

## Contents

1 X Maratón UFPS - y I Maratón Interuniversitaria UFPS+UTP+Amazonía	2
2 Grupo de Estudio en Programación Competitiva	4
3 Instrucciones	4
4 Reglas	4
5 Lista de Problemas	6
6 Fe de Erratas	7



## 1 X Maratón UFPS - y I Maratón Interuniversitaria UFPS+UTP+Amazonía

Desde el año 2014, el programa Ingeniería de Sistemas de la Universidad Francisco de Paula Santander (UFPS) inició un interesante proceso para promover la Programación Competitiva, como parte de las actividades del Semillero SILUX (Linux, Software Libre y Licencias Abiertas). El propósito fundamental fue el desarrollo de competencias de pensamiento computacional, como resolución de problemas, programación de computadores, trabajo colaborativo y liderazgo.

Los pioneros de dicho proyecto fueron dos estudiantes, quienes ya se graduaron y dejaron como herencia una gran motivación. Ahora siguen colaborando con el Semillero que es liderado directamente por los estudiantes, quienes ingresan desde primer semestre con el entusiasmo que trae el sueño de llegar algún día a la Competencia Mundial ICPC (The International Collegiate Programming Contest <https://icpc.baylor.edu/>).

Desde entonces, cada año, el evento más importante es la Maratón Interna de Programación de la UFPS, que propone un conjunto de retos para poner a prueba las habilidades en algoritmia, programación y trabajo en grupo de los estudiantes. En dicho evento un actor fundamental siempre ha sido la Red de Programación Competitiva (RPC), quien apoya toda la logística de preparación de la competencia y además ofrece su plataforma tecnológica y su equipo de trabajo. El lema de RPC es “Aquí crecemos juntos” y la UFPS ya lleva diez (10) años creciendo en Programación Competitiva junto a muchas universidades en varios países de toda Latinoamerica.

El el año 2022, con el apoyo de la Empresa de Tecnología Pragma, realizamos un bonito evento llamado Primer Training Camp de los Santanderes, donde juntamos varias universidades de Santander (UIS y UDI) y Norte de Santander (Ocaña, Pamplona, UDES, Simón Bolívar, Remington), junto con UTP y U. Amazonía.

Y para este año 2023, nos juntamos UTP, UFPS y U. Amazonía para la primera Maratón Interuniversitaria. Estaremos en simultánea, presenciales, en Cúcuta, Pereira y Florencia y en la red estarán los equipos desde Mexico hasta Argentina para cerrar el año 2023 y soñar con el año 2024.

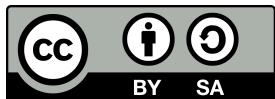
Este año nos complace presentar un conjunto de quince (15) problemas, todos inéditos, escritos por estudiantes, profesores y graduados de varias universidades. Por la pandemia del COVID19, fueron dos años virtuales y este es el segundo año nuevamente presenciales, algo muy importante, porque regresan los globos y las fotografías que registran la energía y alegría de nuestros estudiantes.

Reconocimiento y agradecimiento especial al equipo de trabajo de este año:

- Hugo Humberto Morales Peña - Profesor Universidad Tecnológica de Pereira, Colombia
- Milton Jesús Vera Contreras - Profesor UFPS (desde Cúcuta)
- Diana Espinosa y su semillero - Universidad de la Amazonía, Colombia
- José Manuel Salazar Meza - Graduado UFPS (desde Cúcuta)
- Carlos Alberto Salazar Meza - Estudiante UFPS (desde Cúcuta)
- María Alexandra Velásquez Jaimes - Estudiante UFPS (desde Cúcuta)
- Luis Daniel Carreño Pitalua - Estudiante UFPS (desde Cúcuta)
- Santiago Felipe Alferez Villamizar - Estudiante UFPS (desde Cúcuta)

- Eddy Miguel Ramirez - Profesor (desde Costa Rica)
- Equipo Red de Programación Competitiva (Fabio Avellaneda, Johany Careño)
- Estudiantes, Compañeros Profesores y Directivos de las universidades organizadoras UFPS, UTP y Universidad de la Amazonía.

Este trabajo se comparte bajo licencia Creative Commons Reconocimiento-Compartir Igual 4.0 Internacional (CC BY-SA 4.0)





## 2 Grupo de Estudio en Programación Competitiva

El grupo de estudio en Programación Competitiva hace parte del Semillero de Investigación SILUX (Linux, Software Libre y Licencias Abiertas) y tiene como fin preparar y fortalecer a los estudiantes de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander en competencias de resolución de problemas, algoritmia, programación de computadores, trabajo colaborativo y liderazgo. Se aprovecha el entorno competitivo o gamificación porque favorece el aprendizaje tanto de habilidades hard como habilidades soft.

Los integrantes del grupo de estudio han participado en la Maratón Nacional de Programación desde el año 2015, logrando por 9 años consecutivos la clasificación a fase Regional Latinoamericana. En las dos últimas competencias de 2022 y 2023 la UFPS se ha logrado ubicar en el top 3 y top 6.

## 3 Instrucciones

Puedes utilizar Java, C, C++ o Python, teniendo en cuenta:

1. Resuelve cada problema en un único archivo. Debes enviar a la plataforma únicamente el archivo .java, .c, .cpp, o .py que contiene la solución.
2. Todas las entradas y salidas deben ser leídas y escritas mediante la entrada estándar (Java: Scanner o BufferedReader) y salida estándar (Java: System.out o PrintWriter).
3. En java, el archivo con la solución debe llamarse tal como se indica en la linea "Source file name" que aparece debajo del título de cada ejercicio. En los otros lenguajes es opcional (puede tener cualquier nombre).
4. En java, la clase principal debe llamarse igual al archivo (Si el Source File Name o basename indica que el archivo debe llamarse example.java, la clase principal debe llamarse example).
5. En java, asegúrate de borrar la linea "package" antes de enviar.
6. Tu código debe leer la entrada tal cual se indica en el problema, e imprimir las respuestas de la misma forma. No imprimas lineas adicionales del tipo "La respuesta es..." si el problema no lo solicita explicitamente.
7. Si tu solución es correcta, en unos momentos la plataforma te lo indicará con el texto "YES". En caso contrario, obtendrá un mensaje de error.

## 4 Reglas

1. Gana el equipo que resuelve mas problemas. Entre dos equipos que resuelvan el mismo número de problemas, gana el que los haya resuelto en menos tiempo (ver numeral 2).
2. El tiempo obtenido por un equipo es la suma de los tiempos transcurrido en minutos desde el inicio de la competencia hasta el momento en que se envió cada solución correcta. Habrá una penalización de 20 minutos que se suman al tiempo por cada envío erróneo (esta penalización solo se cuenta si al final el problema fue resuelto).
3. NO se permite el uso de internet durante la competencia. Únicamente se puede acceder a la plataforma en la cual se lleva a cabo la competencia.

4. NO se permite el uso de dispositivos electrónicos durante la competencia.
5. NO se permite la comunicación entre miembros de equipos diferentes. Cada integrante solo puede comunicarse con sus dos compañeros de equipo.
6. Se permite todo tipo de material impreso (libros, fotocopias, apuntes, cuadernos, guías) que el equipo desee utilizar.



