Problem A. Antenna Analysis

Source file name:	Antenna.c, Antenna.cpp, Antenna.java, Antenna.py
Input:	Standard
Output:	Standard

Åke has heard that there may be some suspicious 5G radiation in his city. To test this, he uses the antenna on his roof to measure the 5G level each day. However, he does not know how he should analyze the data.

We are given the measurements for n consecutive days as a list of numbers $x_1, \ldots, x_n$ (where x_i denotes the measurement for day i) and a constant c that measures how much Åke expects the radiation to vary from day to day. We want to find, for each day i, the most significant difference between the measurement on day i and any earlier day, after the expected variations are taken into account. More precisely, the goal is to find the maximum value of

$$|x_i - x_j| - c \cdot |i - j|$$

where $j \leq i$. I.e., we want to find a large difference in 5G level that has happened recently.

Input

The first line of input contains the two integers n and c ($1 \leq n \leq 4 \cdot 10^5$, $1 \leq c \leq 10^6$), the number of measurements and expected day-to-day variation. The second input line contains the n integers $x_1, x_2, \ldots, x_n$ ($1 \leq x_i \leq 10^6$ for $i = 1, 2, \ldots, n$), giving the measurements of the n days.

Output

Output n integers $y_1, \ldots, y_n$, where y_i is the most significant difference on day i.

Example

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

Problem B. Breaking Bars

Source file name: Bars.c, Bars.cpp, Bars.java, Bars.py
Input: `Standard`
Output: `Standard`

Selma is visited by her two grandchildren Elsa and Asle who love chocolate. To be precise, they are especially fond of the brand Nut Cream Puffed Chocolate that comes in bars made up by 6×6 squares. The bars can be broken along the valleys between squares into smaller rectangular bars of integer dimensions. Due to the fragile nature of this type of chocolate, the bars often break into smaller rectangular bars even before you unpack them (but still only of integer dimensions).

Thus Selma finds herself with a set of rectangular bars of various dimensions in her candy stash. She knows how important it is to be fair to children, so not only does she want to give Elsa and Asle the same amount of chocolate, but also identical *collections* of rectangular bars (where an $a \times b$ bar is considered identical to a $b \times a$ bar). To do this, Selma can break her bars into smaller pieces. A *break* is the operation of taking an $a \times b$ bar and breaking it along a valley to produce two bars of dimensions $c \times b$ and $(a-c) \times b$, for some integer $c \in [1, a-1]$, or two bars of dimensions $a \times d$ and $a \times (b-d)$, for some integer $d \in [1, b-1]$. See Figure 1 for an example.

Selma would like to give her two grandchildren identical collections of bars, each collection consisting of at least t squares of chocolate. What is the minimum number of breaks she needs to make to be able to do this?

Explanation of Sample Input 1. First make a vertical break as shown on the 3×5 bar (orange), then make a horizontal break on the newly created 3×2 bar (blue). This way Elsa and Asle can each get one 1×2, one 2×2, and one 3×3 bar, in total 15 squares each.

Input

The first line of input contains two integers n and t ($1 \leq n \leq 50$, $1 \leq t \leq 900$), where n is the number of bars Selma has, and t is the least number of squares she wants each grandchild to receive. Then follows a line containing n bar descriptions. A bar description is on the format "axb" for two integers $1 \leq a, b \leq 6$.

You may assume that the total amount of chocolate squares among the n bars is at least $2t$.

Output

Output the minimum number of breaks needed to obtain two identical collections of bars, each having a total of at least t squares.

Example

Input	Output
4 15 1x2 2x2 3x3 3x5	2
6 7 1x2 2x3 1x4 3x2 4x1 6x6	0
5 3 1x1 1x1 1x1 1x1 1x4	1

Problem C. Candy Contribution

Source file name:	Candy.c, Candy.cpp, Candy.java, Candy.py
Input:	Standard
Output:	Standard

While you were out travelling, you won the lottery. As it happened, the top prize of this lottery was not cash, but candies! Now you are stuck with a big pile of candies which you would like to take home. Fortunately, you have been able to acquire a truck, so now all you have to do is drive home.

