Problem A
Arrested Development
Time Limit: 2 Seconds, Memory Limit: 2G

You are now in charge of two programming interns, and you must develop a large system. There are a number of tasks that need to be completed by the end of the summer. You know how long each intern will take to complete each task, in minutes.

Compute the minimum number of minutes it will take to complete all tasks for development of the system, assuming that the two interns are the only developers, that they work independently and concurrently, that they do not share tasks, and that the amount of time it takes an intern to complete all their tasks is the sum of the number of minutes it takes to do each task one after the other.

Input

The first line of input contains a single integer n ($1 \leq n \leq 50$), which is the number of tasks.

Each of the next n lines contains two integers a and b ($1 \leq a, b \leq 10^5$). Each line represents a single task, where a is the number of minutes it will take the first intern to complete the task, and b is the number of minutes it will take the second intern to complete the task.

Output

Output a single integer, which is the minimum number of minutes needed to complete the development project.

Sample Input 1	Sample Output 1
4 100 1 1 90 1 20 1 20	3

Sample Input 2	Sample Output 2
2 314 1 592 6	7

This page is intentionally left blank.

Problem B
Champernowne Substring
Time Limit: 10 Seconds, Memory Limit: 2G

The Champernowne string is an infinite string formed by concatenating the base-10 representations of the positive integers in order.

It begins `12345678910111213 14...`

It can be proven that any finite string of digits will appear as a substring in the Champernowne string at least once.

Given a string of digits and question marks, compute the smallest possible index that this string could appear as a substring in the Champernowne string by replacing each question mark with a single digit from 0 to 9. Each question mark can map to a different digit. Since this index can be large, print it modulo 998,244,353.

Input

The first line of input contains a single integer t ($1 \leq t \leq 10$), which is the number of test cases.

Each of the next t lines contains a string s ($1 \leq |s| \leq 25$) consisting of digits 0 to 9 or question marks.

Output

Output t lines. For each test case in order, output a single line with a single integer, which is the smallest possible index where the string could appear as a substring in the Champernowne string, modulo 998,244,353.

Sample Input 1	Sample Output 1
9 0 ???1 121 1?1?1 ??5?54?50?5?505?65?5 000000000000 ?2222222 ?3?????????9??8???????1??0 9?9??0????????????2	11 7 14 10 314159 796889014 7777 8058869 38886

This page is intentionally left blank.

Problem C
Comparator
Time Limit: 3 Seconds, Memory Limit: 2G

Many programming languages let you define custom comparators to sort user-defined objects.

IFFY is a programming language where the only programs are functions intended to be comparators. These functions operate on bitstrings, hereafter called "words". Words are 1-indexed starting with the leftmost character.

To write a function on two words in IFFY, the programmer writes a series of if-statements. Each if statement chooses a bit from the first word, a bit from the second word, and evaluates some expression on those two bits. If the expression evaluates to 1 on those chosen bits, then the function returns immediately with a specified return value. Otherwise, execution falls through to the next if statement, not returning anything.

The syntax of each if statement is a b *expr* r, where the integer a specifies a bit from the first word, integer b specifies a bit from the second word, and *expr* is a boolean expression using variables x and/or y, and r is the value (0 or 1) returned from the function if the expression evaluates to 1. The variable x refers to the specified bit in the first word, and the variable y refers to the specified bit in the second word. For example, consider the if statement '`2 3 x|y 0`'. This expression means: if the second bit of the first word or the third bit of the second word is set, then return 0, otherwise continue with the next statement.

The grammar for valid expressions is as follows:

- `x`, `y`, `0`, and `1` are valid expressions.
- If `E` is a valid expression, then `(E)` is a valid expression.
- If `E` is a valid expression, then `!E` is a valid expression.
- If `E1` and `E2` are valid expressions, then all strings of the form `E1 BIN_OP E2` are valid expressions, where `BIN_OP` is one of = (equals), `&` (and), `|` (or), or `^` (xor).

Parentheses take highest precedence, followed by the unary `!`, followed by the binary =, `&`, `|`, and `^`, in that order. All binary operators are left-associative.

