

## Problem A. Alto Adaptation

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

It is always so frustrating when the song you want to sing is *just* outside the range of your beautiful alto singing voice... Thanks to music theory, a note outside your vocal range sounds similar to the intended note when you *transpose* it by one or multiple octaves up or down. However, just transposing a single note that is out of your range sounds weird: you would prefer to sing longer sequences of notes transposed by the same number of octaves.

To simplify some music theory, you represent the song as a sequence of numbers. Each number represents the *pitch* (or “height”) of a music note. The span of your vocal range is represented by two numbers, and you can sing any note between these two endpoints (inclusive). There are twelve notes in an octave, so transposing a note by some number of octaves up or down makes this note some multiple of 12 higher or lower than in the original song. You can switch between different transpositions at any point during the song. This divides the song into intervals of notes transposed by the same number of octaves. Your goal is to make the shortest such interval as long as possible. The original notes of the song can be treated as transposed by 0 octaves.



Concerto by Lorenzo Costa. Public domain on Wikimedia Commons

As an example, consider the third example case, visualized in Figure 1. The first three notes comfortably fit in your vocal range. The subsequent three notes need to be transposed two octaves down to fit your vocal range. The next three notes should be transposed one octave down. The final two notes need to be transposed one octave up. So, the shortest interval of notes in the same transposition consists of the two notes at the end of the song.

### Input

The input consists of:

- One line with three integers  $n$ ,  $\ell$ , and  $h$  ( $1 \leq n \leq 1000$ ,  $0 \leq \ell < h < 120$ ,  $\ell + 11 \leq h$ ), the number of notes, and the low and high end of your vocal range (inclusive).
- One line with  $n$  integers  $a$  ( $0 \leq a < 120$ ), the notes in the song that you want to sing.

### Output

Output the maximum length of the shortest interval of notes that should be transposed by the same number of octaves to make them fit inside your vocal range.

### Example

Input	Output
6 20 42 22 29 32 19 21 23	3
6 40 64 42 42 42 42 42 42	6
11 12 23 15 19 18 40 42 44 26 29 28 4 2	2