Going from one country to another with such a big pile of candies in your truck is not allowed without paying some taxes. And because everybody likes candies, you are allowed to pay these taxes with candies.

After searching a bit on the internet, you have found a list that tells you exactly which borders you can cross with a truck and for each such border what percentage of tax you have to pay to cross it. You cannot pay with fractional candies and the candies are quite nice, so customs will always round up. You only have to pay taxes on the number of candies you bring across the border.

What is the maximum number of candies you can bring home?

Input

The input consists of:

- One line containing two integers n ($2 \le n \le 1 \cdot 10^5$), the number of countries, and m ($1 \le m \le 2 \cdot 10^5$), the number of borders.
- One line containing three integers s ($1 \le s \le n$), the country where you won the lottery, t ($1 \le t \le n$, $t \ne s$), your home country and c ($1 \le c \le 10^9$) the number of candies you won in the lottery.
- Then follow m lines containing three integers u, v ($1 \le u, v \le n$, $u \ne v$) and p ($0 \le p \le 100$) where p is the percentage of tax you have to pay when travelling from country u to v or vice versa.

It is guaranteed you can drive home with your truck, and that each pair of countries is listed at most once.

Output

Output the maximum number of candies you can arrive home with.

Example

Input	Output
4 4 1 4 1000 1 2 25 2 4 10 1 3 4 3 4 30	675
5 5 1 5 6 1 2 17 2 5 19 1 3 1 3 4 1 4 5 1	3

Problem D. Deceptive Directions

Source file name:	Directions.c, Directions.cpp, Directions.java, Directions.py
Input:	`Standard`
Output:	`Standard`

You find yourself on a remote island, searching for a legendary lost treasure. However, despite having gotten your hands on directions leading straight to the treasure, you have a problem. It turns out you have a saboteur in your expedition, and that at some point they edited the precious directions so they might no longer lead to the treasure.

The island can be viewed as a rectangular grid, and the instructions are a sequence of east/west/north/-south steps to take in this grid, from a given starting position. These instructions lead straight to the treasure (but may involve walking around obstacles) in the sense that there is no shorter way of reaching the treasure. However, the saboteur has arbitrarily replaced each step of the instructions by a step in one of the other three directions. In other words, any "west" step has been replaced by "east", "north" or "south". This replacement has been done independently for each step, so one "west" may have been replaced by "north" and another by "south", and so on.

Because of this sabotage, the instructions seem pretty useless. But maybe they can still be used to narrow down the search. Write a program to find all possible locations of the treasure.

Input

The first line of input consists of two integers w and h ($3 \leq w, h \leq 1000$), the width and height of the map. Then follow h lines, each containing w characters, describing the map. Each such character is either a '`.`' symbolizing a walkable space, '`#`' symbolizing an obstacle such as a body of water, dense forest, or a mountain, or '`S`' symbolizing the starting point of the directions.

Finally, there is a line containing a string I ($1 \leq |I| \leq 10^5$) consisting only of the characters '`NWSE`', giving the faulty instruction sequence.

The map has exactly one '`S`' and its boundary consists only of obstacle cells. The faulty instruction sequence is such that there is at least one possible location of the treasure.

Output

Output the map in the same format as the input (without the first line specifying the dimensions), with all possible locations of the treasure indicated by exclamation marks ('`!`').