Here are the truth tables for each operator:

x	$!x$
0	1
1	0

Figure C.1: The truth table for the sole unary operator.

x	y	$x = y$
0	0	1
0	1	0
1	0	0
1	1	1

x	y	$x\&y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x\|y$
0	0	0
0	1	1
1	0	1
1	1	1

x	y	$x\hat{}y$
0	0	0
0	1	1
1	0	1
1	1	0

Figure C.2: Truth tables for the binary operators.

In order for a function $f(x, y)$ to operate properly as a comparator, it must satisfy certain properties. Informally, for $f(x, y)$ to be a comparator, it should impose some ordering $<$, where $f(x, y)$ returns 1 if and only if $x < y$. For example, sample 1 is a valid comparator to sort bitstrings of length 2. More formally, three of the properties that comparators must satisfy are the reflexive, symmetric, and transitive properties, as follows:

1. **Reflexive:** For all x, $f(x, x)$ should return 0.
2. **Symmetric:** For all x, y, if $f(x, y)$ returns 1, then $f(y, x)$ must return 0.
3. **Transitive:** For all x, y, z, if both $f(x, y)$ and $f(y, z)$ return 1, then so must $f(x, z)$.

Given a function in the IFFY language, determine how well it satisfies these three properties by counting how often they are violated. First, among all words of a given length, count the number of words for which the reflexive property fails. Next, count the number of pairs of words for which the symmetric property fails. Finally, count the number of triples of words for which the transitive property fails.

Input

The first line of input contains two integers n ($0 \le n \le 2 \cdot 10^5$) and k ($1 \le k \le 10$), where n is the number of lines in the function and k is the number of bits in the values being compared.

Each of the next n lines contains four tokens of the form a b *expr* r, where a and b ($1 \le a, b \le k$) are the indices of the bits to consider in x and y, *expr* is a valid expression, and r ($0 \le r \le 1$) is the return value. The final line gives the returned value if no if statement is triggered. The total length of all expressions is at most 10^6.

Output

Output a single line with three space separated integers. The first integer is the number of words for which the reflexive property is violated, the second is the number of pairs of words for which the symmetric property is violated, and the third is the number of triples of words for which the transitive property is violated.

Sample Input 1

```
3 2
1 1 (x=0)&(y=1) 1
1 1 (x=1)&(y=(x^x)) 0
2 2 (x=1)|(y=0) 0
1
```

Sample Output 1

```
0 0 0
```

Sample Input 2

```
4 3
2 1 x=0&(y=1) 1
1 2 !x^!!y 0
2 3 ((x|1)=y)&1&1 1
3 1 !x&!x&!x 0
1
```

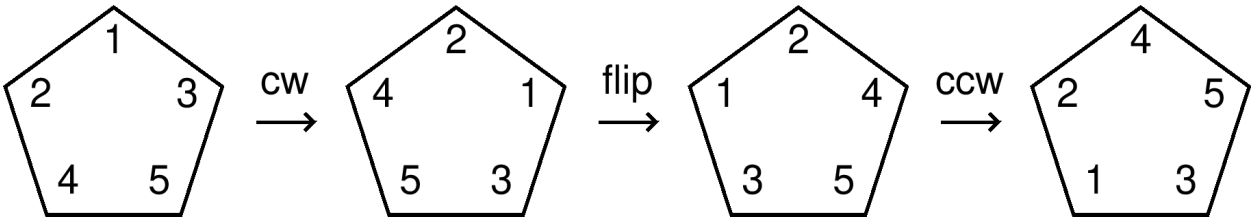
Sample Output 2

```
3 25 52
```

This page is intentionally left blank.

Problem D
Dihedral Group
Time Limit: 1 Second, Memory Limit: 2G

In mathematics, the dihedral group D_n is the group of symmetries of a regular n-gon. Rotations and reflections are elements of D_n, and in fact all elements of the dihedral group can be expressed as a series of rotations and reflections. Elements of D_n *act* on the n-gon by permuting its vertices. For example, consider a regular pentagon with vertices initially labeled 1, 3, 5, 4, 2 (clockwise, starting from the top):



Applying the above three dihedral actions to the pentagon (a rotation, reflection, and then another rotation) produces the following relabelings of the pentagon's vertices:

$$1, 3, 5, 4, 2 \to 2, 1, 3, 5, 4 \to 2, 4, 5, 3, 1 \to 1, 2, 4, 5, 3.$$

You are given an arbitrary clockwise labeling of the vertices of a regular n-gon using the integers 1 through n, and a second sequence to test. Determine whether it's possible to apply some series of dihedral actions to the n-gon so that the test sequence appears as a contiguous clockwise sequence of vertex labels on the transformed polygon.