## Explanation

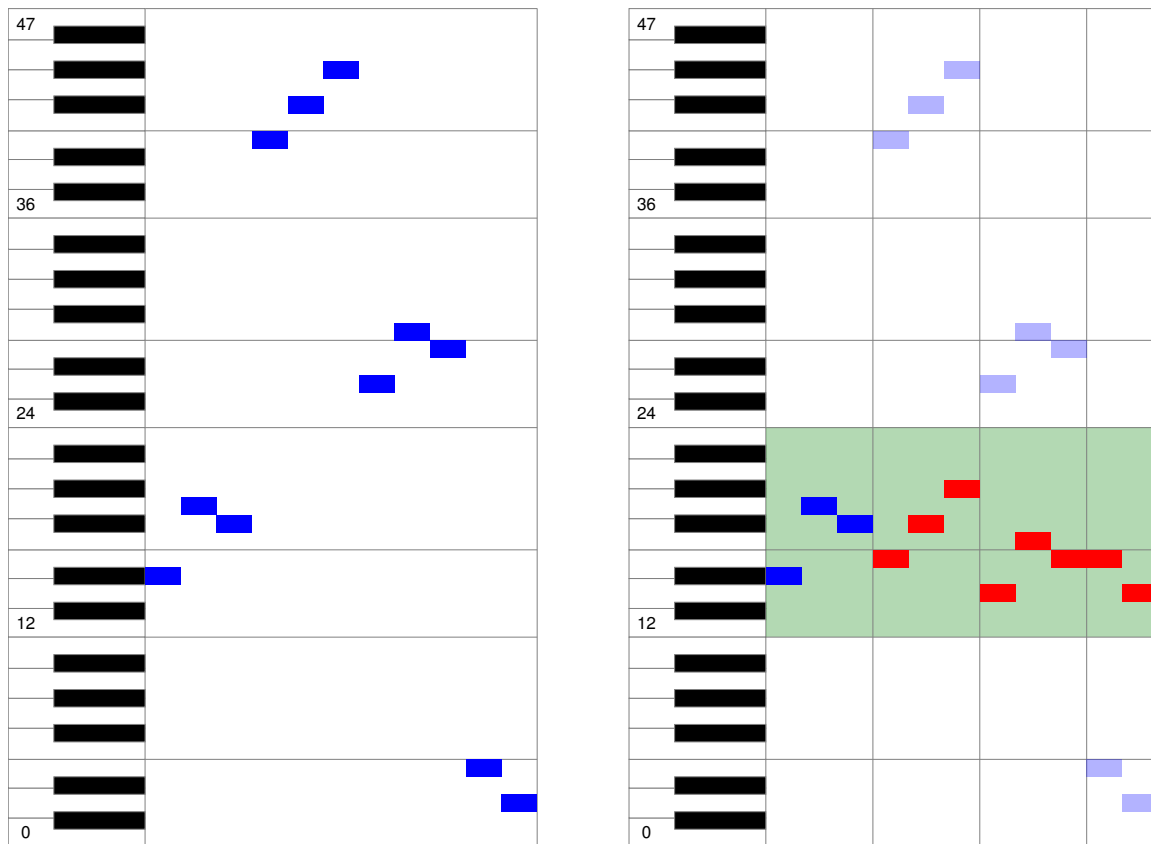


Figure 1. Illustration of the third example case. The green shaded background corresponds to your vocal range. The blue notes are the notes in the original song. The red notes are sung one or two octaves higher or lower than in the original song.

## Problem B. Bottle of New Port

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

At the Bottles and Port Company, you produce and bottle only the finest of vintages. To satisfy the desires of your highly exclusive clientele, it is necessary to study what happens to your wines once a new bottle is opened. When that happens, liquids in the bottle immediately start evaporating. The alcohol in the bottle evaporates at a different rate than the other liquids in the bottle, meaning that over time, the delicate balance of the port wine will be disturbed.

Your goal is to compute how the alcohol percentage changes in the days after opening a fresh bottle. By placing the opened bottle in an advanced chemical apparatus, you are able to measure how fast each liquid in the bottle evaporates.



A bottle of your favourite port, won at a programming competition.  
Picture by Reinier Schmiermann

### Input

The input consists of:

- One line with an integer  $d$  ( $0 \leq d \leq 10^6$ ), the number of days your bottle of new port has remained open.
- One line with two integers  $a$  and  $o$  ( $1 \leq a, o \leq 10^{12}$ ), the initial volume of alcohol in the bottle and the initial total volume of other liquids in the bottle, both in  $\mu\text{L}$ .
- One line with two integers  $\Delta_a$  and  $\Delta_o$  ( $0 \leq \Delta_a, \Delta_o \leq 10^{12}$ ), the evaporation rate of the alcohol and the evaporation rate of the other liquids in the bottle, both in  $\mu\text{L}/\text{day}$ .

It is guaranteed that the bottle is not empty after leaving it open for  $d$  days ( $d \cdot \Delta_a < a$  or  $d \cdot \Delta_o < o$ ).

### Output

Output the alcohol percentage after leaving the bottle open for  $d$  days.

Your answer should have an absolute or relative error of at most  $10^{-6}$ .

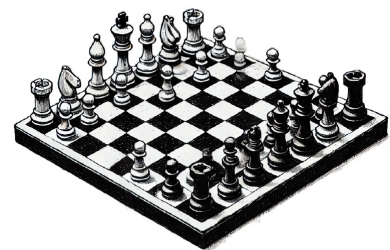
### Example

Input	Output
2 3 8 1 1	14.2857142857143
5 11 89 2 1	1.17647058823529
10 10 40 1 1	0
10 40 10 0 3	100

## Problem C. Chess

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

Two chess giants, Vito and Patrik, will play a game of chess this year in front of the iconic theater on Jane Street, finally proving who is the **greatest player of all time**. However, as standard chess has become boring for them, they decided to modify the rules of the game to make it more interesting. We will mention only the rules that are relevant to this task.



The chessboard will be a square matrix with  $N$  rows and  $N$  columns. Only the chess pieces **knight**, **rook**, and **queen** will be used. The pieces behave in the same way as in standard chess. A rook attacks a square if it is in the same row or column as that rook. A queen also attacks all squares in the same row and column, but in addition to that, it attacks all squares along the same diagonals. Knights attack squares that are two rows and one column away or vice versa. Examples of these moves can be seen in the explanations of the examples.

