of bins.



You can take any two balls from any two adjacent bins and swap them. This constitutes one swap operation. Note that a bin is not a stack: for each swap, you can take any of the two balls from B_i and any of the two balls from B_{i+1}. See the figure below for two examples of a swap operation.



Through these swap operations, you should sort the balls: as a result of the sorting, for each i from 1 to n, bin B_i must contain both balls labeled by i. Additionally, the total number of swap operations should be no more than $0.7n^2$.

Given n bins and $2n$ balls, write a program to find a sorting method for the balls such that the total number of swap operations is no more than $0.7n^2$.

Input

The input consists of exactly one line. The line contains an integer n ($3 \leq n \leq 100$), representing that there are n bins and $2n$ balls.

Output

Let S be the total number of swap operations in your sorting method for the input. Print exactly $S+1$ lines. On the first line, print the integer S ($0 \leq S \leq 0.7n^2$). On each of the following S lines, print three integers j, a, and b, representing one swap operation between the ball labeled by a in the bin B_j and the ball labeled by b in B_{j+1} ($1 \leq j \leq n-1$; $1 \leq a, b \leq n$). Print the swap operations in the order they should be applied.

Example

standard input	*standard output*

3	5
	1 3 2
	2 3 1
	1 3 1
	2 3 1
	1 2 1

Note

Note that the answer is not unique. For example, the sequence

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

is a correct solution for the sample as well.

Problem G. Good Old Palindromic Task

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

A string is called a palindrome if it is read the same forward and backward. Palindromes are useful factors for measuring the complexity of strings by their asymmetry. The asymmetry of a string S of length n can be measured by its palindromic length, $PL(S)$, which is the minimum number of palindrome substrings into which S can be partitioned. More precisely, $PL(S)$ is the minimum number t $(1 \leq t \leq n)$ such that there exist palindrome substrings $S_1, S_2, \ldots, S_t$ whose concatenation $S_1 S_2 \ldots S_t$ becomes S. To make it easier to distinguish, we denote a partition of S into $S_1, S_2, \ldots, S_t$ as $S_1 \mid S_2 \mid \ldots \mid S_t$.

For example, a string $S =$ `abaaca` can be partitioned into two palindrome substrings as `aba` | `aca`, and is not a palindrome itself, so $PL($`abaaca`$) = 2$. A string `acaba` cannot be partitioned into two palindrome substrings, but it can be partitioned into three palindrome substrings, $S =$ `aca` | `b` | `a` or $S =$ `a` | `c` | `aba`, so $PL($`acaba`$) = 3$. For $S =$ `radar`, $PL(S) = 1$ because S is a palindrome. $PL(S) = 5$ for $S =$ `abcde`.

Given a non-empty string S of English lowercase letters, write a program to output $PL(S)$.

Input

The input starts with a line containing a positive integer n $(1 \leq n \leq 10^5)$, where n is the number of letters of a string. The next line contains a string of n English lowercase letters. Note that the string contains no space between the letters.

Output

Print exactly one line. The line should contain a positive integer which is the palindromic length $PL(S)$ of the input string S.

Examples

standard input	*standard output*
6 abaaca	2
5 acaba	3
5 abcde	5
5 radar	1

Problem H. Hazardous Points

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	1024 mebibytes

In the kingdom of CPIC (Committee for Public Infrastructure Conservation), there are n villages numbered from 1 to n and connected by a network of $n-1$ two-directional roads forming a tree structure. Each road connects two villages and has a positive length. Specifically, the i-th road connects village $i+1$ with village p_i ($1 \le p_i \le i$) and has a length of ℓ_i. Due to treacherous terrains and past incidents, some points along these roads are identified as hazardous.

On the i-th road, there are k_i hazardous points located at specific distances $x_{i,1}, x_{i,2}, \ldots, x_{i,k_i}$ from village p_i, satisfying $0 < x_{i,1} < x_{i,2} < \ldots < x_{i,k_i} < \ell_i$. These distances are integers, indicating positions along the road.

The newly established CPIC Safety Committee aims to enhance traveler safety by deploying a protective measure. They can select any two points on the roads, including villages, and secure the shortest path between them. The path can cover all hazardous points located exactly on it, including its endpoints, and its length must not exceed a given length w.