Input

The first line of input has two integers n and m, ($1 \le m \le n \le 5 \cdot 10^4$) where n is the number of vertices of the polygon and m is the length of the sequence to be tested.

The next line contains n space-separated integers d ($1 \le d \le n$). This is the initial labeling of the polygon vertices. It is guaranteed that each integer from 1 to n appears exactly once.

The next line contains m space-separated integers t ($1 \le t \le n$). This is the sequence to be tested.

Output

Output a single integer, which is 1 if the test sequence could appear as a contiguous sequence of vertex labels after applying some series of dihedral actions to the initial polygon, and 0 otherwise.

Sample Input 1	Sample Output 1
3 3 1 2 3 1 3 2	1

Sample Input 2	Sample Output 2
3 1 1 2 3 1	1

Sample Input 3	Sample Output 3
4 2 1 2 3 4 1 3	0

Sample Input 4	Sample Output 4
4 4 1 2 3 4 2 3 4 1	1

Sample Input 5	Sample Output 5
4 4 1 2 3 4 3 2 1 4	1

Sample Input 6	Sample Output 6
5 3 1 3 5 4 2 2 1 3	1

Sample Input 7	Sample Output 7
5 4 1 3 5 4 2 2 1 5 3	0

Problem E
House Deconstruction
Time Limit: 1 Second, Memory Limit: 2G

In the land of Circleland, there is a circle that has equally spaced points around its circumference. The distance between any two adjacent points is 1.

There are people and houses on the circle's points. Each point contains a person, an empty house, or nothing at all. Each person would like to walk to a different house. Each house can contain at most one person. People can only walk along the circumference of the circle; they cannot cut across.

Currently, there are more houses than people, so you'd like to destroy some of the houses. Suppose you destroy a set of houses S. Let $f(S)$ be the minimum total amount of walking needed to get each person to a different non-destroyed house.

Compute the minimum value of $f(S)$, compute how many sets of houses S achieve this minimum value. Since the number of sets S can be large, output it modulo 998,244,353.

Input

The first line of input contains three integers x, n, and m ($1 \le n < m \le 2 \cdot 10^5, n+m \le x \le 10^9$), where x is the number of points around the circle, n is the number of people, and m is the number of houses. The points are labeled 1 to x, and point x is adjacent to point 1.

The next $n + m$ lines each contain two tokens, an integer p ($1 \le p \le x$) and a character t ($t \in \{\texttt{P}, \texttt{H}\}$), where p denotes the position of a person or house, and t denotes the type of the point, either `P` for person or `H` for house. All positions are distinct, and the positions will be given in increasing order.

Output

Output two lines. On the first line output the minimum possible value of $f(S)$. On the second line output the number of sets S that achieve this minimum value, modulo 998,244,353.

Sample Explanation

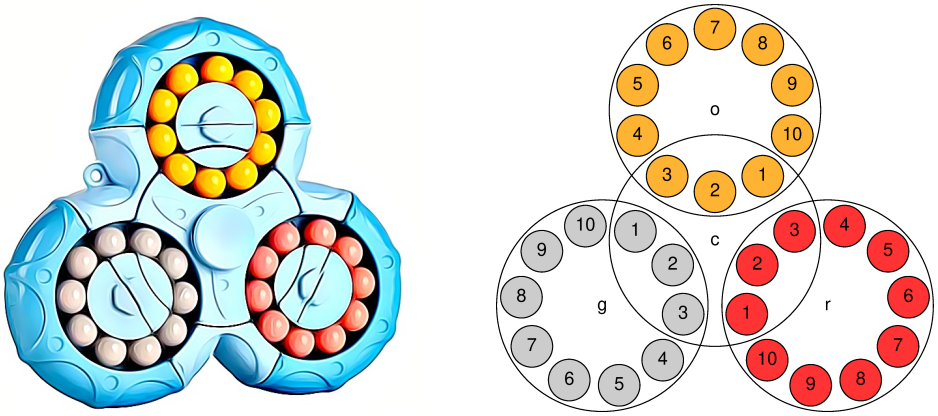
For the first sample, the minimum total walking distance is 2. We can destroy the set of houses at $\{2, 5\}$, $\{4, 5\}$ or $\{5, 6\}$.