**Note:** Each piece also attacks the square on which it is placed. Additionally, pieces attack through other pieces, i.e., a piece attacks squares according to the rules stated above regardless of whether there is another piece between the square and the attacking piece.

Vito is preparing for the long-awaited showdown and needs your help. He has decided to practice his quick observation skills. He will do this by placing  $M$  chess pieces on the board and then determining all the squares that are attacked. Your task is to determine the number of attacked squares on the given board.

### Input

The first line contains positive integers  $N$  and  $M$  ( $1 \leq N \leq 200$ ,  $1 \leq M \leq N^2$ ).

In the next  $M$  lines, each line contains a single uppercase letter of the English alphabet, representing the type of piece, which can be one of 'N', 'R', and 'Q', corresponding to knight, rook, and queen, respectively, and integers  $r_i$  and  $c_i$  ( $1 \leq r_i, c_i \leq N$ ), representing the row and column where that piece is located.

At most one piece can be placed on any square of the board.

### Output

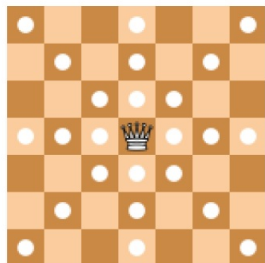
In the first and only line, you need to output the number of attacked squares on the board.

### Example

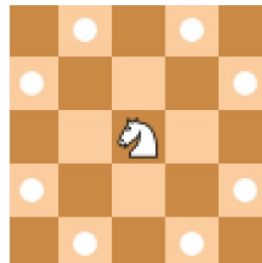
Input	Output
7 1 Q 4 4	25
5 1 N 3 3	9
6 3 R 1 4 Q 2 1 N 5 2	25

## Explanation

**Clarification of the first and second example:** In the sketches, examples are shown. All attacked squares are marked with a dot, except the one on which the piece is located.



(a) first example



(b) second example

## Problem D. Dralinpome

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

Your highly intelligent girlfriend, Jeannetta Ewell from Mississippi, loves to concatenate letters. For each of your anniversaries, you prepare a jar of letter jelly as a gift for Jeannetta. One of her favorite linguistic constrictions is that of a *palindrome*. In a palindrome, each letter reappears in place after a reorientation, reversing the string. For example, the word “racecar” is a palindrome, as it reads the same forwards and backwards.

After writing out “jeannetta” with her letter jelly, by arranging the letters in the order “natejetan”, she suddenly realized that she made a palindrome! This incidence lead to Jeannetta introducing her inventive definition of a *dralinpome*. In a dralinpome, the letters can be rearranged to form a palindrome.

Jeannetta now seeks to know all dralinpomes in the dictionary. However, due to her disinterestedness in programming, she would need to manually consider each word in the dictionary. While she has no idea what to do with an array or a queue, you and your teammates reparticipate in programming competitions each year. Therefore, you decide to help her and prepare a program that testifies of any given word whether it is a dralinpome or not.



The letter jelly that Jeannetta received last anniversary.

### Input

The input consists of:

- One line with a string  $s$  ( $1 \leq |s| \leq 10^5$ ), the word to check. The word consists of only English lowercase letters (a-z).

### Output

Output “yes” if the given word is a dralinpome, or “no” otherwise.

### Example

Input	Output
zoo	yes
yes	no
cacao	yes
multinomiaalcoefficient	yes
racecars	no

## Problem E. Encrypting

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

While Mr. Malnar was traveling by bus to Graz, he noticed that other passengers were looking at his phone and reading the messages he was sending to Patrik. Mr. Malnar decided to put an end to this. Therefore, together with Patrik, he developed a new way of encrypting their messages, which they call the *VolksWagen* cipher.

The received message can be imagined as a table of characters with 2 rows and  $n$  columns. Each letter spans all 2 rows and several columns, and they are separated by spaces. The appearance of letters in the message can be seen in the example test cases.