Given the road network, the positions of the hazardous points, and the maximum allowable path length w, write a program to determine the maximum number of hazardous points that can be covered by optimally choosing the two points and securing the shortest path between them with length not greater than w.

Input

The input starts with a line containing two integers, n and w ($2 \le n \le 250\,000$; $1 \le w \le 10^{18}$), where n is the number of villages and w is the maximum allowable length of the secured path. In the following $n-1$ lines, the i-th line, which provides information about the i-th road, starts with three integers p_i, ℓ_i, and k_i ($1 \le p_i \le i$; $1 \le \ell_i \le 10^{12}$; $k_i \ge 0$), where p_i is the village connected to village $i+1$ by the road, ℓ_i is the length of the road, and k_i is the number of hazardous points on the road. If k_i is positive, the line is followed by k_i integers $x_{i,1}, x_{i,2}, \ldots, x_{i,k_i}$ ($0 < x_{i,1} < x_{i,2} < \ldots < x_{i,k_i} < \ell_i$), representing the distances from village p_i to each hazardous point along the road. The total number of hazardous points $k_1 + k_2 + \ldots + k_{n-1}$ does not exceed 10^6.

Output

Print exactly one line. The line should contain the maximum number of hazardous points that can be covered by a shortest path of length w or less between any two points on the roads.

Examples

standard input	*standard output*
`4 2`	`2`
`1 2 1 1`	
`1 610 2 1 100`	
`3 2001 0`	
`2 2`	`1`
`1 2 1 1`	
`8 6`	`4`
`1 2 1 1`	
`1 3 2 1 2`	
`2 1 0`	
`3 4 1 2`	
`2 3 1 1`	
`1 4 1 3`	
`3 4 1 1`	

Problem I. Integer Points Covered By Squares

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

On the plane, there are n points whose y-coordinates are either -9999, 0, or 9999. Let P be the set of these n points. Your task is to cover all the points in P by the minimum possible number of squares: each square has a side length of 10 000 and sides parallel to coordinate axes. Each square covers all points inside and on the boundary of the square.

Input

The input starts with a line consisting of a single integer n ($1 \le n \le 300\,000$), representing the number of input points in P. In each of the following n lines, there are two integers x and y, representing the x- and y-coordinates of a point in P, respectively ($-10^9 \le x \le 10^9$ and $y \in \{-9999, 0, 9999\}$). You may assume that all the n input points are distinct.

Output

Print exactly one line. The line should consist of a single integer that represents the minimum possible number t such that there exist t squares with a side length of 10 000 and sides parallel to coordinate axes whose union covers all the input points in P.

Examples

standard input	*standard output*

5
0 9999
0 0
0 -9999
200 0
10000 9999
	2

5
10 -9999
0 0
3 9999
9000 -9999
10003 9999
	2

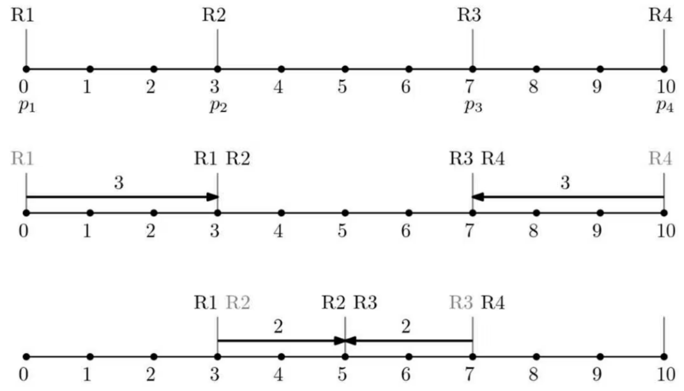
6
10 -9999
0 0
3 9999
9000 -9999
10003 -9999
10003 9999
	3

Problem J. Justify Battery Capacity

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

ICPC street is currently an undeveloped area, with a large-scale development plan scheduled soon. Before starting the development, information about n important points along the street will be collected using n remote-controlled robots, with each robot collecting information from one of these important points. Now, the goal is to combine all the collected information into a single robot to find the most efficient development approach. To combine the information, the robots can move left or right, and also can combine the information that they have with information from other robots. Also, each robot is powered by its own battery, and all the robots are equipped with identical batteries. Specifically, let $p_1, p_2, \dots, p_n$ represent the positions of the important points where the robots collect information, arranged from left to right. Then the requirements are as follows:

1. The ICPC street is considered as a one-dimensional interval $[0, L]$ with a positive integer L. The important points $p_1, p_2, \dots, p_n$ are always represented as integers on the interval, including two endpoints of the interval. That is, $p_1 = 0$ and $p_n = L$. Initially, each robot is positioned at one of the important points, so it has the information about this important point before beginning to move. Note that there is exactly one robot at each of these points initially, which means n is also the number of robots, and that $2 \le n \le L + 1$.
2. For combining the information a robot has with information from other robots, robots can move freely to the left or to the right, consuming 1 unit of battery power for 1 unit of distance traveled, regardless of direction. All robots are equipped with the same batteries. Each battery has an integer capacity P. A robot can move only in integer units of distance.
3. When two or more robots meet at the integer position on the street, they can combine their information. For example, if a robot with information about p_1 and p_2 meets a robot with information about p_3 and p_4, both robots will then have information about the positions p_1, p_2, p_3, and p_4.
4. Robots consume the battery only for movement. They do not use the battery when changing direction or when combining the information they have with information from other robots.
5. After all movements, at least one robot must have information about all the positions $p_1, p_2, \dots, p_n$.



For example, the figure above shows an example with $L = 10$ and $n = 4$. Robots 1, 2, 3, and 4 (R1, R2, R3, R4 in the figure) collect information (and are initially positioned) at $p_1 = 0$, $p_2 = 3$, $p_3 = 7$, and $p_4 = 10$, respectively. Then the following sequence of steps can be performed with a battery capacity of $P = 3$:

1. Robot 1 moves to p_2, and Robots 1 and 2 combine each other's information.
2. Robot 4 moves to p_3, and Robots 3 and 4 combine each other's information.
3. Robot 2 moves 2 units to the right, Robot 3 moves 2 units to the left, and they combine each other's information at the position 5 on the street.

After completing the process, Robots 2 and 3 will have information about all the positions p_1, p_2, p_3, and p_4.

Since the battery is much more expensive than the other parts of a robot, it is important to determine the minimum battery capacity required for each robot for efficient data collection. Given L, n, and the positions of the important points $p_1, p_2, \ldots, p_n$, write a program to calculate the minimum battery capacity P required for at least one robot to have information about all the points.

Input

The input starts with a line containing two positive integers, L and n ($1 \leq L \leq 10^6$; $2 \leq n \leq L+1$), where n is the number of robots and important points on the street and L is the position of the right endpoint of the street. The following line contains n distinct integers $p_1, p_2, \ldots, p_n$ ($0 = p_1 < p_2 < \ldots < p_n = L$) that represent the positions of the important points on the street.

Output

Print exactly one line. The line should contain a single integer: the minimum battery capacity P required so that at least one robot can have information about all the important points.

Examples

standard input	standard output
10 4 0 3 7 10	3
100 5 0 97 98 99 100	49
1 2 0 1	1

Problem K. Know The String Rank

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

Let w and u be strings consisting of the English lowercase alphabet. Denote their lengths as $|w| = n$ and $|u| = k$. We say that string u is a subsequence of string w if there exists a strictly increasing sequence of integers $i_1, \ldots, i_k$ such that $u[j] = w[i_j]$ for all $j = 1, \ldots, k$. Here, $v[i]$ denotes the i-th character of the string v.

Let $w[i{:}]$ denote the suffix $w[i] \ldots w[n]$. If $i > n$, then $w[i{:}]$ is the empty string denoted by λ.

Given a nonempty string w and a positive integer k, we define the *k-set* of w to be the set $Q_k(w)$ of subsequences of w whose lengths are $0, 1, \ldots, k$. This implies that, for any string w and any integer k, the empty string λ belongs to $Q_k(w)$ by definition.

For example, when $w = \texttt{aaba}$, we have $Q_3(\texttt{aaba}) = \{\lambda, \texttt{a}, \texttt{b}, \texttt{ba}, \texttt{ab}, \texttt{aa}, \texttt{aba}, \texttt{aab}, \texttt{aaa}\}$.