For the second sample, we can destroy the set of houses at $\{6, 31415926\}$ for a minimum total walking distance of 4. Note that even though the minimum walking distance can be achieved in multiple ways, it is only counted once since the set of destroyed houses is the same.

Sample Input 1	Sample Output 1
6 2 4 1 P 2 H 3 P 4 H 5 H 6 H	2 3

Sample Input 2	Sample Output 2
1000000000 2 4 1 P 6 H 31415926 H 999999998 H 999999999 H 1000000000 P	4 1

Problem F
Magic Bean
Time Limit: 2 Seconds, Memory Limit: 2G



A "Magic Bean" is a combination fidget toy and puzzle, with 30 colored beads moving in three circular tracks. In its solved state, the upper circle contains ten indistinguishable orange beads, the lower left circle contains ten indistinguishable grey beads, and the lower right circle contains ten indistinguishable red beads. The beads can be rotated in each circle. In addition, there is a fourth circle in the middle that can be rotated, and in doing so exchanges consecutive triples of beads among the circles.

Your brother just borrowed your Magic Bean and arbitrarily rotated those circles, scrambling the beads. Your job is to solve the puzzle, by finding a sequence of valid rotations of the four circles that leads back to the solved state. You need not find the shortest such solution, but your solution should use no more than 240 moves. The provided input will be the state of the puzzle after applying some sequence of at most 240 moves.

Input

Input consists of exactly three lines. Each line contains a string of length ten, consisting only of the characters `o`, `g` and `r`. The first line describes the ten beads in the top circle, the second describes the lower-left circle, and the third describes the lower-right circle.

The character `o` designates an orange bead, `g` a gray bead, and `r` a red bead. The beads are listed in clockwise order, numbered according to the diagram.

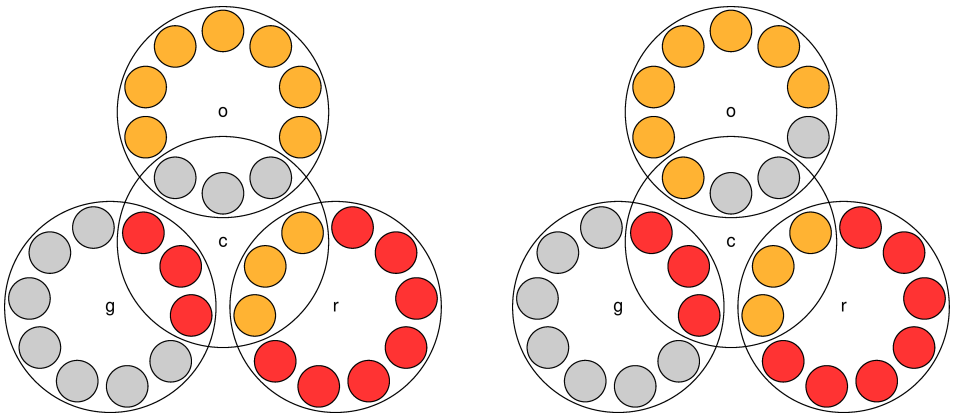
The input configuration is the result of applying at most 240 moves on a solved puzzle.

Output

Output a single integer k, with $0 \leq k \leq 240$, which is the number of moves you needed to solve the puzzle.

Next, if $k > 0$, output the k moves on k lines, with one move per line, in order. Each move consists of a single character, specifying the circle to be rotated, followed immediately by a single digit specifying how far to rotate the circle in a clockwise direction. The top circle is specified with '`o`', the lower left circle with '`g`', the lower right circle with '`r`', and the center circle with '`c`'. For the first three circles, the digit specifying the rotation has a value from 1 to 9; for the central circle, it has a value from 1 to 2.

The following two images show the input positions for the two samples.



Sample Input 1	Sample Output 1
gggooooooo rrrgggggggg ooorrrrrrr	1 c2

Sample Input 2	Sample Output 2
ggooooooog rrrgggggggg ooorrrrrrr	2 o1 c2

Problem G
Manhattan Walk
Time Limit: 1 Second, Memory Limit: 2G

It's a grid system! You begin at the top left corner and want to walk to the bottom right corner. Every location is at integer coordinates, and has an arrow pointing down or right and a timer that, every few seconds, flips the arrow from down to right, or from right to down. When you begin your walk, every arrow is pointing down or right with equal probability, and every timer has a real value chosen uniformly in the range from zero to its maximum wait time.