The letter 'v' is represented as follows:

```
\../
.\./
```

The letter 'w' is represented as follows:

```
\../\../
.\../\../
```

From now on, Patrik and Mr. Malnar will communicate exclusively using the letters 'v' and 'w' (without quotes). However, Mr. Malnar is having trouble reading these messages. He has asked you to help him decipher the message he received. Of course, he has not revealed to you how to decrypt their cipher.

### Input

The first line contains a positive integer  $N$  ( $1 \leq N \leq 1000$ ), the number of columns in the message.

In the next 2 rows, there are  $N$  characters each, representing one row of the message. It is guaranteed that empty columns will be exactly between two different letters, and there will be exactly one empty column between two letters. (*An empty column* is considered to be one that contains only the character '.'.)

### Output

In a single line, print the letters that appear in the message in order.

### Example

Input	Output
32 \../\../\../\../\../\../\../\../ .\../.\../.\../.\../.\../.\../.\../.	VWVWV
27 \../\../\../\../\../\../\../ .\../.\../.\../.\../.\../.\../.	WVWV

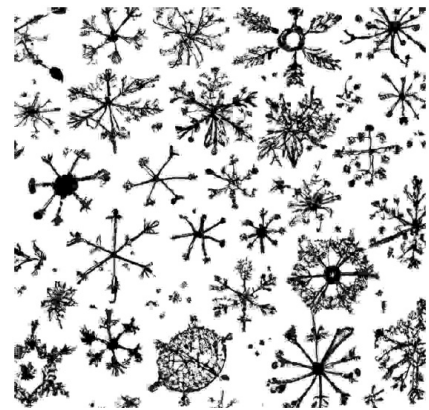


## Problem F. Flakes

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

Lana likes to draw specific snowflakes. A snowflake of size  $x$  is defined as follows:

- The center of the snowflake is the character '+'.
- Above and below the character '+' there is a sequence of  $x$  characters '|'.
- To the left and right of the character '+' there is a sequence of  $x$  characters '-'.
- On the diagonal from the upper left corner to the center and from the center to the lower right corner of the snowflake there is a sequence of  $x$  characters '\'.
- On the diagonal from the upper right corner to the center and from the center to the lower left corner of the snowflake there is a sequence of  $x$  characters '/'.



Sometimes Lana connects several snowflakes, but even then the size of each snowflake is determined separately, regardless of whether the snowflakes share some characters.

```

\ | /
-+-
/ \

\ . | . / . .
. \ | / . . .
--+-
. / | \ . . .
/ . | . \ | /
. . . . -+-
. . . . / | \

\ . | . / .
. \ | / . .
--+-
. / | \ . .
/ . . . \ .

```

On the left is an example of a snowflake of size 1.

In the middle is an example of connected snowflakes, the left one of size 2 and the right one of size 1.

On the right is an example of a snowflake of size 1. It is missing one character '|' to be of size 2.

Lana is currently drawing snowflakes on a piece of paper of size  $n \times m$ . However, she got a bit confused and did not draw all the snowflakes completely in accordance with her usual snowflake shapes. Namely, some snowflakes are missing some characters, so their size is equal to the smallest length of the corresponding character sequence from the center in one of the eight directions. Moreover, she drew some characters that are not part of any snowflake.

Can you help Lana determine the size of the largest snowflake in the drawing?

### Input

The first line contains two integers  $n$  and  $m$  ( $1 \leq n, m \leq 50$ ), the size of the drawing.

In each of the following  $n$  lines there are  $m$  characters describing the drawing.

The characters that can appear in the drawing are '+', '-', '\', '|', '/' and '.'. The ASCII values of these characters are 43, 45, 92, 124, 47 and 46 respectively.

## Output

In the first and only line you should output the size of the largest snowflake in the drawing.

## Example

Input	Output
<pre> 5 6 \.\ /. ---+--- ./ \.. ./ . \ /.. .. </pre>	1
<pre> 7 7 \. ./.. .\ /... --+---. ./ \... /. .\\ / ....-+- ..../ \ </pre>	2
<pre> 7 7 \ / \ / -+- -+- / \ / \ ---+--- \ / \ / -+- -+- / \ / \ </pre>	1

## Explanation

**Clarification of the first example:** Only one snowflake is drawn.

In the directions up-left, up and up-right from the center of the snowflake there are sequences of length 1.  
In the directions right and down-right from the center of the snowflake there are sequences of length 2.  
In the directions left, down-left and down from the center of the snowflake there are sequences of length 3.

