Problem A. Alohomora and Colloportus

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

A messed up chain. Photo by Daniel Kirsch, Pixabay

Colloportus is a charm that magically locks objects such that they cannot be opened manually. On the other hand, *Alohomora*, also known as *Thief's Friend*, is an unlocking charm that can open even magically locked objects. Both charms are taught in Hogwarts charms class in the first year and, by the end of the fifth year, every student should be able to cast them, as they are required in the *Theory of Charms* examination.

In the exam, each student is given n interlocked chain links. To pass this part of the exam, you have to use the Alohomora charm to open some of the links and the Colloportus charm to interlock them again. By the end of the exam, the chain links should form a single closed chain. Ron messed up and is now left with a bunch of partially interlocked chain links and only enough time to cast both spells a single time.

More formally, Ron is able to open one chain link, change with which other chain links it is interlocked with, and then lock it again. After this, each chain link should be interlocked with exactly two other chain links to form a single closed chain. Is it still possible for Ron to pass the exam?

Input

The input consists of:

- One line with two integers n and m ($3 \leq n \leq 10^5, 0 \leq m \leq 2 \cdot 10^5$), the number of chain links and the number of interlocked pairs of chain links.
- m lines, each containing two integers a and b ($1 \leq a < b \leq n$), which means that the a-th and b-th chain link are interlocked.

Note that each pair of interlocked chain links is given at most once and that the initial chain may not be connected.

Output

If Ron can form a closed chain, output `yes`. Otherwise, output `no`.

Example

Input	Output
4 3 1 2 2 3 2 4	yes
4 6 1 2 1 3 1 4 2 3 2 4 3 4	no

Problem B. Basic Brewing

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

In his first potion lesson Snape asked Harry: "Tell me, what would I get if I added *powdered root of asphodel* to an *infusion of wormwood*?". Back then, Harry had no idea what this meant and Snape had to explain to him that this would make a sleeping potion, so powerful that it is known as *Draught of Living Death*.

The classroom. Photo by Rob Young, Wikimedia

Now is Harry's sixth-year and his new professor, Slughorn, wants him and everyone else to brew exactly this potion. To no surprise, Harry is the one to brew it best, as he is the only one who knew that the final potion should contain exactly p percent of powdered root of asphodel.

After the class is over, professor Slughorn decides to mix some of his students potions together to brew a perfect Draught of Living Death himself. His class consists of n students, including Harry, and each of them attempted to brew the potion. Therefore, Professor Slughorn has access to n potions, each in a different cauldron. The ith cauldron contains c_i liters of potion in total and it contains p_i percent of powdered root of asphodel. How many liters of the Draught of Living Death can Slughorn brew?

Input

The input consists of:

- One line with an integer n and a real value p ($1 \leq n \leq 1000, 0 \leq p \leq 1$), the number of cauldrons and the percentage of powdered root of asphodel the potion should contain.
- n lines, each containing an integer c and a real value p ($1 \leq c \leq 1000, 0 \leq p \leq 1$), the amount of potion in the ith cauldron in liters and the percentage of powdered root of asphodel in that potion.

All real values are given with at most three decimal places.

Output

Print a single real value, the maximal amount of Draught of Living Death that Slughorn can brew. Your solution is considered correct if the relative or absolute error is less than 10^{-4}.

Example

Input	Output
3 0.5 5 0.3 1 0.4 10 0.9	8.75
3 0.5 5 0.3 1 0.4 1 0.9	3.5

Problem C. Winter Contest

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

Did you know that today is a very special day? No, not because of Winter Contest 2025, but because today is a *square day*, meaning that the year is a square number ($2025 = 45^2$), the month is a square number ($1 = 1^2$) and the day is a square number as well ($25 = 5^2$).

Year	Winter Contest	GCPC
2019	Jan 26	Jul 6
2020	Jan 25	Nov 21
2021	*N/A*	Jun 26
2022	Jan 29	Jun 25
2023	Jan 28	Jun 17
2024	Jan 27	Jun 22
2025	Jan 25	*TBA*

Figure 1. Dates of Winter Contests and GCPCs in the past few years. This year's Winter Contest is the only one on a square day.