## 5 Lista de Problemas

A continuación la lista de los quince (15) problemas a resolver. Un primer reto y logro es resolver alguno de los problemas. Un segundo reto es lograr resolver cualquiera de los problemas más rápido que todos los demás. Un tercer reto es lograr resolver todos los problemas. Un cuarto reto es lograr ubicarse en el top 3 de la competencia. Un quinto reto es lograr ubicarse en el top 5 o ser el mejor equipo novato o el mejor equipo femenino ;)

Pero recuerda que solo al competir ya se es ganador ;)

1. A Love Calculator (Page 8)
2. Betting on Luck (Page 9)
3. Conner's Towers (Pages 10-11)
4. Dictionary (Pages 12)
5. Emergency in Treenity (Pages 13-14)
6. For Honor Reasons (Pages 15-16)
7. Guardians (Pages 17-18)
8. Hemerology (Pages 19-20)
9. I Love Python (Page 21)
10. Judging Your Tetris Skills (Pages 22-24)
11. Kepler Barriers (Pages 25-26)
12. Longest Periodic Substring (Page 27)
13. Matchsticks Puzzle (Pages 28-29)
14. Night of the Little Candles (Pages 30-31)
15. OMG!!! A Rainy Road (Pages 32-33)

**Nota 1:** Los enunciados de los problemas están en idiomas inglés y español.

**Nota 2:** A continuación, en la siguiente página, encontrará el detalle de tiempos límite en segundos para C++, Java y Python, para cada caso de prueba, conforme a las pruebas realizadas por el equipo de trabajo de RPC y UFPS, usando el servidor de Boca de RPC y UFPS. Tenga en cuenta que los problemas son juzgados de manera automática con varios casos de prueba.

TimeLimit				
Letra	Nombre	C, C++	Java	Python
A	A Love Calculator	1	1	1
B	Betting On Luck	2	3	5
C	Conner's Towers	1	7	3
D	Dictionary	3	6	7
E	Emergency in Treenity	2	5	5
F	For Honor Reasons	2	5	5
G	Guardians	1	5	5
H	Hemerology	1	1	1
I	I love Python	1	1	1
J	Judging Your Tetris Skills	1	1	1
K	Kepler Barriers	2	10	10
L	Longest Periodic Substring	2	5	5
M	Matchsticks Puzzle	3	3	1
N	Night of the little candles	1	1	1
O	OMG!!! A Rainy Road	1	5	2

## 6 Fe de Erratas

**Errare humanum est**, lo importante es reconocer y corregir el error. Si durante esta competencia se presentan errores, aparecerán aquí y se actualizarán en linea, informando a través de la plataforma:

1. En el problema G se corrigió la versión de inglés, que tenía una pequeña diferencia respecto a la versión en español. Este ajuste no afectó a los participantes.
2. En el problema F se ajustó el ejemplo de entradas y salidas, para consistencia con una aclaración del enunciado. Este ajuste no afectó a los participantes.
3. En el problema F el basename estaba Honor en el cuadernillo, pero era Fhonor en Boca. Este ajuste no afectó a los participantes.
4. En el problema E había un error en un caso de prueba, gracias a los amigos mundialistas de Argentina por la clarificación. Se juzgó de nuevo el reto y no se afectó el resultado final de la competencia.

Entendemos la inconformidad que estos errores generan, pero también confiamos en su comprensión. No es una tarea sencilla llevar a cabo una maratón de este tipo. Se requiere la colaboración voluntaria de muchas personas, quienes se esfuerzan mucho más allá de sus obligaciones. Y es algo que hemos logrado mantener durante diez años entre la UFPS y RPC, siempre con mucho entusiasmo y cariño para mantener viva la llama de la Programación Competitiva.

## Problem A. A Love Calculator

Source file name: Alove.c, Alove.cpp, Alove.java, Alove.py

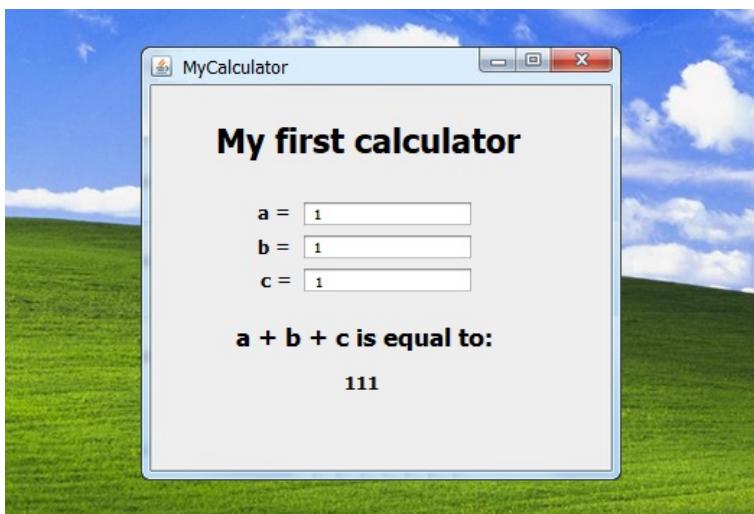
Input: standard input

Output: standard output

Author(s): Carlos Alberto Salazar Meza - UFPS (Student)

María is a student at the Francisco de Paula Santander University known for being the favorite student of Professor Eduardo, a world-famous academic with countless achievements.

For this reason, when María failed her first exam with Eduardo, this kind professor offered to give her an extra grade if she did a special job: create a calculator that sum three numbers  $a$ ,  $b$  and  $c$ . María accepted the proposal and got to work, the next week she presented the following program:



The sum of  $a = 1$ ,  $b = 1$  y  $c = 1$  is equal to 111.

Everyone in the class was surprised, but they were even more surprised to see that the famous professor Eduardo congratulated María for creating a concatenation calculator, and decided to give the following problem for his class:

There are three integers, you must assign each one of these values to  $a$ ,  $b$  and  $c$  in such a way that María's calculator generates the greatest result. A result  $x$  is greater than another result  $y$  if the decimal value of  $x$  is greater than the decimal value of  $y$ .

Can you help the class master María's calculator and solve the problem?

### Input

The only line contains three integers  $x$ ,  $y$  and  $z$  ( $0 \leq x, y, z < 10^7$ ) without leading zeros, the numbers that should be assigned to  $a$ ,  $b$  and  $c$ .

It is guaranteed that the result from María's calculator will be less than 10 digits.

### Output

Prints a line with the greatest possible result obtained when entering the data into the calculator.

### Examples

Input	Output
1 4 12415	4124151
21 789 12	7892112

## Problem B. Betting on Luck

Source file name: Betting.c, Betting.cpp, Betting.java, Betting.py  
 Input: standard input  
 Output: standard output  
 Author(s): Carlos Alberto Salazar Meza - UFPS (Student)

Alexa and Beto are the two most famous gamblers of Barcelona, and they are looking for a new way to waste their money and youth. In this search, they found the “*ultimate rolls dice game*”, which consists of the following rules:

- The game consist of  $n$  rounds.
- Alexa have a multiplicator  $m$  that is initially equal to 1.
- In a round, each player rolls two dices and adds the values obtained,  $d$  is equal to the absolute difference between Alexa's sum and Beto's sum.
- If Alexa's sum is greater than Beto's sum, she win the round and must choose exactly one of the following actions:
  1. Change the value of  $m$  to  $d$ .
  2. Add  $m * d$  to her score and reset the value of  $m$  to 1.
- If Beto's sum is greater than or equal to Alexa's sum, he wins the round, so he adds  $d$  to his score and resets the value of  $m$  to 1.

After finishing the  $n$ th round, each player receives from the other an amount of money equal to his own final score.

Alexa is a risk taker, so when she wins a round, the higher the value of  $d$ , the greater the probability that she change the value of  $m$  seeking to get more points. To be more precise, the probability  $p$  that Alexa will change the value of  $m$  is given by  $p = \frac{d}{10}$ . Additionally, she is so focused on the game that she does not take the turn number into account when choosing her action.

Alexa is having a bad streak, so she would like to know the expected value of the final score that each player will have after  $n$  rounds to know if it is worth betting or not, can you help her?