Example

Input	Output
5 5 ##### #...# #.S.# #...# ##### N	##### #...# #!S!# #.!.# #####
7 5 ####### #..#..# #..S..# #..#..# ####### ESS	####### #!.#..# #..S..# #..#..# #######

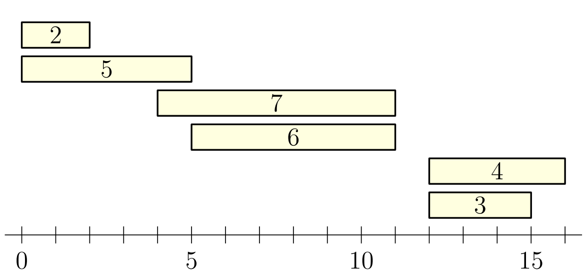
Problem E. Eavesdropper Evasion

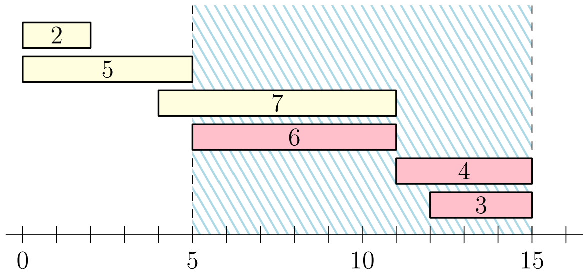
Source file name:	Evasion.c, Evasion.cpp, Evasion.java, Evasion.py
Input:	`Standard`
Output:	`Standard`

Alice wants to send n messages to Bob over a communication channel. The ith message takes t_i time steps to send. At each *integer* time step, Alice can start sending any number of her messages. Once started, a message must be transmitted in its entirety (it cannot be paused and resumed later). Any number of messages can be sent in parallel over the channel without affecting the transmission time of individual messages.

An attacker has the capability to disable the security protocols of the channel for an interval of x continuous time steps, but only once (i.e., after doing this, they cannot wait a while and then disable it for another x time steps). While the security is disabled, the attacker is able to listen in, and any message that is sent *in its entirety* during those x time steps is considered exposed.

What is the minimum time needed for Alice to send all n messages to Bob so that *at most* two messages are exposed, no matter when the attacker chooses to disable the security?





Left: Illustration of a solution to Sample Input 1. Right: sending the message of length 4 a time step earlier would not be a solution, because the three messages of length 6, 4, and 3 would then be exposed to an eavesdropper listening in from time step 5 to time step 15.

Input

The first line of input contains the two integers n and x ($1 \le n \le 20\,000$, $1 \le x \le 10\,000$), the number of messages Alice wants to send and the number of time steps someone may listen in. This is followed by a line containing n integers $t_1, \ldots, t_n$ ($1 \le t_i \le 10\,000$), the number of time steps it takes to transmit each message.

Output

Output the minimum number of time steps to complete transmission of all n messages so that at most two of them can be exposed.

Example

Input	Output
6 10 2 3 4 5 6 7	16
7 6 9 3 2 3 8 3 3	11

Problem F. Fortune From Folly

Source file name:	Folly.c, Folly.cpp, Folly.java, Folly.py
Input:	Standard
Output:	Standard

Your friend Ómar's favourite video game is *Striker-Count*. But he has now grown tired of actually playing the game and is more interested in the lootboxes found in the game. Inside each lootbox there is an item of some level of rarity. Ómar is only interested in acquiring the rarest items in the game. When he starts the game, he chooses two numbers n and k, such that $k \leq n$. He then opens lootboxes in the game until k of the last n lootboxes included an item of the highest rarity.

This activity amuses Ómar, but does not interest you in the slightest. You are more interested in the numbers: you know that each lootbox Ómar opens has probability p of containing an item of highest rarity, independently for each lootbox. You want to find the expected number of lootboxes Ómar will open before concluding his process.

Input

The only line of the input contains the two integers n and k ($1 \leq k \leq n \leq 6$), and the real number p ($0 < p \leq 1$ and p has at most four decimals after the decimal point), with meanings as described above.