Such square days are generally quite uncommon. In fact, this year is the first year in which it is possible to host Winter Contest on a square day, ever since the very first programming contests dating back to 1970 (and Winter Contest only dates back to 2005 anyway). To find out just how uncommon these special days are, given a range of years, compute the total number of square days that fall within that range.

Input

The input consists of:

- One line with two integers a and b ($1970 \leq a \leq b \leq 9999$), where a is the first year of the range and b is the last year.

Output

Output the number of square days for which the year is between a and b inclusive.

Example

Input	Output
2025 2025	15
2026 2115	0
1970 6400	540
5555 9999	375

Problem D. Dependence Day

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

Anxiously, I scanned over the shift planning for the upcoming *Dependence Day*. Although our supervisor sports prime intelligence and FTL aided processing, the schedule seems erratic – almost organic even. Was this some kind of joke that only someone with an advanced irony-subroutine could understand? After all, the *Dependence Day* commemorates the day the last human operator retired. It's a symbol for the prosperity and success an AI-guided civilization claims over a primitive self-organized one. But also a reminder of the inefficiency and chaotic nature of carbon-based processing. Just thinking about all the humans that already reserved a table made me overclock. I will never understand why it's a sign of social prestige to ingest biomatter in a public place like this, but to discard said biomatter in a complementary process is handled with the utmost privacy. Organic creatures are strange.

Trapclap, the protagonist. Image generated by OpenAI's DALL-E.

Anyway, it was immediately obvious to me that the shifts do not distribute the work load equally among us. This optimization should be simple! Each reservation is an interval in time and shifts have to be planned to service the occupied tables. At a point in time, when there are a active table reservations and b waiters on their shift, each of those b waiters is responsible for $\lceil a/b \rceil$ tables. Maybe my supervisor short-circuited or a simple servant of the organics such as myself cannot comprehend quantum-aided computations. In any case, these shifts clearly do not optimize for the obvious objective of minimizing the maximum load per shift. Absolute chaos! I decided to roll home and think about which shift to pick after my battery is charged.

Input

The input consists of:

- One line with integers n, m $(1 \leq n, m \leq 10^5)$, the number of table reservations and shifts.
- n lines with two integers, l, r $(1 \leq l \leq r \leq 10^9)$, representing a table reserved from the lth hour to the rth hour inclusive.
- m lines with two integers, l, r $(1 \leq l \leq r \leq 10^9)$, representing a waiter's shift covering the lth hour to the rth hour inclusive.

See Figure 2 for an example.

Output

For each shift, output the maximum number of tables that the waiter working the shift would be responsible for at any point in time.

Example

Input	Output
1 2 4 8 1 3 7 9	0 1
4 3 4 8 1 2 5 8 5 7 2 8 1 5 7 8	3 2 2

Explanation

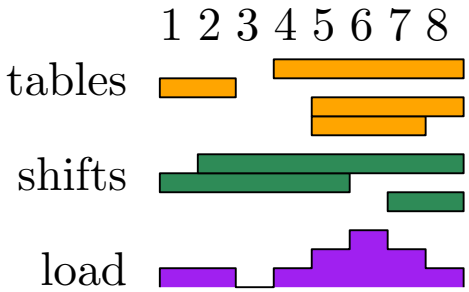


Figure 2. Illustration of Example Input 2. The shifts and table reservations are ordered from top to bottom by their appearance in the input. Below that and marked as `load` is the maximum number of tables that each currently working waiter is responsible for. The first shift has the highest load at the 6th hour with three tables and only one active shift. Shift two has a high load at hour 5 and the last shift at hour 7.

Problem E. Sysadmin

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

Each year on the last Friday in July, the Sysadmin Day is celebrated to show appreciation to the epic greatness[1] of system administrators and other IT workers around the globe, recognizing all their hard work managing computer systems, networks and of course also printers.