For a string w, we define the *rank* of w to be the minimum integer t such that the t-sets for all suffixes of w are all different. In other words, the rank of w is $\min\{t \geq 1 \mid Q_t(w[i{:}]) \neq Q_t(w[j{:}]), \forall 1 \leq i < j \leq n\}$.

For instance, when $w = \texttt{aaba}$, the 2-sets $Q_2(\texttt{aba})$ and $Q_2(\texttt{aaba})$ are equal. On the other hand, for $t = 3$, we have

- $Q_3(\lambda) = \{\lambda\}$,
- $Q_3(\texttt{a}) = \{\lambda, \texttt{a}\}$,
- $Q_3(\texttt{ba}) = \{\lambda, \texttt{a}, \texttt{b}, \texttt{ba}\}$,
- $Q_3(\texttt{aba}) = \{\lambda, \texttt{a}, \texttt{b}, \texttt{ba}, \texttt{ab}, \texttt{aa}, \texttt{aba}\}$,
- $Q_3(\texttt{aaba}) = \{\lambda, \texttt{a}, \texttt{b}, \texttt{ba}, \texttt{ab}, \texttt{aa}, \texttt{aba}, \texttt{aab}, \texttt{aaa}\}$.

Therefore, the rank of the string $w = \texttt{aaba}$ is 3.

Given a string w, write a program to output its rank.

Input

The input consists of a single nonempty string w, which consists only of lowercase characters from the English alphabet. The length of the string is at most $3 \cdot 10^6$.

Output

Print exactly one line. The line should contain a positive integer to represent the rank t of the input string w.

Examples

standard input	*standard output*
aabbb	3
abacb	2
azadzzadaz	4
a	1

Problem L. Largest and Smallest Areas

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

There is a triangle whose coordinates of three vertices A, B, and C are all integers. If you select a point on each side of the triangle whose coordinates are integers and connect those points, a new triangle is created. When creating a new triangle, no vertex of the given triangle can be selected as a vertex of the new triangle.

Depending on which points you select and connect, the area of the newly created triangle may be large or small.

You are to write a program that finds out the largest and smallest areas of the newly created triangle if they exist.

For example, as shown in the figure below, the coordinates of the three vertices of the given triangle are $(4, 8)$, $(-8, -1)$, and $(7, -7)$. The yellow triangle shown in Fig. L.1(a) has the largest area among those that satisfy the condition. The blue triangle shown in Fig. L.1(b) has the smallest area.

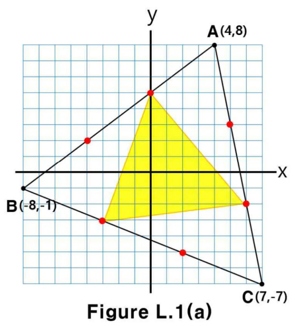


Figure L.1(a)

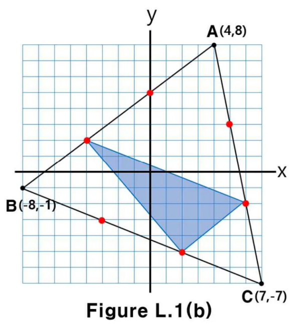


Figure L.1(b)

There may not be a point on any side of the given triangle whose coordinates are integers, in which case the triangle you are looking for does not exist.

It is guaranteed that the three points of the given input do not lie on a straight line.

Input

Let the triangle's vertices be $A = (A_x, A_y)$, $B = (B_x, B_y)$, and $C = (C_x, C_y)$. The input consists of a line containing six integers: A_x, A_y, B_x, B_y, C_x, and C_y, in this order. Each coordinate is an integer between -10^9 and 10^9, inclusive.

Output

Let the area of the newly created triangle with the largest area be $S_{\max}$, and the area of the triangle with the smallest area be $S_{\min}$. If such triangles can be found, print $2S_{\max}$ and $2S_{\min}$ in that order, where both $2S_{\max}$ and $2S_{\min}$ are positive integers. If such triangles cannot be found, print a single integer -1 instead.

Examples

standard input	*standard output*
`4 8 -8 -1 7 -7`	`69 46`
`-8 1 7 11 7 -5`	`121 23`
`0 0 1 10 10 0`	`-1`