At any moment in time, if you are at a given location, you can:

- Move down one location if the new location is on the grid and the arrow is pointing down.
- Move right one location if the new location is on the grid and the arrow is pointing right.
- Wait for the timer to finish its countdown so that the arrow flips from down to right, or from right to down.

When you arrive at a new location, you are able to see the timer and can therefore take that into account when deciding which action to take. However, you are not able to look ahead—you can only see the timer and arrow for the exact grid point you occupy. You hate waiting, and want to minimize the total amount of time you're waiting for an arrow to flip.

What is the expected amount of time you have to wait if you make decisions optimally?

Input

The single line of input contains three integers r, c ($1 \leq r, c \leq 10^3$), and p ($1 \leq p \leq 10^9$), where r is the number of rows in the grid, c is the number of columns in the grid, and p is the maximum value a timer can show.

Output

Output a single number, which is the expected time you have to wait if you make optimal decisions. Your answer will be accepted if the absolute or relative error is within 10^{-6} of the judge's answer.

Sample Input 1	Sample Output 1
2 3 8	2.875

Sample Input 2	Sample Output 2
5 5 5	2.43223387

This page is intentionally left blank.

Problem H
MountainCraft
Time Limit: 5 Seconds, Memory Limit: 2G

It's a glorious day in the video game meta-world of MountainCraft! True to its name, MountainCraft boasts an expansive open world full of mountains to explore. Your avatar in the game wakes up on a remote island, and stares at the mountains on the horizon.

The view of the horizon can be modeled as a Cartesian plane, with the x-axis ($y = 0$) separating land from sea. Each mountain peak is represented by a point (x, y), and the mountain's sides have slope $+1$ and -1, forming a triangle with that peak at the top.

Your avatar can only see the parts of the mountains which are in a viewport. The visible edges of the mountains, where they do not overlap other mountains, are rendered in bold. If mountains overlap, the overlapping parts are not rendered in bold. Edges do not have to intersect for mountains to overlap.

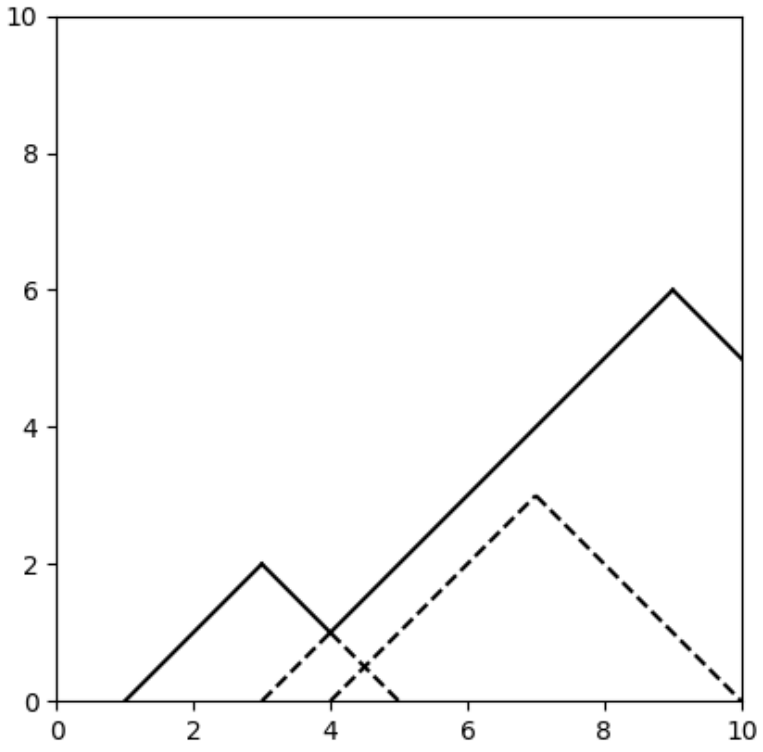


Figure H.1: The first sample input after all mountains appear.

Unfortunately, due to graphical glitches, mountains can appear and disappear. After each change, you want to know the total length of all the bold lines currently visible in the viewport.