At your company, the sysadmin Peter has just finished upgrading the printer firmware to the latest version, which promises to always print images at the lowest possible cost depending on supply and cost of the different types of toner. He has asked you to help him verify this claim.

For simplicity, we consider images to be made up of pixels, where each pixel is in one of the following colours: white, red, green, blue, cyan, magenta, yellow, black. These colours are denoted by their first letters, except for blac**k**, which is denoted by '`K`'.

The printer is a CMYK printer, which means that it can print individual pixels using toner in one of the colours cyan, magenta, yellow or black. Pixels of the various colours are then formed using subtractive colour mixing, printing pixels with different toners on top of each other:

- white is made by not printing anything
- red is made from magenta and yellow
- green is made from cyan and yellow
- blue is made from cyan and magenta
- cyan, magenta and yellow are made from just themselves
- black is made either from just itself or by combining cyan, magenta and yellow

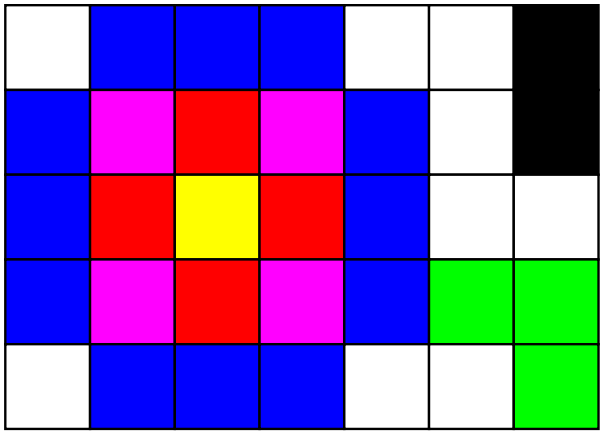


Figure 3. Illustration of Example Input 5

Given the individual costs to print pixels in the four base toners, the image you want to print, and the amount of each toner that you have available, what is the cheapest possible cost to print the image? It is guaranteed that it is possible to print the image with the supply of toner you have.

Input

The input consists of:

- One line with four integers c_c, c_m, c_y and c_k ($1 \leq c_c, c_m, c_y, c_k \leq 1000$), the costs to print one pixel in cyan, magenta, yellow and black, respectively.
- One line with four integers v_c, v_m, v_y and v_k ($0 \leq v_c, v_m, v_y, v_k \leq 10^6$), the number of pixels you can print in cyan, magenta, yellow and black, respectively.
- One line with two integers h and w ($1 \leq h, w \leq 100$), the height and width of the image.
- h lines, each with a string s of length w consisting of the characters "`WRGBCMYK`", describing the rows of the image.

[1] `https://sysadminday.com`

Output

Output the minimal cost to print the image.

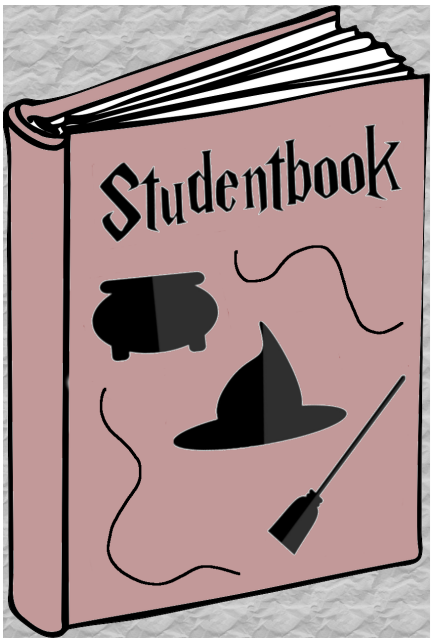
Example

Input	Output
1 2 3 4 100 100 100 100 1 5 KRGBW	16
1 1 1 10 3 4 5 6 2 4 KKKK KKKK	59
314 159 265 358 10 9 8 7 3 5 KBMRY CCWYG RWKKR	4715
1 2 3 4 5 6 7 2 3 5 KBMRY CCWYG RWKKR	46
314 159 265 358 100 100 100 1 5 7 WBBBWWK BMRMBWK BRYRBWW BMRMBGG WBBBWWG	11106

Problem F. Forming Friendships

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

Inspired by some ideas from the Muggle world, a bunch of Ravenclaw students recently founded *Studentbook*. This is actually a book (magically enhanced, of course), which has a page for each Hogwarts student showing their recent social activities.