Output

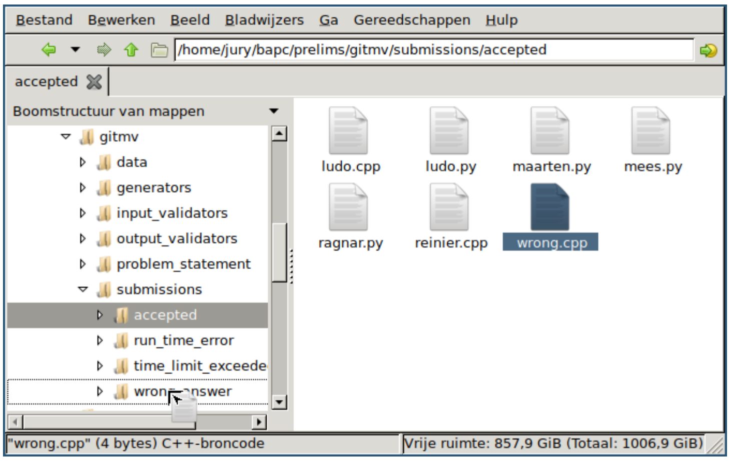
Output the expected number of lootboxes Ómar must open, with an absolute error of at most 10^{-1}. It is guaranteed that the input is such that this expected number does not exceed 10^9.

Example

Input	Output
3 2 0.0026	74445.39143490087
6 1 0.0026	384.61538461538464

Problem G. git mv

Source file name:	Gitmv.c, Gitmv.cpp, Gitmv.java, Gitmv.py
Input:	`Standard`
Output:	`Standard`



During development, you recently moved a file from one location to another. To keep your development team up to date with the change you made, you want to send them a short description of the change, without making use of any versioning software.

Both the source location and destination are valid Unix path names, that is, a nonempty string consisting of lowercase letters and "/" such that no "/" occurs at the begin or the end, nor does it contain two consecutive forward slashes.

You need to find the shortest string of the form "`A{B => C}D`" such that:

- The source location is "`ABD`" and the destination is "`ACD`", where double forward slashes should be read as one forward slash. For example, if a file is moved from "`a/c`" to "`a/b/c`", we can describe this movement by "`a/{ => b}/c`", meaning the source location was "`a/c`" and not "`a//c`".
- The string A is empty or ends with a forward slash, and similarly D is empty or starts with a forward slash.
- Both B and C do not start or end with a forward slash.

Input

The input consists of:

- One line containing the source location.
- One line containing the destination location.

Both lines will contain at most 10^6 characters, will not begin or end with a forward slash and will not contain any directory name twice. The two strings are guaranteed to be different.

Output

Output the shortest replacement string that transforms the source location to the destination, satisfying the above constraints.

Example

Input
`www/public/passwords` `private/passwords`
Output
`{www/public => private}/passwords`

Input
home/linus/downloads/image home/linus/pictures/recent/image
Output
home/linus/{downloads => pictures/recent}/image

Problem H. Hiring Help

Source file name: Help.c, Help.cpp, Help.java, Help.py
Input: `Standard`
Output: `Standard`

A certain large unnamed software development company has n developers. The productivity of each coder working for the company has been rigorously measured in terms of two key performance indicators: the number of lines of code they write per hour, and the number of bugs they fix per hour.

When a project needs to be done, the manager in charge of the project is allocated some budget of t man-hours of programmer time. The manager can then staff different coders on the project, up to a total of t hours. For instance if there are three programmers, the manager can allocate any non-negative real numbers t_1, t_2, and t_3 hours of their respective work hours, as long as $t_1 + t_2 + t_3 \leq t$. If the three programmers write l_1, l_2, and l_3 lines of code per hour, a total amount of $t_1 \cdot l_1 + t_2 \cdot l_2 + t_3 \cdot l_3$ lines of code will then be written for the project. Similarly if they fix b_1, b_2, and b_3 bugs per hour, a total of $t_1 \cdot b_1 + t_2 \cdot b_2 + t_3 \cdot b_3$ bugs will be fixed.