Input

The first line of input contains two integers q ($1 \le q \le 2 \cdot 10^5$) and w ($1 \le w \le 10^9$), where q is the number of queries and w is the width of the viewport. The viewport stretches from 0 to w.

Each of the next q lines contains two integers x ($0 \le x \le w$) and y ($0 < y \le 10^9$), the peak of some mountain. If (x, y) is the peak of a mountain that is visible, that mountain will disappear. Otherwise, that mountain will become visible.

Output

Output q lines. For each query in order, output a single line with a single number, which is the total length of the bold lines rendered. Your answer will be accepted if the absolute or relative error is within 10^{-6} of the judge's answer.

Sample Explanation

In the first sample, the first two mountains intersect at the point $(4.5, 0.5)$ The parts of the mountains below that point are not rendered in bold.

In the second sample, the left mountain appears, then disappears, then appears again. The right mountain appears and then disappears.

Sample Input 1	Sample Output 1
3 10 3 2 7 3 9 6	5.656854 12.727922 12.727922

Sample Input 2	Sample Output 2
5 100 31 41 59 26 31 41 59 26 31 41	101.823376 120.208153 73.539105 0.000000 101.823376

Problem I
Not Another Constructive!
Time Limit: 1 Second, Memory Limit: 2G

Sick of solving geometry problems, you decide to solve the following constructive problem: find a string of length n that contains exactly k not necessarily contiguous subsequences of NAC.

This problem seems too familiar though. Here's the twist - your friend has given you part of the string, so you must fill in the remaining characters!

Input

The first line of input contains two integers n ($1 \leq n \leq 40$) and k ($0 \leq k \leq 2{,}500$), where n is the length of the string and k is the number of not necessarily contiguous subsequences of NAC that the output must contain.

The second line contains a string of length exactly n, consisting only of uppercase letters and/or question marks.

Output

Output a string of upper case letters, replacing each question mark in the input string with an uppercase letter so that the resulting string has exactly k subsequences of NAC. If this is not possible, output -1. Any uppercase letters in the input string must be kept in their position. There may be multiple possible solutions for any given test case; any correct solution will be accepted.

Sample Input 1	Sample Output 1
22 2 N??A??????C??????????	NOTANOTHERCONSTRUCTIVE

Sample Input 2	Sample Output 2
18 0 COUNTINGSATELLITES	COUNTINGSATELLITES

Sample Input 3	Sample Output 3
2 1 ??	-1

This page is intentionally left blank.

Problem J
Passport Stamps
Time Limit: 1 Second, Memory Limit: 2G

You just got your new passport, fresh with pages ready to be stamped by immigration officers. Sadly, because your passport has so many pages, immigration officers are too lazy to try to use your pages efficiently, so you may need to get a new passport sooner than you think...

You have some trips prepared. For each trip, when you go through passport control, the immigration officer will look for some contiguous pages, none of which are stamped, and then stamp all of them. Because the officer is lazy, there is no guarantee which contiguous pages get stamped.

You will start making these trips in order until your passport no longer has enough contiguous empty pages to satisfy the next trip, at which point you will apply for a new passport. Your plans are fixed - you will not skip a trip in the middle even if it means you could make more future trips.

If the immigration officers conspire against you, find the first trip where it is possible you cannot travel because your passport does not have enough consecutive blank pages (or if you can always make all trips without running out of blank pages).

Input

The first line of input contains two integers n ($1 \leq n \leq 10^5$) and p ($1 \leq p \leq 10^{18}$), where n is the number of trips you have planned, and p is the number of pages in your new passport.

Each of the next n lines contains a single integer c ($1 \leq c \leq 10^{18}$), which is the number of contiguous pages which will need to be stamped for that trip.