It is guaranteed that, in the given tests, the required expected values can be represented as an irreducible fraction  $\frac{x}{y}$  where  $x$  and  $y$  are integers and  $y \not\equiv 0 \pmod{10^9 + 7}$ .

### Input

The input contains one integer  $n$  ( $1 \leq n \leq 2 * 10^5$ ), the number of rounds that Alexa and Beto will play.

### Output

Print the expected value of the final score of Alexa and Beto modulo  $1000000007$  ( $10^9 + 7$ ).

### Examples

Input	Output
3	872293200 782407417
6	164529281 564814827

## Problem C. Conner's Towers

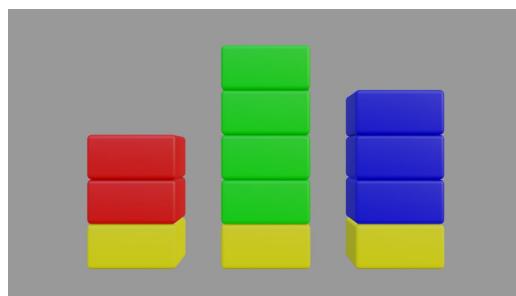
Source file name: Conner.c, Conner.cpp, Conner.java, Conner.py

Input: standard input

Output: standard output

Author(s): Eddy Ramírez - UNA & TEC Costa Rica (Professor)

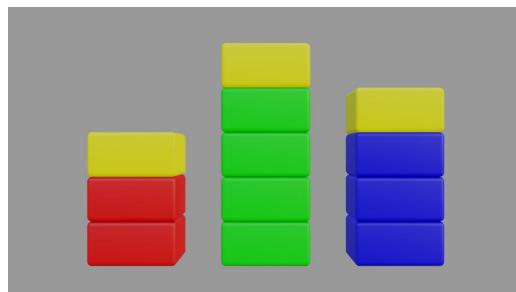
Conner is a baby who really enjoys playing with wooden blocks. To keep him entertained, his dad has set him a challenge. Given a quantity  $A$  of red blocks,  $B$  of green blocks, and  $C$  of blue blocks, he will stack the blocks according to their colors. Additionally, a yellow block will be placed at the base of each stack. For example, the initial arrangement for  $A = 2$ ,  $B = 4$ , and  $C = 3$  would look like the image shown:



Posición inicial

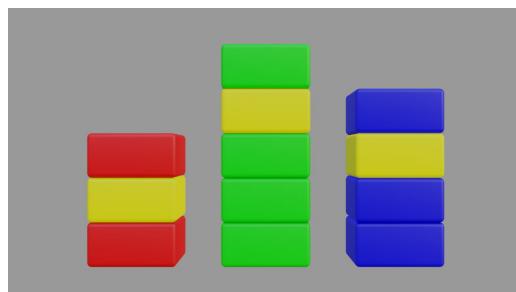
From here, a “move” consists of taking the block located at the three bases and placing them on top of each stack of blocks.

So, for the described case, the position after move 1 would be as shown:



Position after move 1

And after move 2, the position would be:



Position after move 2

Conner is also a baby who is starting to notice colors and really likes yellow. A move is considered successful for Conner if a yellow block remains at any base.

You must create a program that, given the values  $A$ ,  $B$ ,  $C$  and  $N$  (where  $N$  is the number of moves), indicates how many moves would be successful for Conner. The initial position is not considered successful.

## Input

The input consists of multiple test cases. The first line contains a number  $T$  with  $1 \leq T \leq 10^5$ , indicating the number of test cases.

The next  $T$  lines each contain 4 numbers separated by a space:  $A, B, C, N$ , where  $0 \leq A, B, C, N \leq 10^{18}$ . These numbers represent the quantity of red, green, and blue pieces respectively, with  $N$  indicating the number of moves to be made.

## Output

For each test case, you should print on a single line only one number representing the quantity of moves that Conner would consider successful.

## Examples

Input	Output
2	15
100 101 0 15	7
2 3 4 10	



## Problem D. Dictionary

Source file name: Dictionary.c, Dictionary.cpp, Dictionary.java, Dictionary.py

Input: standard input

Output: standard output

Author(s): Hugo Humberto Morales Peña - UTP (Professor)

There is a word dictionary (where there are obviously no repeated words) on which we want to consult the number of words that are in the range between two words, based on the lexicographic order.

For example, if you have a dictionary with the following set of words:

```
{camila, angel, margot, bambi, sabrina, mabel, celia}
```

and it is asked about the number of words that are in the range between the words: **candy** and **patrick**, based on the lexicographic order, the answer would be 3 because the words that are in the range according to their lexicographic order are: **celia**, **mabel** and **margot**.

### Input

The input consists of a single test case. The first line contains two positive integers  $n$  and  $q$  ( $1 \leq n \leq 10^6$ ,  $1 \leq q \leq 5 \cdot 10^5$ ) which respectively represent the quantity of words in the dictionary and the total number of queries to be made about the dictionary.

Then  $n$  lines are presented, each containing a word of maximum 10 lowercase letters in the English language alphabet.

Finally, in the test case  $q$  lines are presented, each of them presenting two different words  $w_1$  and  $w_2$  (separated by a blank space), each word is of maximum length 10 and consists only of lowercase letters of the English language alphabet. It is guaranteed that word  $w_1$  comes before word  $w_2$  in the lexicographic order.

### Output

The output of the problem must contain  $q$  lines, each containing a non-negative integer representing the number of words that are in the range  $[w_1, w_2]$  in the lexicographic order of the words in the dictionary.

Input	Output
7 9	3
camila	7
angel	1
margot	7
bambi	3
sabrina	2
mabel	0
celia	4
candy patrick	2
angel sabrina	
candy conner	
angel vivian	
celia penny	
daisy margot	
daisy india	
ada lara	
lara penny	

## Problem E. Emergency in Treenity

Source file name: Emergency.c, Emergency.cpp, Emergency.java, Emergency.py

Input: standard input

Output: standard output

Author(s): María Alexandra Velásquez Jaimes - UFPS (Student)

Far away from any civilization, in one of the most mysterious places on earth, lies the fairy kingdom, Treenity. This ancient and immense kingdom has the peculiarity that it can be represented as a tree of  $n$  nodes, where node 1 corresponds to the capital and the remaining nodes are magical cities. The connections between these cities are established through magical paths, specifically there are  $n - 1$  paths that interconnect the entire kingdom. The cost of using a path  $i$  between two cities is  $w_i$  units of magic dust.

Treenity had enjoyed centuries of tranquility, but this peace was interrupted by an emergency that could mark the end of its history: a powerful wizard who wishes the total annihilation of this magical kingdom. In response, the fairy queen decided to work tirelessly day and night to create a protection spell, but due to the short time available she was unable to complete it, so the effect only arises in cities such that the magic dust cost to reach to them from the capital is even, (the cost of magic dust to go from the capital to a city  $u$  is equal to the sum of the costs of magic dust on the way from the capital to city  $u$ ).

Luckily, the queen has a spell to change the distribution of cities in her kingdom, which is capable of removing the path that connects a pair of cities  $u$  and  $p$  and creating a new one between city  $u$  and another city  $v$ , with a new cost of magic dust units, and without affecting the kingdom's particularity of being represented as a tree. Here  $p$  is the closest node to  $u$  on the path between  $u$  and the capital. In other words,  $p$  is the direct parent of  $u$  in the tree.

In order to save the kingdom of Treenity, the queen needs to determine the number of cities that are not protected by her spell, i.e. those whose cost of magic dust to be reached from the capital is odd. To carry out this task, she sends you, her most loyal servant, with the responsibility of calculating this number.