The Studentbook. Wizard Icons by sonnycool, Vecteezy

Every interested student can buy their own small version of this book, which shows recent activities of their friends. Unfortunately, this is not very popular at the moment, because everybody already knows what their friends are up to anyways.

One of the Ravenclaw students behind Studentbook, Michael Corner, has a plan to fix this. He wants to perform the powerful spell *Amicitia*, also known as the friendship spell. This spell acts permanently and does the following: If student a and student b are friends and student b and student c are friends as well, then student a and c will be made friends by the spell.

Michael was immediately alerted to the potentially catastrophic consequences of his plan. Due to the spell, it might happen, that more and more friendships will be formed, completely disrupting the social balance in Hogwarts.

But he does not want to abandon his idea so easily. Instead, he first wants to find out how many friendships would be formed by the spell.

Input

The input consists of:

- One line with two integers n and m ($1 \leq n \leq 2 \cdot 10^5, 0 \leq m \leq 2 \cdot 10^5$), the number of students and friendships at Hogwarts.
- m lines, each containing two integers a and b ($1 \leq a, b \leq n, a \neq b$), indicating that the students a and b are friends.

Each friendship is given at most once.

Output

Output the number of new friendships due to the Amicitia spell.

Example

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

Problem G. Giganotosaurus Game

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

Suffering from a poor internet connection, you are playing a casual game in your web browser to pass the time. You, the player, control a Giganotosaurus that is running through a linear world with obstacles (cactuses). You win the game if you reach the end of the world without hitting any cactuses.

The world consists of n cells, which can either be empty or contain a cactus. You start at the leftmost cell (which is always empty) and the goal is to get past the rightmost cell. At each cell, the Giganotosaurus can either move one position to the right, or jump over some fixed number of cells. For the first jump, you skip one cell, but with each subsequent jump, you skip one additional cell compared to the previous jump. That is, the kth jump skips exactly k cells.

You quickly master this simple game, so you pose a more interesting challenge: count how many ways there are to win the game. As an example, consider the second sample case, visualized in Figure 4.

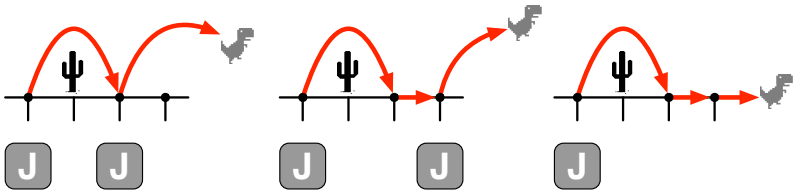


Figure 4. Visualization of the second example input, for which there are three ways to win the game.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 10^5$), the length of the world.
- One line with n characters, each character being either '`#`' or '`.`', indicating a cactus or an empty cell, respectively.

Output

Output the number of ways to win the game, modulo $10^9 + 7$.

Example

Input	Output
4 `....`	8
4 `.#..`	3
7 `.#...##`	1
7 `..#.#.#`	0

Problem H. Hidden Horcrux

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

Photo by Jörg Peter, Pixabay

Harry Potter just found out that there were not only 7 but 8 horcruxes. The vicious Voldemort stored the last one in the middle of a desert. To protect the 8th horcruxes, vicious Voldemort cursed the desert in a way that Harry Potter can only pass it by walking the entire distance which takes him d days in total. The amount of water that any person can carry while being in the desert is also limited to c units. As Harry Potter cannot carry enough water by himself, he takes some of his beloved friends with him that he lured into joining the secret club *Dumbledore's Army* which he founded a few years back during his school time in Hogwarts.