Due to the uncertain economy, the company has a hiring freeze, meaning that no new coders are hired to the company. However, under certain conditions, a manager is allowed to bring in outside help by outsourcing a project to an external consultant rather than doing it in-house. But this is only allowed if it is not possible to do the project equally efficiently in-house. In particular, if the consultant writes ℓ lines of code and fixes b bugs in t hours, and there exists some allocation of our existing coders which would write at least ℓ lines of code *and* fix at least b bugs in at most t hours, then a manager is *not* allowed to hire this consultant (regardless of whether those existing coders would actually have time to work on the project or whether they are already too busy with other projects).

While no new coders are hired, employees do sometimes decide to leave the company. Given a chronological list of events – requests to use a consultant, and employees quitting – find out which of the requests will be approved.

Input

The first line of input consists of a single integer n ($0 \leq n \leq 2 \cdot 10^5$), the number of coders (initially) at the company. The employees are numbered from 1 to n (names are too personal). Then follow n lines, the ith of which contains two integers ℓ_i and f_i ($1 \leq \ell_i, f_i \leq 10^8$), the number of lines of code and the number of bugs fixed per hour by coder i.

Next follows a line with a single integer e ($1 \leq e \leq 10^5$), the number of events. This is followed by e lines, describing the events in chronological order. An event is a line in one of the following two forms:

- "`c` t ℓ f", for three integers t, ℓ and f ($1 \leq t \leq 100$, $1 \leq \ell, f \leq 10^8$): a request to take in a consultant for a project of t hours, where the consultant would write ℓ lines of code and fix f bugs in those t hours.
- "`q` i", for an integer i ($1 \leq i \leq n$): coder i quit the company.

You may assume that no coder quits more than once.

Output

For each request to take in a consultant, output "`yes`" if the request is approved, and "`no`" if it is not approved.

Example

Input	Output
4 200 100 100 200 100 100 200 200 5 c 10 2000 2000 c 5 750 750 q 4 c 3 600 600 c 10 1500 1500	no no yes no
8 400 300 300 200 300 400 200 300 500 500 100 500 100 100 500 100 12 c 4 1611 1601 c 3 602 601 c 2 399 795 c 1 395 206 q 7 q 6 q 5 q 4 c 4 1611 1601 c 3 602 601 c 2 399 795 c 1 395 206	no no no no yes no no no

Problem I. Intact Intervals

Source file name:	Intervals.c, Intervals.cpp, Intervals.java, Intervals.py
Input:	`Standard`
Output:	`Standard`

Gustav is an astronaut on the Nordic Celestial Planetary Craft (NCPC), a large space station in orbit around Mars. Today, one of Gustav's tasks is to look over the satefy routines on board.

The space station consists of n modules arranged in a circle, so that module i is connected to module $i+1$ for $i = 1 \dots n-1$, and module n is connected to module 1. Each module i has a non-negative integer type a_i, representing the kind of equipment that can be found there. Different modules can have the same type. In case of emergency, the equipment must be rearranged so that each module i instead gets type b_i, for some list $b_1, b_2, \cdots, b_n$. Here, the list b is a rearrangement of the list a.

Gustav has noticed that if some module connections are severed, causing the space station to split into separate parts, it may become impossible to perform this rearrangement of the equipment. He decides to estimate how likely it is that the safety routines can be followed, by calculating in how many ways the space station can be separated into two or more parts such that it is still possible to rearrange the equipment according to the emergency procedures.

In other words, your task is to count in how many ways the circular list a can be partitioned into *at least two* non-empty contiguous intervals, in such a way that the circular list b can be obtained by rearranging elements within each interval. Since this number can be quite big, you should find its remainder modulo $10^9 + 7$.

For example, consider Sample Input 1 below. Here the list a could be split into [1|223|4], indicating that the connection between modules 1 and 2, and the connection between modules 4 and 5, are severed. Note that the connection between module 5 and 1 remains in this split. The second possible way in which a could be split is [12|2|34].