To save Treenity, the queen begins to use her distribution change spell and asks you, her most loyal servant, to count the number of cities that are not protected by her protection spell, that is, those in which The cost of magic dust to get to them from the capital is odd.

In this situation, two types of events can occur:

- Type 1: The queen uses the distribution change spell to remove the path between node  $u$  and  $p$  (her direct parent), and create a new path between  $u$  and  $v$  with a new magic dust cost  $w$ .
- Type 2: The queen asks you for the number of cities that are not protected by her protection spell.

### Input

The first line contains an integer  $n$  ( $3 \leq n \leq 10^5$ ), – the number of magical cities in the kingdom of Treenity.

The following  $n - 1$  lines contain three integers  $u$ ,  $v$  and  $w$  ( $1 \leq u, v \leq n, 1 \leq w \leq 10^9$ ) indicating that there is a path with cost  $w$  of magic dust between cities  $u$  and  $v$ .

The next line contains an integer  $q$  ( $1 \leq q \leq 10^5$ ) – the number of events.

Each of the following  $q$  lines begins with an integer  $t_i$  ( $1 \leq t_i \leq 2$ ), which represents the type of event.

If  $t_i = 1$ , then the  $i$ -th event is a queen distribution change spell, and is followed by three integers  $u$ ,  $v$  and  $w$  ( $1 \leq u, v \leq n, (u \neq 1), 1 \leq w \leq 10^9$ ) representing a distribution change spell.

If  $t_i = 2$ , then the  $i$ -th event is a request from the queen.



## Output

For each 2-type event you must print a single line with the number of cities that are not currently protected by the queen's protection spell.

## Examples

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

## Problem F. For Honor Reasons

Source file name: Forhonor.c, Forhonor.cpp, Forhonor.java, Forhonor.py

Input: standard input

Output: standard output

Author(s): Carlos Alberto Salazar Meza - UFPS (Student)

The kingdoms of the continent of Endor, famous for living in constant conflict, finally want an era of peace and are doing all the preparations so that it can happen, so they hired a council of wise men who will determine requests to establish alliances between the kingdoms in order to achieve peace. They also hired you, the best courier on the continent, to deliver the requests between them.

Despite wishing for an era of peace, the inhabitants of the continent are the proudest and most arrogant on the planet and will never agree to create an alliance with a kingdom with which they were at war in the past, nor with a kingdom that is an ally of an enemy kingdom. If such a request were to come to them, they would see it as a mockery against their honor and would start a war on the spot.

The wise men insist on continuing the process without taking into account the pride of the inhabitants. So to avoid a disaster, you decide not to deliver those requests that will cause a war, and you contact the famous historian Selram to find out exactly in which cases this would happen.

Selram gives you a list of wars that occurred in the past and tells you the following:

There are  $n$  kingdoms on Endor, numbered from 1 to  $n$ . Let  $A$  and  $B$  be the kingdoms selected by the council to carry out an alliance request,  $X$  be a kingdom that belongs to the same alliance as  $A$  and  $Y$  be a kingdom that belongs to the same alliance than  $B$ , a war will break out if at least one of the following conditions is met:

- $A$  and  $B$  were at war in the past
- $A$  and  $Y$  were at war in the past
- $X$  and  $B$  were at war in the past
- $X$  and  $Y$  were at war in the past

With this information in hand you can begin your task of delivering the alliance requests determined by the wise men council. But before that, Selram needs your help with his record of important historical events, so for each request he asks you to tell him whether it was not delivered to prevent a war, whether it is unnecessary because the kingdoms already belong to the same alliance, or if it should be delivered to strengthen peace on the continent.

### Input

The first line consists of two integers  $n$  and  $m$  ( $2 \leq n \leq 10^5$ ,  $0 \leq m \leq 2 * 10^5$ ), which indicate the number of kingdoms on the continent and the number of wars in Selram's list.

The following  $m$  lines contain two integers  $u$  and  $v$  ( $u \neq v$ ) – the identifiers of the kingdoms that were at war in the past.

The next line consists of an integer  $q$  ( $1 \leq q \leq \min(2 * 10^5, \frac{n*(n-1)}{2})$ ), the number of alliance requests that the council of wise men determined.

The following  $q$  lines contain two integers  $u$  and  $v$  ( $u \neq v$ ) – the identifiers of the kingdoms involved in the alliance request determined by the wise men council. The requests for alliance are given in chronological order. It is guaranteed that the council of wise men will not determine more than one alliance request between the same pair of kingdoms.



## Output

For each alliance request determined by the wise men, print a line containing the response according to the following cases:

- “**AVOID WAR**” if that alliance request will cause a war.
- “**NEXT**” if that alliance request is unnecessary.
- “**FOR THE PEACE**” if that alliance request should be delivered.

## Examples

Input	Output
5 1	FOR THE PEACE
1 2	FOR THE PEACE
6	AVOID WAR
1 3	AVOID WAR
2 4	FOR THE PEACE
3 4	NEXT
1 2	
1 5	
3 5	

## Problem G. Guardians

Source file name: Guardians.c, Guardians.cpp, Guardians.java, Guardians.py

Input: standard input

Output: standard output

Author(s): Juan José Ortiz Plaza - UNIAMAZONIA (Student)

In a distant kingdom, heroes Yari and Flora embark on an epic adventure through a mysterious maze full of dungeons. The maze can be represented as a tree with  $n$  nodes, where each node  $i$  is a dungeon that is protected by a guardian whose health is represented by an integer  $h_i$ . Moreover, there is only one way to enter and exit the maze, that is through two special portals located at nodes  $a$  and  $b$ , which are initially unknown to them.

Yari and Flora are extremely capable heroes, but they have one flaw, they are very cowardly. For this reason, they created a special plan to avoid being ambushed along the way or getting trapped inside a dungeon, which has two rules:

- Only visit dungeons on the path from  $a$  to  $b$ .
- A dungeon can only be explored if the health of guardian protecting it is less than or equal to their combined skill  $x$ , otherwise they do not explore that dungeon and continue on their way.



Also, as a preventive measure in order to prepare enough supplies for their adventure, they want to do some simulations to find the maximum number of dungeons they could explore if the entry node was  $a$  and the exit node was  $b$ . Can you help them?

### Input

The first line contains two integers  $n$  and  $q$  ( $1 \leq n, q \leq 10^5$ ), the number of dungeons and the number of simulations that Yari and Flora will do, respectively.

The second line contains  $n$  integers  $h_1, h_2, \dots, h_n$  ( $1 \leq h_i \leq 10^9$ ), representing the health of the guardian protecting the  $i$ th dungeon.

The following  $n - 1$  lines describe the connections of the maze. Each line contains two integers  $u$  and  $v$  ( $1 \leq u, v \leq n, u \neq v$ ), indicating that there is a path between dungeons  $u$  and  $v$ .

The following  $q$  lines contain three integers  $a$ ,  $b$  and  $x$  ( $1 \leq a, b \leq n, 1 \leq x \leq 10^9$ ), representing the entry and exit nodes of the maze in the simulation, and the combined ability of Yari and Flora, respectively.

### Output

For each simulation, print the maximum number of dungeons that can be explored by Yari and Flora.