Therefore, the size of the snowflake is 1.

**Clarification of the second example:** Two connected snowflakes are drawn, the left one of size 2 and the right one of size 1.

## Problem G. Genealogy Gumbo

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

You just got a new job as a Byzantine Ancestry Project Coordinator where you try to piece together ancient family trees of people long dead. Because you can no longer ask them any questions, you have to rely on documents left from that era where names are written patrilineally: ancestry is only traced by the name of someone's father. One example of such name would be "Basil, son of Alexios". You are trying to prove your research hypothesis: the entire Byzantine population is descended from just a single common ancestor.

In order to prove this hypothesis, you need to determine whether such a family tree exists: find one possible family tree with a single root. You cannot create imaginary people or construct familial relations for which there is no historical evidence. One father can have multiple children, but one child cannot have multiple fathers. It is possible that different people have the same name.



A reconstruction of the family tree you have found. CC BY-SA 4.0 by Shakko on Wikimedia

### Input

The input consists of:

- One line with an integer  $n$  ( $1 \leq n \leq 10^5$ ), the number of parental relations specified.
- $n$  lines, each with a name of the form " $A$ , son of  $B$ ".

Each name has at most 20 characters and consists of one English uppercase letter (A-Z), followed by only English lowercase letters (a-z).

### Output

If it is possible for the given people to have a single common ancestor, output "possible". Otherwise, output "impossible".



## Example

Input	Output
3 Jacob, son of Isaac Isaac, son of Abraham Ishmael, son of Abraham	possible
2 Basil, son of Alexios Procopius, son of Constantine	impossible
3 Alvin, son of Bert Bert, son of Charles Charles, son of Alvin	possible
2 James, son of Harry Harry, son of James	possible
3 Albert, son of Brent Cody, son of Brent Brent, son of Daniel	possible
3 Aaron, son of Aaron Aaron, son of Bobby Bobby, son of Chris	possible
3 Chris, son of Bobby Bobby, son of Aaron Aaron, son of Aaron	possible

## Problem H. Hidden Sequence

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

Three players are having a game night, playing a long sequence of different games. In each game, one of the three players wins. In order to keep track of the order of winners, each of the three players decides to keep a list of all winners in order. However, due to the dopamine rush as the result of winning a game, everyone always forgets to write themselves on their own list of winners. Thus, the list of player 1 only contains the order in which players 2 and 3 won the different games, the list of player 2 only contains the order in which players 1 and 3 won, and the list of player 3 only contains the order in which players 1 and 2 won.

Given these three lists, determine the actual order in which all three players won their games.



Three-player chess; one of the games of the evening. Public domain by Dr Jacek Filek on Wikimedia Commons

### Input

The input consists of:

- Three lines, the  $i$ th of which contains a string  $s_i$  ( $1 \leq |s_i| \leq 10^5$ ), the order in which the two players different from  $i$  won their games.

It is guaranteed that it is always possible to determine the unique order in which all three players won their games.

### Output

Output a single string containing the order in which all three players won their games.

### Example

Input	Output
2 1 21	21
23 13 12	123
23232323 11331133 12121212	121323121323

## Problem I. Item Selection

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

You are browsing a website that lists items for sale. The website has a paging UI that displays a fixed number of items per page, one page at a time.

For example, if there are 55 items and the page displays exactly 20 at a time, then there are 3 pages in total. Items 1 through 20 are on page 1, items 21 through 40 are on page 2, and items 41 through 55 are on page 3.

You may navigate and select items using these UI elements:

- A checkbox for every item on the current page. After you click a checkbox, a selected item becomes unselected, and an unselected item becomes selected. You cannot click a checkbox for an item that is not on the current page.
- A “Select All” button. All unselected items on the current page become selected after you click this button.
- A “Deselect All” button. All selected items on the current page become unselected after you click this button.
- A “Next Page” button. Clicking it navigates to the next page and increments the current page number by one. This button is disabled on the last page.
- A “Previous Page” button. Clicking it navigates to the previous page and decrements the current page number by one. This button is disabled on the first page.