In Sample Input 2 below, the only possible way to split the list a into at least two non-empty parts is to separate the two modules. But then it is impossible to rearrange the parts to create the list b. Hence, the answer is 0.

Input

The first line of input contains a single integer n ($2 \leq n \leq 10^6$), the number of modules. The second line contains the n integers $a_1, \dots a_n$ ($0 \leq a_i \leq 10^9$). The third and final line contains the n integers $b_1, \dots, b_n$ ($0 \leq b_i \leq 10^9$).

The list b is guaranteed to be a rearrangement of the list a.

Output

Print one integer, the number of safe separations modulo $10^9 + 7$.

Example

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

Problem J. Joint Jog Jam

Source file name: Jogjam.c, Jogjam.cpp, Jogjam.java, Jogjam.py
Input: `Standard`
Output: `Standard`

Like so many good stories, this one begins with a claim that Kari is a faster runner than Ola, who of course challenges Kari to a run-down. Dubbed (rather ironically) Non-Competitive Pace Challenge, they want to see who can run the furthest in a certain amount of time t. Clearly they both choose to run in straight lines with constant speed.

Kari wrote an app to make sure that Ola does not cheat, but the app requires that their phones constantly communicate over Bluetooth.

After their run, Kari needs to ensure that they were never too far apart from each other at any time during the run. Write a program that computes the maximum distance between Kari and Ola at any point during the run.

Input

The input consists of a single line containing eight integers describing four points:

- the starting position of Kari,
- the starting position of Ola,
- the ending position of Kari, and
- the ending position of Ola,

in that order. Each point is given by two integers x and y ($0 \leq x, y \leq 10^4$), the coordinates of the point.

Output

Output the maximum distance between Kari and Ola during their run, with an absolute error of at most 10^{-6}.

Example

Input	Output
0 0 0 0 1 1 2 2	1.4142135624
0 0 0 1 0 2 2 1	2.2360679775
5 0 10 0 5 0 10 0	5

Problem K. Knot Knowledge

Source file name:	Knowledge.c, Knowledge.cpp, Knowledge.java, Knowledge.py
Input:	`Standard`
Output:	`Standard`

Sonja the scout is taking a test to see if she knows all the knots a scout is supposed to know. The Scout's Big Book of Knots has descriptions of 1 000 different knots, conveniently numbered from 1 to 1 000. For the test, Sonja needs to learn a specific set of n of these knots. After some intense studying, she has learned all except one of them, but she has forgotten which knot she does not yet know.

Given the list of knots Sonja needs to learn, and the ones she has learned so far, find the remaining knot to learn.

Input

The first line of input consists of an integer n ($2 \leq n \leq 50$), the number of knots Sonja needs to learn. This is followed by a line containing n distinct integers $x_1, \ldots, x_n$ ($1 \leq x_i \leq 1\,000$), the knots that Sonja needs to learn. Finally, the last line contains $n-1$ distinct integers $y_1, \ldots, y_{n-1}$ ($1 \leq y_i \leq 1\,000$), the knots that Sonja has learned so far. You may assume that each knot Sonja has learned is one of the n knots she was supposed to learn.

Output

Output the number of the remaining knot that Sonja needs to learn.

Example

Input	Output
4 1 2 4 3 4 2 3	1
4 10 101 999 1 1 999 101	10

Problem L. Locust Locus

Source file name:	Locus.c, Locus.cpp, Locus.java, Locus.py
Input:	`Standard`
Output:	`Standard`

There are two different species of *periodical cicadas* that only appear from hibernation every 13 and 17 years, respectively. Your old grandpa tells you that he saw them simultaneously back in '92. You start pondering how many years you have to wait until you see them again. You collect information about other pairs of periodical cicadas and when they were last observed to find out when the next simultaneous appearance is.

Given several different pairs of cicadas and their last simultaneous appearance, find the next year that one of the pairs reappears.