## Examples

Input	Output
10 3	1
10 5 4 9 2 3 8 7 9 4	1
1 4	2
1 6	
1 5	
4 10	
5 2	
5 8	
4 9	
1 3	
5 7	
6 2 2	
10 8 2	
2 5 8	

## Problem H. Hemerology

Source file name: Hemero.c, Hemero.cpp, Hemero.java, Hemero.py  
Input: standard input  
Output: standard output  
Author(s): Milton Jesús Vera Contreras - UFPS (Professor)

Hemerology is the “*study of technical, historical or religious topics about calendars*” (Wikipedia). The variety of calendars is astonishing: Julian, Muslim, Gregorian, Turkish, Chinese, Russian, Egyptian, Aztec, Inca, Mayan and many more. And everything you learn when studying hemerology is wonderful, like the following rhyme in Spanish for our calendar:

Treinta días tiene noviembre  
con abril, junio y septiembre,  
los demás de treinta y uno,  
excepto febrero mocho,  
que sólo tiene veintiocho  
y veintinueve funesto  
cuando es un año bisiesto.

You can even learn about Computer Science with hemerology. For example, Professor Donald Knuth mentions and explains several calendar algorithms, I hope you have read them or will read them sometime.

In practice, a calendar can be simplified to three concepts: days, months and years. The days are grouped forming months, the months are grouped forming years, and the years advance without limits until the end of time and without the rare leap years. Following this simplification, it is easy to calculate the time elapsed between two given dates.

Could you write, as quickly as possible, an efficient program that calculates the difference between two dates (day, month and year) of any fictitious calendar?

### Input

The entry consists of three lines: the first line has  $n + 1$  numbers (separated by white space) that describe a fictitious calendar: the first number  $1 \leq n \leq 10^6$  is the number of months of a year, and then  $n$  numbers  $1 \leq m_1, m_2 \dots m_{n-1}, m_n \leq 10^6$ , with the number of days of each of the months of the year of that fictitious calendar.

The second line has the oldest date, which consists of three numbers (separated by a blank space):  $1 \leq day_1 \leq 10^6$ ,  $1 \leq month_1 \leq 10^6$ , and  $1 \leq year_1 \leq 10^6$ .

And the third line has the most recent date, which also consists of three numbers (separated by a blank space):  $1 \leq day_2 \leq 10^6$ ,  $1 \leq month_2 \leq 10^6$ , and  $1 \leq year_2 \leq 10^6$ .

The two dates will always be valid in that fictitious calendar.

### Output

A number  $t$  with the total number of days from the oldest date to the most recent date, according to the fictitious calendar.



## Examples

Input	Output
12 31 28 31 30 31 30 31 31 30 31 30 31 25 11 2023 25 11 2023	0
12 31 28 31 30 31 30 31 31 30 31 30 31 25 11 2023 2 12 2023	7
12 31 28 31 30 31 30 31 31 30 31 30 31 1 1 2023 31 12 2023	364
12 31 29 31 30 31 30 31 31 30 31 30 31 1 1 2024 31 12 2024	365
12 31 28 31 30 31 30 31 31 30 31 30 31 1 1 2000 31 12 2010	4014
3 1000000 1000000 1000000 1 1 1000000 1000000 3 1000000	2999999
1000000 "1000000 times 1000000" 1 1 1 1000000 1000000 1000000	99999999999999999999

## Note

Use Fast I/O

## Problem I. I Love Python

Source file name: Ilove.c, Ilove.cpp, Ilove.java, Ilove.py

Input: standard input

Output: standard output

Author(s): Milton Jesús Vera Contreras - UFPS (Professor)

Some students love Python, because they can solve problems with a single line of code. And they hate Java and C++, because they have to write a lot of code and sometimes bad code or hard-code. They are the complete opposite of this meme:



$$\begin{aligned}
 a^n &= \sum_{k=0}^n \binom{n}{k} a^{n-k} b^k \\
 &= \pi r^2 \quad f(x) = a_0 + \sum_{n=1}^{\infty} a_n x^n \\
 &= -2ab + b^2 \quad E = mc^2 \quad e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \dots \\
 &\stackrel{b=2-a-c}{=} E = mc^2 \quad \sin \alpha \pm \sin \beta = 2 \sin \frac{1}{2}(\alpha \pm \beta) \\
 &= a^2 + 2ab + b^2 \quad A = \pi r^2 \quad (1+x)^n = 1 + nx + \frac{n(n-1)x^2}{2!} + \dots \\
 &\stackrel{c=a+b}{=} A = \pi c^2 \quad f^a + f^b = \\
 &= 2 \cos^2 \frac{1}{2}(a+\beta) \cos^2 \frac{1}{2}(a-\beta) \quad E = mc^2 \\
 &\stackrel{c=a+b}{=} (1+x)^n = 1 + nx + \frac{n(n-1)x^2}{2!} + \dots \quad A = \pi r^2 \quad x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \\
 &+ \cos^2 \frac{1}{2}(a+\beta) \cos^2 \frac{1}{2}(a-\beta) \quad (x+a)^n = \sum_{k=0}^n \binom{n}{k} a^k b^{n-k} \\
 &\stackrel{c=a+b}{=} x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (x+a)^n = a^2 - 2a \\
 &+ \cos^2 \frac{1}{2}(a+\beta) \cos^2 \frac{1}{2}(a-\beta) \quad (a-b)^2 = a^2 - 2ab \\
 &\stackrel{c=a+b}{=} x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (a-b)^2 = a^2 - 2ab \\
 &+ \cos^2 \frac{1}{2}(a+\beta) \cos^2 \frac{1}{2}(a-\beta) \quad x^2 = \frac{a^2 - 2ab}{a^2} \\
 &\stackrel{c=a+b}{=} x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad b^2 = c^2
 \end{aligned}$$

Solve with Math Equation



```

if a > b:
    case = 1
else:
    if a < b:
        case = 2
    else:
        if a == b:
            case = 3
        else:
            if a != b:
                case = 4
            else:
                if otherCase:
                    case = 5
                else:
                    #OMG !!!

```

Hard-coding IF/ELSE

Thinking about those students, the teacher proposed this challenge as the simplest of the competition: We have an integer  $n$  ( $0 \leq n \leq 10^{17}$ ) and a digit  $d$  ( $0 \leq d \leq 9$ ) and we want to determine the number  $m > n$  closest to  $n$  whose last digit (rightmost digit) is  $d$ .

Prove the teacher your love for Python by solving this problem with the fewest number of lines ;)

### Input

Two numbers separated by a space:  $n$  ( $0 \leq n \leq 10^{17}$ ) and  $d$  ( $0 \leq d \leq 9$ ).

### Output

The number  $m > n$  closest to  $n$  whose last digit (rightmost digit) is  $d$ .

### Examples

Input	Output
15 4	24
30 1	31
151 1	161
922337203685477581 0	922337203685477590

## Problem J. Judging Your Tetris Skills

Source file name: Tetris.c, Tetris.cpp, Tetris.java, Tetris.py  
Input: standard input  
Output: standard output  
Author(s): Milton Jesús Vera Contreras - UFPS (Professor)

Tetris is a very important video game in the history of technology. It appears in the movie Pixels, starring Adam Sandler, Kevin James, Peter Dinklage, Josh Gad and the beautiful Michelle Monaghan. This film presents a hypothetical case of an alien invasion, in which video games come to life within the human world.

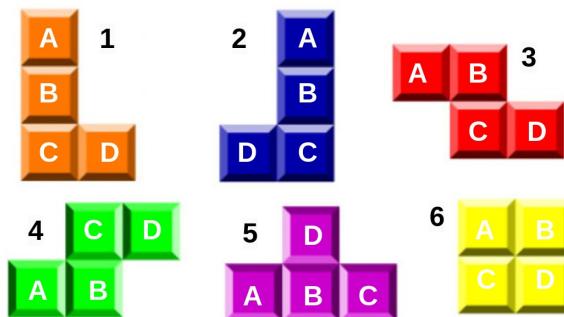


Thinking about this movie, about the rise of the Metaverse, about the apocalyptic ideas of Artificial Intelligence and about how difficult it is to convince students to abandon Windows and the Mouse to use Linux and text-based command shell, the professor sought a computer programming exam from 2004, almost 20 years ago, and adapted it for the 10th UFPS+RPC Marathon 2023.