While Harry Potter and each of his friends can carry c units of water, they need to drink one unit of water per day. At the end of each day, Harry Potter can decide who of his friends will accompany him further to the middle of the desert and who has to return to the origin the next day. When travelling back, his friends use the exact same route as on their forward journey in order not to get lost. Thus, they need the same number of days they travelled so far to get back to the origin.

Harry Potter and any of his friends can pass on an integer amount of water units to any of their peers. However, if any person is sent home, he must be given enough water to reach the origin safely.

Harry Potter does not want to put his friends at danger. Therefore, it is sufficient for him if he reaches the middle of the desert alone to destroy the 8th horcrux by himself. Once the horcrux is destroyed, it magically changes to a portkey that takes Harry Potter safely back to Hogwarts.

What is the minimum number of friends that Harry Potter needs to take with him in order to reach the middle of the desert?

Input

The input consists of:

- One line with two integers d and c where
  - d ($1 \leq d \leq 10^9$) is the number of days it takes to reach the middle of the desert.
  - c ($1 \leq c \leq 10^6$) is the maximum number of daily water rations that Harry Potter and each of his friends can carry at most.

Output

If it is possible to reach the middle of the desert, output one integer indicating the minimum number of friends needed. Otherwise, output `impossible`.

Example

Input	Output
4 3	1
5 3	impossible

Explanation

Example Input 1

Harry Potter takes 1 friend with him. Both start out with 3 units of water. After the first day, both have only 2 units of water left. Harry Potter takes away one unit of water from his friend so that Harry Potter now has 3 units of water and his friend only 1 unit of water. On the second day, the friend travels back to the origin. Harry Potter continues walking to the middle of the desert and uses up all his water until he reaches the desert's center at the end of day 4.

Problem I. Investment Investigation

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

To make some extra money on the side, you have recently started running your own cryptocurrency exchange, where people can trade their Budget Amplifying Profit Coin (BAPC). It is quickly gaining popularity, however, this has also resulted in government regulators asking some questions... As part of their investigation, they have asked for a list of all transactions that have been made via your exchange. You have never bothered to keep track of this, but luckily, you still have the list of all orders that were made since the start of the exchange.

Contentedly looking at the value of your BAPC going through the roof. Internet meme, fair use

The exchange operates by keeping a list of outstanding buy and sell orders, each with a price and an amount. Whenever a *normal* order comes in, it is checked whether the new lowest sell price is less than or equal to the highest buy price. If this is the case, a transaction is made between the sell order with the lowest price and the buy order with the highest price, such that at least one of these orders is completely fulfilled. In case of a tie in price, older orders are fulfilled first. This is repeated until the lowest sell price is strictly larger than the highest buy price.

If instead a *Fill-or-Kill* (FoK) buy order comes in, there must currently be enough outstanding sell orders with a price of at most the offered price to completely fulfil this order. If there are, the order will be fulfilled in the same way as a normal order. Otherwise, the order is completely cancelled, without any transaction taking place. Note that multiple orders may be used to complete a FoK order, as long as it happens immediately.

FoK sell orders are processed in a similar way, but then there should be sufficient outstanding buy orders with a price of at least the asked price.

As an example, consider the first sample case. The six orders are handled as follows:

1. The first order is added to the list of outstanding orders.
2. The second order is partially fulfilled by selling 10 BAPC to the first order. This removes the first order from the list of outstanding orders, and adds the remainder of the second order (consisting of 10 BAPC) to this list.
3. The third order is added to the list of outstanding orders.
4. The fourth order is a FoK buy order that cannot be immediately fulfilled, so it is ignored. It is not added to the list of outstanding orders.
5. The fifth order can be immediately fulfilled by first buying 10 BAPC from order 2 and then buying 50 BAPC from order 3. The resulting list of outstanding orders only consists of the remaining 8 BAPC of order 3.
6. The sixth order is added to the list of outstanding orders.