Input

The first line of input contains a single integer k ($1 \leq k \leq 99$), the number of pairs of periodical cicadas. Then follow k lines, each containing three integers y, c_1 and c_2 ($1800 \leq y \leq 2021$, $1 \leq c_1, c_2 \leq 99$), the year this pair was last observed and cycle lengths for the first and second species, respectively. You may assume that none of the k pairs reappears earlier than 2022.

Output

Output the first year a pair reappears.

Example

Input	Output
3 1992 13 17 1992 14 18 2001 5 7	2036
2 2020 2 3 2019 3 4	2026

Problem M. Marvelous Marathon

Source file name:	Marathon.c, Marathon.cpp, Marathon.java, Marathon.py
Input:	`Standard`
Output:	`Standard`

A marathon race is being planned in the beautiful countryside. The course will be somewhere along a long, bidirectional, road. The organizers want to determine exactly where along this road the race should be in order to maximize the experience for the runners, so that they get to enjoy as much beautiful scenery as possible and are distracted from their tired limbs. The scenery varies in beauty depending on where on the road you are, and also in which direction you are running. Because of this, the organizers are fine with having the runners make *up to* two U-turns, as long as no part of the road is used more than once in each direction.

We model the road of length m meters as a rectangular grid of size $2 \times m$, where each cell has a non-negative "beauty" value associated with it. The columns represent each meter of the road ordered from start to to end. The top row in a column represents the beauty for this part of the road when running in the direction towards the end of the road, and the bottom row in a column represents the beauty when running towards the start of the road. A race of length x is then some set of exactly x of the cells in the grid. Those x cells must form a path in the grid where no cell is visited more than once, we only move to the right or down from cells in the top row, and we only move to the left or up from cells in the bottom row. See Figure 3 for an example race. The "total beauty" of a race is the sum of the beauty values of the included cells.

The road is long, so rather than providing a list of all of the $2m$ beauty values, each side of the road is divided into a small number of segments, where the cells within a segment have some constant beauty value (and cells with beauty 0 are simply omitted).

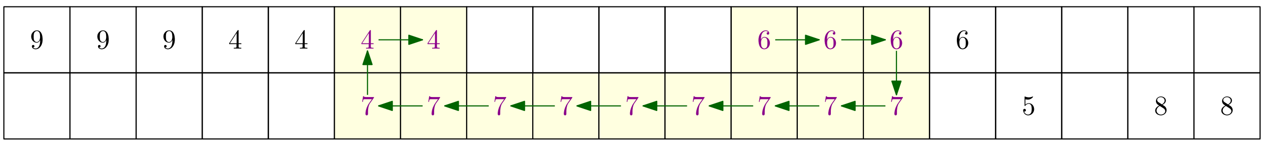


Illustration of Sample Input 1. The numbers in the cells indicate the beauty value for each meter of the road (with omitted values being 0). The highlighted cells and arrows mark the optimal race, involving two U-turns.

Input

The first line of input contains the three integers m, x and n ($1 \le m \le 10^9$, $1 \le x \le 2m$, $0 \le n \le 200$), the length of the road, the length of the race and the number of segments.

This is followed by n lines describing the segments. Each such line contains three integer a, b, v ($0 \le a, b \le m$, $1 \le v \le 10^9$, and $a \neq b$), describing a segment with endpoints a and b having beauty value v. If $a < b$, this is the segments of cells in the top row of the grid in the range $[a, b)$, and if $a > b$, this is the segments of cells in the bottom row of the grid in the range $[b, a)$.

The parts of the road that are not covered by any segments have beauty value 0. Each cell in the grid is covered at most once (that is, there are no overlapping segments in the same direction).

Output

Output the maximum possible total beauty the race can have.

Example

Input	Output
19 14 6 14 5 7 11 15 6 3 7 4 16 15 5 19 17 8 0 3 9	89
100000 42195 2 30000 60000 500000000 40000 10000 1000000000	35548500000000