At that time, exams were on paper and the teacher's brain functioned as a compiler and interpreter. The web and HTML were quite new, there was no cloud computing, nor IDE in the cloud (like our RPC IDE <https://www.rpcide.cloud/>). However, in this era, technological advances are wonderful. and all the creativity and agility of competitive programming lovers is required.

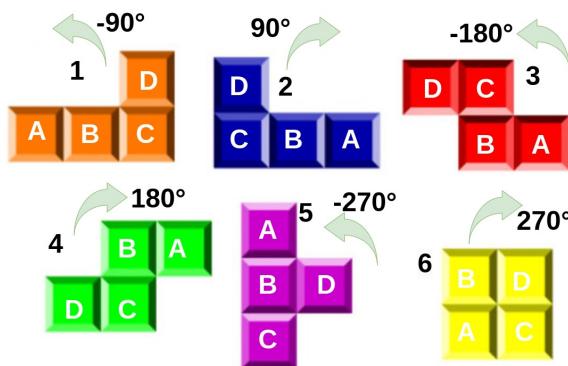
Do you accept the challenge of solving this almost 20-year-old exam in 2023?

Tetris consists of several figures made up of boxes. The following image shows six 6 of the many figures that a Tetris can have, numbered from 1 to 6 and labeling the boxes with the letters A, B, C and D.



The game allows you to rotate the figures 90°, 180° and 270° to the left or right. The following image shows examples of rotations of the six figures.

Imagine that due to some alien invasion, computers lose multimedia and only text can be used. In that



case, the Tetris figures could be represented as a small  $3 \times 3$  matrix and the rotations would be made taking the center of the matrix as the rotation point. The two images above would look like this (printing dash “-” instead of whitespace):

<b>1</b>	<b>2</b>	<b>3</b>	<b>1</b> $-90^\circ$	<b>2</b> $90^\circ$	<b>3</b> $-180^\circ$
A - - B - - C D -	- - A - - B - D C	- - - A B - - C D	- - - - - D A B C	- - - D - - C B A	D C - - B A - - -
<b>4</b>	<b>5</b>	<b>6</b>	<b>4</b> $180^\circ$	<b>5</b> $-270^\circ$	<b>6</b> $270^\circ$
- - - - C D A B -	- - - - D - A B C	- - - A B - C D -	- B A D C - - - -	A - - B D - C - -	- - - - B D - A C

Following this representation we want to simulate a game of Tetris, where we receive the number of the figure 1, 2, 3, 4, 5 or 6 and a set of rotations to the left or right of  $90^\circ$ ,  $180^\circ$  and  $270^\circ$ . The simulation should print the figure in its final state, after all rotations have been applied.

## Input

The input consists of a single line of  $n$  numbers (separated by white space) that describe the scenario to be simulated. The first number  $1 \leq f \leq 6$  is the figure. The second number  $0 \leq r \leq 100$  is the number of rotations applied to the figure in its original state (according to the image in the statement). Then  $r$  numbers  $c_1, c_2, \dots, c_{r-1}, c_r$ , with rotations applied to the figure, whose values can be 90, 180 and 270 for clockwise rotations and  $-90$ ,  $-180$  and  $-270$  for counterclockwise rotations. It is guaranteed to only be those values.

## Output

Print three lines, each line with three characters “A”, “B”, “C”, “D” or dash “-”, without blank spaces, representing the Tetris figure, after having applied the rotations.



## Examples

Input	Output
1 6 90 180 90 -90 90 -90	--- --D ABC
2 5 270 90 -90 90 90 90	--- D-- CBA
3 7 270 90 -90 -90 90 90 -180	DC- -BA ---
4 6 90 180 90 -90 90 180	-BA DC- ---
5 7 270 90 -90 -90 90 90 -270	A-- BD- C--
6 5 270 90 -90 90 270	--- -BD -AC

## Note

Use Fast I/O

## Problem K. Kepler Barriers

Source file name: Kepler.c, Kepler.cpp, Kepler.java, Kepler.py

Input: standard input

Output: standard output

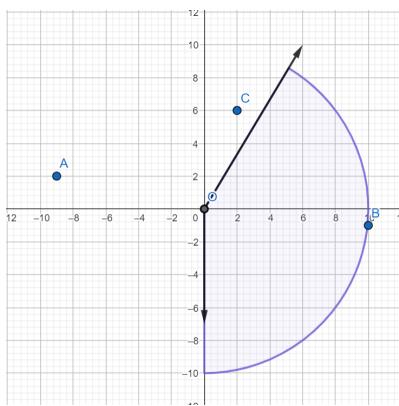
Author(s): Luis Daniel Carreño Pitalua - UFPS (Student)

In Teleportia's domain, technology has reached new horizons due to the invention of Teleportation Towers. These structures, powered by quantum energy, have completely revolutionized the way people move and connect across the domain.

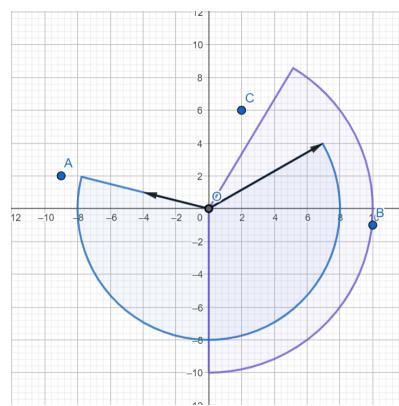
For Teleportation Towers to function perfectly, it is necessary to create a Teleporter Core that serves as a central connection point between all towers. It is enough for a tower to be visible from the Teleporter Core for a connection to exist between the two.

However, due to the very particular characteristics of this domain, a series of quantum barriers, called Kepler Barriers, form every night. These barriers are arranged forming an arc of circumference with respect to the Teleporter Core and are generated one after the other throughout the night (for some unknown reason). During their existence, barriers are capable of blocking the tower's view towards the Teleporter Core, temporarily disabling its operation. Fortunately, the barriers disappear at dawn and the towers return to operating normally.

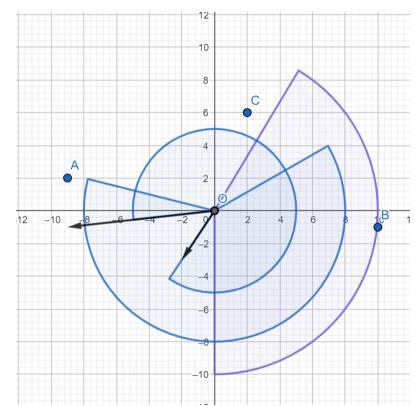
Teleportia can be seen as a two-dimensional plane where the Teleporter Core is located at the coordinate (0, 0). Each of the Teleportation Towers is located at a coordinate  $(x, y)$  of the plane. For its part, each of the Kepler barriers is defined by two vectors,  $\vec{v}$  and  $\vec{u}$ , which determine the angle covered by the barrier. This angle is equal to the one formed from the vector  $\vec{v}$  to the vector  $\vec{u}$  in a counterclockwise direction. A radius  $r$  is also defined, which indicates at what distance the Kepler barrier will be generated with respect to the Teleporter Core.



(a) Occurrence of the first barrier



(b) Occurrence of the second barrier



(c) Occurrence of the third barrier

Visualization of the first test case.

Given the order in which the Kepler barriers appear during a night and the position of the existing Teleportation Towers in the domain, it is necessary to determine how many towers are disabled after the appearance of each of the Kepler barriers. This will allow us to have greater knowledge about the impact that these barriers have on the operation of the towers and thus be able to take appropriate measures to optimize their performance.