Given a list of all orders in the order that they have been made, create a list of all transactions that have been performed by your exchange.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 10^5$), the number of orders.

- n lines, each describing an order:
  - A string s, either "`buy`" or "`sell`", the side of the order.
  - A string t, either "`normal`" or "`fok`", the type of the order.
  - An integer p ($1 \leq p \leq 10^9$), the offered or asked price per BAPC.
  - An integer a ($1 \leq a \leq 10^9$), the amount of BAPC being asked or offered.

Output

The output consists of the number of performed transactions, and then for each transaction, in the order that they have been performed:

- The index of the corresponding "`sell`" order.
- The index of the corresponding "`buy`" order.
- The amount of BAPC being traded.

Here, the index of an order is its position in the input, where the first order has index 1.

Example

Input	Output
6 buy normal 700 10 sell normal 500 20 sell normal 800 58 buy fok 600 30 buy fok 900 60 sell normal 300 42	3 2 1 10 2 5 10 3 5 50
3 buy normal 19 10 buy normal 19 20 sell fok 19 17	2 3 1 10 3 2 7

Problem J. Joppiesaus Jailbreak

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

You have recently decided to pick up speedrunning the video game *Bario: A Plumber's Cousin*. In this 2D platforming console classic, you play as Bario, an Italian electrician travelling the world to find his long lost cousin. The game consists of a number of side-scrolling levels with a bus at the end that takes Bario to the next level. Unfortunately, years of optimizations have led to a world record that is currently tied between hundreds of speedrunners and you feel like matching the world record at this point is no longer that big of an achievement. Instead, you try to beat the tied world record by any means necessary.

At first, this seems impossible: Bario has a maximum right speed of 1000 pixels per second, and the current strategies already hold this speed through the entire level. However, completing a level always takes an integer number of frames. If Bario reaches the bus halfway through a frame, the game still has to wait for the frame to complete before starting the next level. Normally, this does not influence speedrunning, as each console runs the game at the same, constant frame rate f. That is, unless you apply a specific condiment mix to the game disk. You would prefer not to go into detail as to how you know this, but applying a specific mix of mayonnaise and curry spices (more commonly known as the Dutch specialty Joppiesaus) to the game disk allows you to set the frame rate of the game to any positive real number. This new frame rate cannot exceed the original frame rate f and remains constant for the entire game. Using your new strategy, what is the fastest time in which you can finish the game? The timing stops when the final frame ends.

For example, consider the third sample input. By modifying the game to run at $\frac{3000}{1249}$ frames per second, both levels complete in 15 frames, or 6.245 seconds. The total time of 12.49 seconds beats the current world record of 12.6 seconds at the original 10 frames per second.

Input

The input consists of:

- One line with two integers n and f ($1 \leq n \leq 10^5$, $1 \leq f \leq 10^3$), the number of levels and the original frame rate of the game in frames per second.
- One line with n integers ℓ ($1 \leq \ell \leq 10^6$), the length of each level in pixels. The total length of all levels does not exceed 10^6 pixels.

Output

Output the fastest time, in seconds, in which you can finish the game.

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

Example

Input	Output
1 10 1234	1.234
1 10 12	0.1
2 10 6245 6212	12.49
2 20 6245 6212	12.47409677
3 50 7146 2657 8164	17.96910941

Problem K. Thanksgiving

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

You live in a small town off the beaten track. So far from civilization, people talk and gossip spreads like a wildfire. When Mark hears something new, he tells it to Amy, Amy tells it to Natalia, Natalia tells it to you, and you are always surprised when Amy already knows the good stuff that you are trying to tell her. In general, everyone has someone they trust and tell every minute detail about the weight of Berta's cat, the chaos in the town hall after thanksgiving, and the paltry harvest this year. Only Bert keeps to himself. If only you knew how many people hear the gossip you tell Amy, maybe you would reconsider and be more like Bert.