The website has pre-selected some items for you based on its machine learning recommendation algorithm. The recommendation may or may not work for you. You know exactly the items that you want to purchase, which may differ from the pre-selected items. What is the minimum number of checkbox and button clicks required to select exactly the items you actually want?

### Input

The first line of input has five integers  $n$ ,  $m$  ( $1 \leq m \leq n \leq 10^3$ ),  $s$  ( $1 \leq s \leq \lceil \frac{n}{m} \rceil$ ),  $p$ ,  $q$  ( $0 \leq p, q \leq n$ ), where:

- $n$  is the number of items. The items have item numbers from 1 to  $n$ .
- $m$  is the fixed number of items displayed per page.
- $s$  is the number of the page currently displayed.
- $p$  is the number of preselected items.
- $q$  is the number of items you want.

Each of the next  $p$  lines contains an integer  $i$  ( $1 \leq i \leq n$ ). These are the item numbers of the preselected items. These  $p$  items are distinct and are listed in increasing order. It is possible that the website has pre-selected none of the items ( $p = 0$ ), in which case the input has no lines for pre-selected items.

Each of the next  $q$  lines contains an integer  $j$  ( $1 \leq j \leq n$ ). These are the item numbers of the items you want to buy. These  $q$  items are distinct and are listed in increasing order. It is possible that you want to buy none of the items ( $q = 0$ ), in which case the input has no lines for items you want.



## Output

Output a single integer, which is the minimum number of checkbox and button clicks required to select exactly the items you want.

## Example

Input	Output
11 4 1 5 5 1 4 9 10 11 1 3 6 7 8	7



## Problem J. Judge Meetings

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

The judges for the UCF Local Programming Contest have very busy summers. In order to have effective meetings, at least 3 judges have to be present. Since you enjoy Locals and want the contest to go on, given their vacation schedules, help the judges figure out how many days over the summer they could have meetings.

Given the list of days that each judge is away on vacation, determine the number of days that at least 3 judges could meet (are not away on vacation).

### Input

The first input line contains two integers:  $n$  ( $3 \leq n \leq 10$ ), indicating the number of judges, and  $m$  ( $30 \leq m \leq 120$ ), indicating the number of days for summer vacation. These days are numbered from 1 to  $m$ . Each of the next  $n$  input lines will contain information about each judge's vacations. The first item on each of these input lines is an integer,  $v$  ( $1 \leq v \leq 10$ ), indicating the number of vacations that judge has taken. This will be followed by  $v$  pairs of integers,  $s$  and  $e$ , indicating that the judge took a vacation starting on day  $s$  of the summer and ending on day  $e$  of the summer. For all pairs of  $s$  and  $e$ ,  $1 \leq s \leq e \leq m$ . For each judge, no vacations will overlap but it's possible that a judge could end a vacation on one day and leave for another vacation on the following day, meaning that judge is not available for a meeting at all between those two vacations. The vacations for a judge are not necessarily listed in order, i.e., the vacations for a judge are listed in a random order.

### Output

Print the number of days during the summer that at least 3 judges are available to meet (not on vacation).

### Example

Input	Output
5 90 2 30 40 50 60 1 1 45 1 40 90 2 33 40 41 50 1 45 90	62
3 60 2 30 39 1 10 1 1 20 1 45 60	14

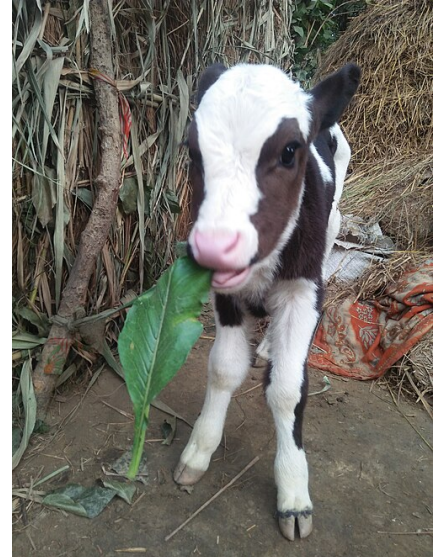
## Problem K. Koehandel

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

You are playing a game of *Betting And Purchasing Cows* (BAPC), and Old MacDonald wants to buy one of your cows using a bid-off. This works as follows.