### Input

The first line contains two integers  $N, M$  ( $1 \leq N, M \leq 10^5$ ), indicating the number of Teleportation Towers in the domain and the number of Kepler barriers that will appear at night, respectively.



Each of the following  $N$  lines contains two integers  $x, y$  ( $-10^9 \leq x, y \leq 10^9$ ); which indicate the position in the domain of each of the towers. It is guaranteed that two different towers do not share the same location.

The following  $M$  lines describe each of the Kepler barriers, each containing five integers. The first two integers  $x_1, y_1$  ( $-10^9 \leq x_1, y_1 \leq 10^9$ ) define the vector from which the barrier starts. The next two integers  $x_2, y_2$  ( $-10^9 \leq x_2, y_2 \leq 10^9$ ) define the vector where the barrier ends, and the last integer,  $r$  ( $0 < r \leq 10^9$ ), indicating the distance at which the barrier will be generated with respect to the Teleporter Core. The barriers are given in chronological order of appearance.

It is guaranteed that no tower or barrier coincides with the point  $(0, 0)$ .

## Output

Print a single line with  $M$  integers where the  $i$ -th integer indicates the number of towers disabled after the appearance of the  $i$ -th barrier.

## Examples

Input	Output
3 3 -9 2 10 -1 2 6 0 -7 6 10 10 -4 1 7 4 8 -2 -3 -9 -1 5	1 2 3
5 3 6 -6 -3 10 -10 -9 -1 10 -5 2 7 10 -10 0 9 -7 -5 3 -5 9 5 8 1 0 7	2 3 4
2 2 3 3 -3 3 2 1 -1 2 3 1 2 -5 4 5	1 1

## Problem L. Longest Periodic Substring

Source file name: Longest.c, Longest.cpp, Longest.java, Longest.py

Input: standard input

Output: standard output

Author(s): Jose Manuel Salazar Meza - UFPS (Graduate)

A string  $s$  of length  $n$  is called *periodic* if there exists a string  $p$  of length  $k$  ( $k < n$ ) such that  $s$  can be formed by concatenating two or more repetitions of  $p$ . In that case,  $s$  is said to have a *period* equal to  $k$ . For example, the strings “abababab” and “abcabc” are *periodic* because “abababab” can be formed by 4 repetitions of the string “ab” or 2 repetitions of the string “abab”, and “abcabc” can be formed by 2 repetitions of the string “abc”, while the strings “ababa” and “abcdefg” are not *periodic*.

A substring of  $s$  is a non-empty string  $x$  consisting of continuous characters from  $s$ . For example, “lucho”, “luchoneta” and “choneta” are substrings of “luchoneta”, while “ufps” is not.

Given a string  $s$ , your task is to find the longest *periodic* substring of  $s$  or report that there is none. If there are multiple substrings that satisfy the condition, indicate how many **different** ones there are and choose the lexicographically smallest one.

**See notes for clarification.**

### Input

The input consists of a single line containing the string  $s$  ( $1 \leq |s| \leq 2 * 10^5$ ) – a sequence of lowercase English letters.

### Output

If there are no *periodic* substrings in  $s$ , then print 0.

Otherwise, print the number of **different** substrings that satisfy the condition, and on the next line, print the lexicographically smallest longest *periodic* substring of  $s$ .

### Examples

Input	Output
aaaaa	1 aaaaa
aababcbcabab	2 abab
xmaratoninternaufps	0

### Note

Two substrings  $x$  and  $y$  of a string  $s$  are different if either  $|x| \neq |y|$ , or there exists an  $i$  ( $1 \leq i \leq |x|$ ), such that  $x_i \neq y_i$  (and  $|x| = |y|$ ). Here  $|a|$  denotes the length of the string  $a$ , and  $a_i$  denotes the character at position  $i$  of the string  $a$ .

A string  $x$  is lexicographically less than a string  $y$ , if either  $x$  is a prefix of  $y$  (and  $x \neq y$ ), or there exists an  $i$  ( $1 \leq i \leq \min(|x|, |y|)$ ), such that  $x_i < y_i$ , and for any  $j$  ( $1 \leq j \leq i$ )  $x_j = y_j$ .

## Problem M. Matchsticks Puzzle

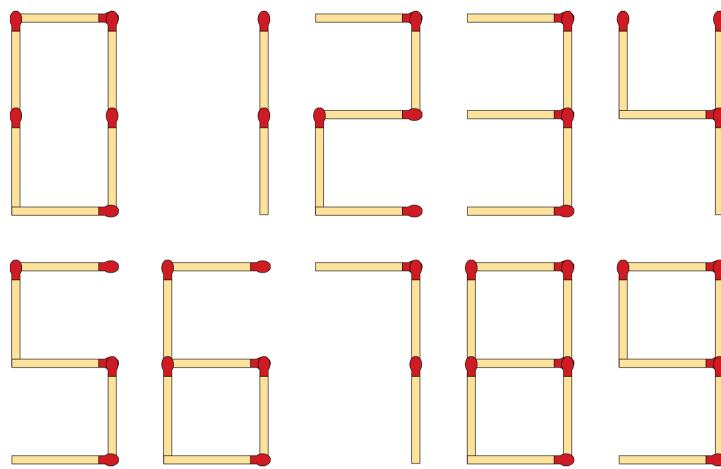
Source file name: Matchsticks.c, Matchsticks.cpp, Matchsticks.java, Matchsticks.py

Input: standard input

Output: standard output

Author(s): Jose Manuel Salazar Meza - UFPS (Graduate)

Ever since he was a child, Manuel has always been interested in puzzles, riddles, and any game that challenges him to solve a problem by thinking and being creative. The other day, he came across a matchsticks puzzle that stated: Given the number 508 made with matchsticks, what is the largest number that can be obtained by moving exactly 2 matchsticks? Could you solve it?



The image shows how to represent the ten decimal digits with matchsticks.

Some correct answers could be 999, 5051 or 51181, depending on the rules of the game. For Manuel, this problem was not a challenge and he managed to solve it correctly in a short amount of time. However, after solving it, he thought of a more challenging problem: Given a number  $s$  with  $n$  digits made with matchsticks, what is the smallest and largest number that can be obtained by moving exactly  $k$  matchsticks, for each  $k$  from 1 to  $M$ , where  $M$  is equal to the total number of matchsticks in the number  $s$ ?

For this problem, he defined some rules:

- Moving a matchstick consists of lifting up it and put down it elsewhere where there was no a matchstick initially. A matchstick can only be moved once.
- It is allowed to remove digits or add new ones both to the left and right of the number  $s$ , as long as, in the end, the formed number does not have empty spaces between digits. It is not allowed to add new digits in between the initial  $n$  digits.
- If you want to convert an existing digit into a different one, you can only convert it into a **single** new digit. For example, 4 can be converted into 3 by moving one matchstick and adding another, but 0 cannot be converted into 11 by lifting up two matchsticks nor into 111 by moving two matchsticks.
- Moving exactly  $M$  matchsticks is equivalent to rearranging all the matchsticks as desired.
- In all cases, The  $M$  matchsticks must be part of some digit in the formed number.

This problem is indeed much more challenging for Manuel, and now both he and you are ready to face it.  
Good luck!

## Input

The first line contains an integer  $n$  ( $1 \leq n \leq 12$ ) indicating the number of digits in the number  $s$ .

The second line contains the number  $s$ , which consists of  $n$  digits made with matchsticks.

## Output

Print a line for each  $k$  from 1 to  $M$  indicating the smallest and largest number that can be obtained by moving exactly  $k$  matchsticks, separated by a space. If there is no answer for a particular  $k$ , print an “\*”.