Output

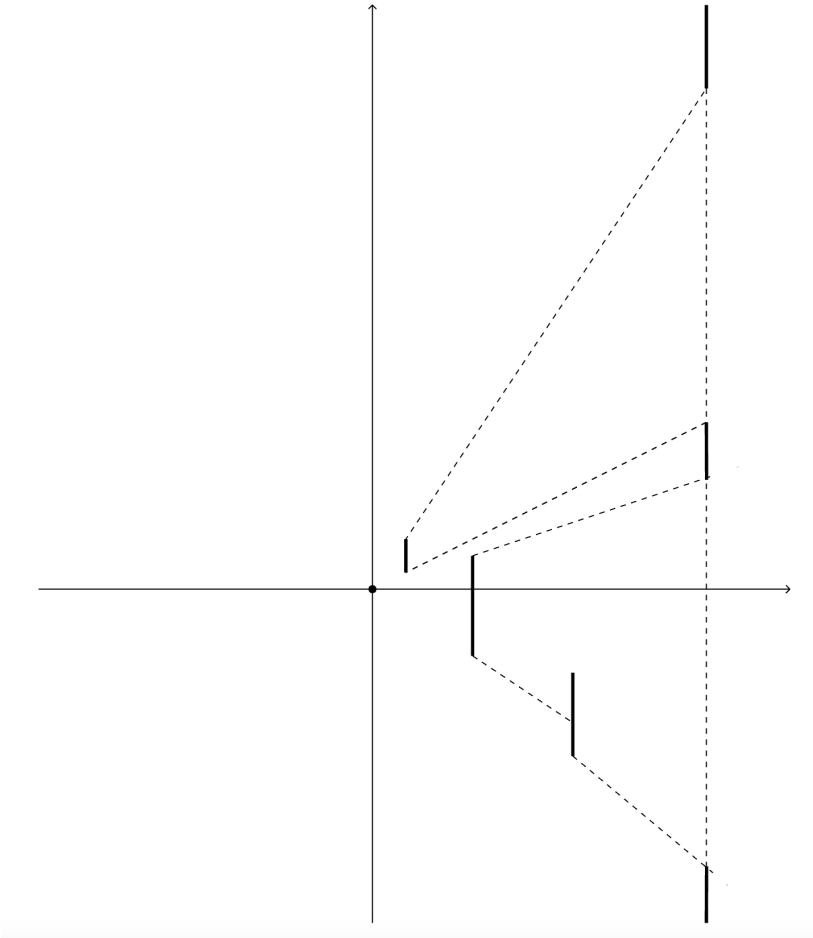
Output a single integer, which is the minimum number of trips you could possibly take before you would need a new passport because there aren't enough consecutive blank pages for your next trip. Output n if you can take all trips, even in the worst case.

Sample Input 1	Sample Output 1
5 15 1 2 3 4 5	3

This page is intentionally left blank.

Problem K
Shadow Line
Time Limit: 10 Seconds, Memory Limit: 2G

You have a point light source at the origin in the 2D plane. To the right there is an infinitely tall wall. There are some opaque vertical line segments between the light source and the wall. As a result, each line segment casts a shadow onto the wall. All of these shadows overlap to form one or more intervals on the wall.



Now imagine moving the light source along the x-axis in the negative direction, effectively pulling the light further away from all the objects in a straight line. As the light source moves, the shadows move as well, potentially changing the number of shaded intervals on the wall. Your job is to compute the sum of the lengths of the intervals along the x-axis for which the light source creates a single shaded interval on the wall.

Input

The first line of input contains two integers n ($1 \leq n \leq 3{,}000$) and w ($2 \leq w \leq 10^6$), where n is the number of opaque vertical segments, and w is the x-coordinate of the infinitely tall wall.

Each of the next n lines contains three integers x ($0 < x < w$), y_{low} and y_{high} ($-10^6 \leq y_{low} < y_{high} \leq 10^6$). Each set of three integers describes a line segment from (x, y_{low}) to (x, y_{high}). All y-coordinates will be unique. No two line segments will intersect or overlap.

Output

Output a single number, which is the sum of the lengths of the intervals along the x-axis for which the light source creates a single shaded interval on the wall. If this sum includes an unbounded interval (i.e. there is a single shaded interval when the light source is infinitely far away), print -1 instead. Your answer will be accepted if the absolute or relative error is within 10^{-6} of the judge's answer.

Sample Input 1	Sample Output 1
3 20 2 1 3 6 -4 2 12 -10 -5	16.0

Problem L
Square of Triangles
Time Limit: 3 Seconds, Memory Limit: 2G

You are given the squares of the lengths of the sides of four triangles. Determine if it is possible to arrange them (via translation, rotation, and reflection) into a square. No triangles may overlap, and there should be no gaps or holes.