He bets some number of BAPC coins, which he puts in a cup that he places upside down on the table. You then openly bet some number of your own coins. The cup is removed to reveal MacDonald's bet. The bets are exchanged: you receive the bet from MacDonald and he receives your bet. The player that bet the most BAPC coins receives one cow from the opponent, regardless of who started the bidding. In case of a tie, no cows are exchanged.

You love the cows in the game, so your primary goal is to have the most possible cows after the trade. Your secondary goal is to have the highest possible number of coins after the trade. You pay close attention when Old MacDonald is secretly putting their coins in the cup, and by listening very carefully, you have determined the exact number of coins that he put in the cup. Given this information and the number of coins you have, how many coins should you bet?



This cute cow you would not to want to part with for anything.  
CC BY-SA 4.0 by Ashish Rajvanshi on Wikimedia Commons

### Input

The input consists of:

- One line with two integers  $c$  and  $n$  ( $0 \leq c, n \leq 10^9$ ), the number of BAPC coins in Old MacDonald's cup and the number of coins that you have.

### Output

Output the number of coins that you should bet, keeping your goals in mind.

### Example

Input	Output
6 19	7
42 37	0
25 25	25

## Problem L. Luminosity

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

In the heart of the Glass Valley lies a mysterious star temple, a place known for its collection of magical crystals that shine like stars. Each crystal holds special power and emits a radiant glow that illuminates the entire valley, as long as it remains untouched.

The temple guardian's nightly task is to touch **ONLY** the crystals located within the specified ranges of the valley's residents, while honoring all of their requirements. Each resident's request tells the guardian which range of crystals must **NOT** stop shining **BEFORE** their bedtime, as they fear the darkness.

The guardian starts his journey at the temple entrance and must carefully coordinate their movements to dim the crystals such that they stop shining at the correct moment. The crystals are arranged in a line, spaced one meter apart from each other (the first crystal is one meter away from the entrance). The guardian can move at a speed of one meter per second and may stop when needed. The time it takes for the guardian to touch and dim a crystal is negligible. Given the residents' requests, the temple guardian wants to know the minimum number of seconds required to fulfill all requests (the guardian does not need to return to the starting position).



### Input

In the first line is an integer  $N$  ( $1 \leq N \leq 5000$ ), number of residents' requests.

In the next  $N$  lines are integers  $l_i, r_i, t_i$  ( $1 \leq l_i \leq r_i \leq 10^{18}$ ,  $1 \leq t_i \leq 10^{18}$ ), representing the left and right bounds of the crystal range and the resident's bedtime, respectively.

### Output

In the first and only line output the minimum time in seconds required for the guardian to fulfill all the requests.

### Example

Input	Output
3 1 1 1 3 3 5 5 5 3	7
3 1 2 1 1 1 5 1 3 4	6
3 6 6 6 8 8 7 9 9 9	9



## Explanation

**Clarification of the second example:** The temple guardian will first take 3 seconds to reach the 3rd crystal. Then, he will wait for 1 second and dim the 3rd crystal. After that, he will take 1 second to go to the 2nd crystal and dim it. Finally, he will take 1 second to reach the 1st crystal and dim it. In total, his journey will last 6 seconds, which is the minimum time required to fulfill all the requests.



## Problem M. Most Scenic Cycle

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

The government of the Independent Country of Problem Creators (ICPC) finally saved enough money to construct high speed rail infrastructure. The new rail system has  $V$  stations and  $E$  bidirectional direct railway lines that each connect two stations together. The head of ICPC Rail Infrastructure Planning, Skib E. Dee, has seen enough programming problems about tree-topology transportation networks in other countries to know that such a network would be a recipe for disaster: a single broken railway line would split the network into disconnected pieces and disrupt travel for days. Instead, Dee decided that the ICPC rail network will be **robustly connected**: every pair of stations  $s_1, s_2$  must be connected by at least two paths which do not share any direct railway lines, and do not share any railway stations other than  $s_1$  and  $s_2$  themselves.