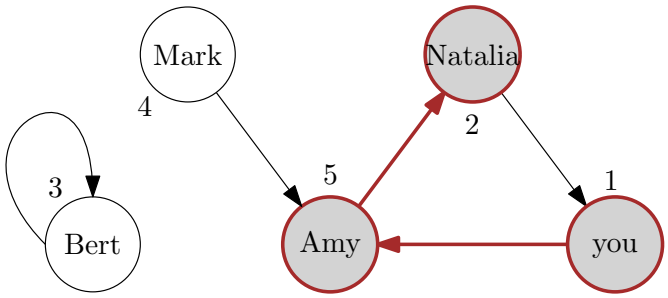


Figure 5. Illustration of Example Input 2 with the characters from the story. If you speak to Amy, then Amy, Natalia, and you will know the gossip. Thus, the answer is 3.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 1000$), the number of people.
- One line with n integers $p_1, \ldots, p_n$ ($1 \leq p_i \leq n$), meaning person i tells their gossip to person p_i.

See Figure 5 for an example.

Output

Output the number of people that will hear your gossip. You are person number 1 and you always count yourself towards the answer.

Example

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

Problem L. Magic Marbles

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

A frozen marble on ice. Image by rihaij, Pixabay

You may know *Wizard's Chess*, which is a magical version of chess, but there are more magical games in the wizarding world. Another one you may recognize is *Magic Marbles*, which is similar to the muggle game *Marble Temple*. In this game, you are given a chain of differently coloured marbles m, which you need to destroy. To do this, you can add additional marbles to the chain. This may sound counterproductive, but if there ever is a *run* of at least k consecutive marbles that are of the same colour, then all of these marbles magically disappear instantly. The resulting gap is closed by moving the remaining marbles closer together, which can lead to new runs that then again disappear.

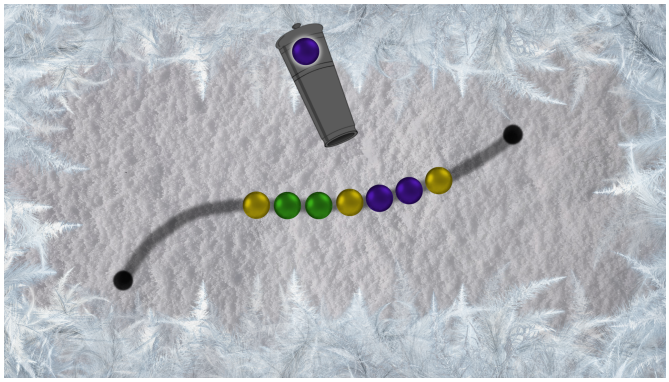


Figure 6. The initial state of the first example. The first marble will be added between the second yellow marble and the first blue marble. Since this results in a run of three blue marbles, all of them instantly disappear.

This year there is a great Magic Marbles tournament in Hogwarts. However, you fear that some magicians are not as sincere as you and may try to cheat in this magic tournament. Therefore, you decided to simulate the game without magic.

Input

The input consists of:

- One line with three integers n, k, q ($1 \leq n, q \leq 2 \cdot 10^5, 2 \leq k \leq 4 \cdot 10^5$), the initial length of the marble chain m, the minimal run length where marbles disappear, and the number of marbles that will be added during the game.
- One line with n space-separated integers m_i ($1 \leq m_i \leq 10^6$), where m_i is the colour of the i-th marble of the initial chain m. It is guaranteed that each run of marbles of the same colour has length less than k.
- q lines, each containing two integers p_x and m_x ($0 \leq p_x \leq |m|, 1 \leq m_x \leq 10^6$), the position where the next marble will be inserted and the colour of the marble that is inserted. $|m|$ denotes the current length of the marble chain. If the position p_x is 0, the marble will be added in front of the currently first marble. Otherwise, the marble will be added after the p_x-th marble.

Output

For each of the q insertions of a new marble, output a single integer $|m|$, the length of the marble chain after that insertion and any deletions that were caused by it.

Example

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