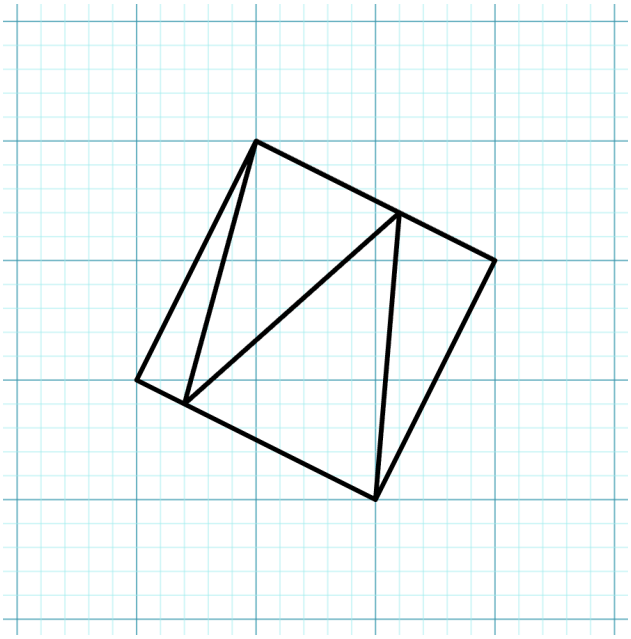


Figure L.1: A solution to the third test case in the sample input.

Input

The first line of input contains a single integer t ($1 \leq t \leq 20$), which is the number of test cases.

Each of the next $4 \cdot t$ lines describes t test cases, consisting of four triangles each, one triangle per line. Each triangle consists of three integers a, b and c ($1 \leq a, b, c \leq 10^7$). Each of the integers is equal to the **square** of the length of a side of a triangle. For example, if the three sides of a triangle have lengths 3, 4 and 5, then the input would be `9 16 25`. The integers will **not** necessarily be perfect squares. It is guaranteed that the given triples each represent a triangle of positive area.

Output

Output t lines. For each test case in order, output a single line with a single integer, which is 1 if the four triangles of the test case can be arranged into a square, and 0 otherwise.

Sample Input 1	Sample Output 1
3 1 1 2 2 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 5 125 130 125 20 145 45 130 145 145 145 80	1 0 1

Problem M
Training, Round 2
Time Limit: 6 Seconds, Memory Limit: 2G

Ashley is training for another programming contest on Brandon's Online Judge. Brandon's Online Judge still has the same feature which allows Ashley's coach, Tom, to load in a list of problems for Ashley to work on.

Tom has curated some problems for Ashley to work on. Each problem is parameterized by "implementation difficulty" with a range, and "thinking difficulty" with a range.

Ashley starts out with a given implementation skill and thinking skill level. Ashley will train on Tom's curated list of problems as follows: she will look at the first problem on the list and either solve it or skip it. She will then repeat this for every problem in the list in the order Tom loaded the problems. Once she has skipped a problem, she can never go back to it. Ashley will only solve a problem if her current implementation skill and current thinking skill both land within the range, inclusive, for the given problem. After solving a problem, Ashley will reflect on her experience, which allows her to increase either her implementation skill level by one or her thinking skill level by one. She cannot increase both simultaneously, nor can she skip the reflection.

Compute the maximum number of problems that Ashley can solve if she plans her reflections optimally.

Input

The first line of input contains three integers n ($1 \leq n \leq 5{,}000$), i and t ($0 \leq i, t \leq 10^9$), where n is the number of problems Tom has given Ashley, i is Ashley's starting implementation skill level and t is Ashley's starting thinking skill level.

Each of the next n lines contains four integers i_{low}, i_{high} ($0 \leq i_{low} \leq i_{high} \leq 2 \cdot 10^9$), t_{low}, t_{high} ($0 \leq t_{low} \leq t_{high} \leq 2 \cdot 10^9$). Each line describes a problem, where its implementation difficulty is in the range $[i_{low}, i_{high}]$ and its thinking difficulty is in the range $[t_{low}, t_{high}]$, each inclusive. Ashley can only solve a problem if her current implementation skill levels falls within the implementation difficulty range and her thinking skill level falls within the thinking difficulty range, each inclusive.

Output

Output a single integer, which is the maximum number of problems Ashley can solve if she chooses her reflection after each solution optimally.

Sample Input 1	Sample Output 1
3 0 0 0 1 0 1 1 1 0 1 1 1 1 1	3