Of course, ICPC cannot afford to build too many redundant railway lines. To balance efficiency and resiliency, Dee has also designed the network to be **regionally connected**. A cycle is a nonempty path from a station to itself which doesn't repeat any railway station (apart from the first station, which must repeat exactly once as the last station of the cycle). In order for the network to be regionally connected, there must exist a set  $F$  of  $E - V + 1$  regional cycles satisfying three properties:

- every direct railway line in the transportation network belongs to at least one regional cycle;
- if two regional cycles share any direct railway lines, then all railway lines and stations shared by those cycles lie along a connected path;
- for each subset  $f$  of  $F$ , at most  $|f| - 1$  pairs of regional cycles in  $f$  share any direct railway lines.

To promote the new high speed rail, Dee needs to create a timelapse video of a train travelling around some cycle in the railway network. Each direct railway line has a (possibly negative) scenic value representing how nice the view out the train window is along that line. Dee wants to send the train around whichever cycle maximizes the sum of scenic values of the direct railway lines on the cycle – compute this maximum possible sum. (The most scenic cycle that Dee is looking for does **not** have to be a regional cycle.) In order to ensure this cycle is not boring, it must traverse at least two direct railway lines, and must not repeat any railway station (apart from the first station, which must repeat exactly once as the last station of the cycle).

### Input

The first line of input contains two space-separated integers  $V$  ( $2 \leq V \leq 2 \cdot 10^5$ ) and  $E$  ( $V \leq E \leq 4 \cdot 10^5$ ), the number of stations and direct railway lines in the rail network, respectively.

The next  $E$  lines of input describe the direct railway lines. Each line contains three space-separated integers  $a, b$ , and  $s$  ( $1 \leq a, b \leq V$ ;  $-10^9 \leq s \leq 10^9$ ), signifying that a direct railway line exists between stations  $a$  and  $b$  with scenic value  $s$ . No direct railway line connects a station to itself, but **multiple direct railway lines might exist between the same two stations**. It is guaranteed that the input graph will be both *robustly connected* and *regionally connected*.

### Output

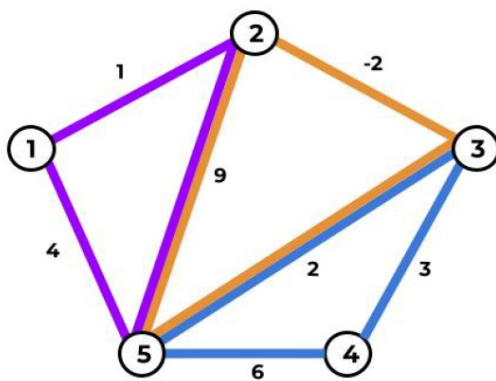
Print the sum of scenic values around the cycle in the railway network that maximizes this sum.

## Example

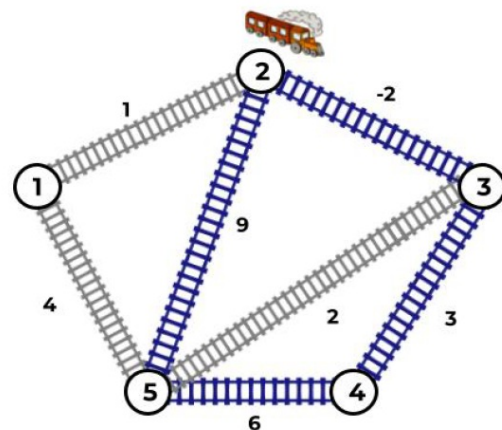
Input	Output
6 9 1 2 9 2 3 9 3 4 9 3 4 -9 4 1 9 1 5 1 5 6 1 6 2 1 3 4 8	36
5 7 1 2 1 2 3 -2 3 4 3 4 5 6 5 1 4 5 3 2 2 5 9	16

## Explanation

For the railway network in Example Input 2, one possible choice for the regional cycles in  $F$  are  $1 \rightarrow 2 \rightarrow 5 \rightarrow 1$ ,  $2 \rightarrow 5 \rightarrow 3 \rightarrow 2$ , and  $3 \rightarrow 4 \rightarrow 5 \rightarrow 3$  (pictured on the left). The most scenic cycle (pictured on the right, in blue) has a scenic value sum of  $9 + 6 + 3 - 2 = 16$ .



Sample 2 Regional Cycles



Sample 2 Most Scenic Cycle