## Examples

Input	Output
1 1 2	3 3 5 5 17 71 17 71 2 71
2 74	* 13 711 8 711 8 117 12 711 * 8 711
3 013	012 913 047 1017 031 9171 021 11311 08 77711 08 711111 08 711111 047 171111 08 711111 012 711111 021 711111 * 08 711111

## Problem N. Night of the Little Candles

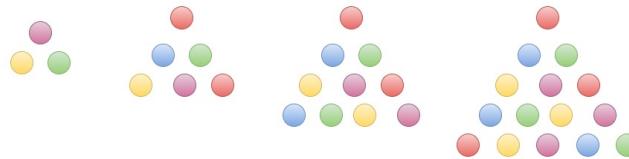
Source file name: Night.c, Night.cpp, Night.java, Night.py  
Input: standard input  
Output: standard output  
Author(s): Milton Jesús Vera Contreras - UFPS (Professor)

An important Colombia's traditions is **The Night of the little candles**, which is celebrated on December 7, on the eve of the Immaculate Conception of the Virgin Mary, a holiday of the Catholic religion. In each family they gather and light colored little candles, thanking or asking for wishes or favors from the Virgin, who will pass that night through the homes of those who light the candles.



At the Francisco de Paula Santander University (UFPS) Cúcuta, that date usually is the start of exams, so some students put little candles praying to save the semester. This year 2023, the students raised money and bought many little candles,  $n \leq 10^{18}$ , and they are going to put them at the UFPS.

The students decided to place the candles forming triangles with sides  $L \geq 2$  candles, because they have the omen that this figure will help them save the semester. They begin with a triangle with side  $L = 2$  little candles, then another with side  $L = 3$  and so on, always growing  $L$  by 1, as in the following figure:



When the candles are not enough to make a new triangle by growing  $L$  by 1, they start again from  $L = 2$  and repeat the procedure as many times as necessary, until there are no more little candles or it is no longer possible to make a triangle.

Could you help students determine how many triangles they could make with  $0 \leq n \leq 10^{18}$  little candles, and if there are leftover little candles, how many are left over?

### Input

The input consists of a number  $0 \leq n \leq 10^{18}$ , the total number of little candles purchased by the students.

### Output

Print two numbers separated by white space. The first number  $t$  is the total number of candle triangles that the students made and the second number  $s$  is the number of candles left over.

## Examples

Input	Output
10	2 1
100	10 2
1000	22 1
10000000000000000000	1836092 0
99999999999999999999	1836091 2

## Problem O. OMG!!! A Rainy Road

Source file name: Rainy.c, Rainy.cpp, Rainy.java, Rainy.py  
Input: standard input  
Output: standard output  
Author(s): Jorge Andres Marles Florez - UFPS (Student)



The best team of Fornique, UFPechugonaS, needs to go to the ICPC Regional Finals, but, due to budget cuts, they only have money to go by road.

Fornique is a country with  $N$  cities, numbered from 1 to  $N$ , connected by  $m$  two-way roads between them, each road connects a pair of cities  $u$  and  $v$ , and takes  $t$  time to go across that road. UFPechugonaS are in city 1, and the ICPC Regional Finals take place in city  $N$ , they would like to know the minimum time to go from city 1 to city  $N$ .

However, the government of Fornique has declared an emergency by the rains, and some roads could be blocked by landslides, in order to not get stuck in traffic or even worse, UFPechugonaS will ride the smartest vehicle in the world, La Luchoneta, it comes with a powerful IA called MagicalGirl, which can predict where and when will be landslides, also, it can predict when some road with a landslide will be released.

To avoid getting buried in some landslide, if the team wants to go across some road, and that road is blocked by a landslide at the time, they must wait for it to be released, also, they can't go inside a road if they don't have enough time to exit it before a landslide occurs.

Unfortunately, all UFPechugonaS' teammates are busy studying for the contest, so they asked you for help. Given the number of cities, roads, and landslides predicted, What is the minimum time La Luchoneta will take in the trip?

### Input

The first line contains 3 integers,  $N$  ( $1 \leq N \leq 10^5$ ),  $m$  ( $n-1 \leq m \leq \min(10^5, \frac{n(n-1)}{2})$ ) and  $d$  ( $0 \leq d \leq 10^5$ ) – the number of cities, the number of roads, and the number of landslides predicted, respectively. It is guaranteed that there is a path between cities 1 and  $N$ .

Then, follow  $m$  lines containing 3 integers,  $u$ ,  $v$  ( $1 \leq u, v \leq n, u \neq v$ ) and  $t$  ( $1 \leq t \leq 10^8$ ), indicating that there is a two-way road between cities  $u$  and  $v$  that takes  $t$  units of time to cross. It is guaranteed that there is no more than one road between two cities  $u$  and  $v$ .

Then, follow  $d$  lines containing 4 integers,  $u$ ,  $v$  ( $1 \leq u, v \leq n, u \neq v$ ),  $l$  and  $r$  ( $0 \leq l < r \leq 10^{10}$ ), indicating that the road between cities  $u$  and  $v$  will be blocked from time  $l$  to time  $r$ . It is guaranteed that there exists a road between  $u$  and  $v$ .

## Output

Output a single integer, the minimum time required to move from city 1 to city  $N$ .

## Examples

Input	Output
10 14 5 1 2 1 1 4 1 2 3 2 3 4 1 4 5 3 4 6 15 4 8 2 5 6 1 5 7 6 6 7 7 6 9 1 7 9 9 8 9 1 9 10 3 2 3 1 2 5 6 0 8 4 8 0 8 6 7 0 10000 7 9 10 12	13
2 1 2 1 2 1 1 2 0 2 1 2 1 3	4

## Note

In the first test case, the order of actions will be:

1. Go to city 1 to city 4 (after that,  $t = 1$ )
2. Go to city 4 to city 5 (after that,  $t = 4$ )
3. Wait in city 5, 4 units of time (after that,  $t = 8$ )
4. Go to city 5 to city 6 (after that,  $t = 9$ )
5. Go to city 6 to city 9 (after that,  $t = 10$ )
6. Go to city 9 to city 10 (after that,  $t = 13$ )

In the second test case, the road from 1 to 2 will be blocked from time  $t = 0$  to time  $t = 3$ . Therefore, it can only be used at times  $t \geq 3$ . After waiting, and after another unit of time crossing the road, the total time will be  $t = 4$  units of time.

# Enunciados en Español

## Contents

1 X Maratón UFPS - y I Maratón Interuniversitaria UFPS+UTP+Amazonía	2
2 Grupo de Estudio en Programación Competitiva	4
3 Instrucciones	4
4 Reglas	4
5 Lista de Problemas	6
6 Fe de Erratas	7



## 1 X Maratón UFPS - y I Maratón Interuniversitaria UFPS+UTP+Amazonía

Desde el año 2014, el programa Ingeniería de Sistemas de la Universidad Francisco de Paula Santander (UFPS) inició un interesante proceso para promover la Programación Competitiva, como parte de las actividades del Semillero SILUX (Linux, Software Libre y Licencias Abiertas). El propósito fundamental fue el desarrollo de competencias de pensamiento computacional, como resolución de problemas, programación de computadores, trabajo colaborativo y liderazgo.

Los pioneros de dicho proyecto fueron dos estudiantes, quienes ya se graduaron y dejaron como herencia una gran motivación. Ahora siguen colaborando con el Semillero que es liderado directamente por los estudiantes, quienes ingresan desde primer semestre con el entusiasmo que trae el sueño de llegar algún día a la Competencia Mundial ICPC (The International Collegiate Programming Contest <https://icpc.baylor.edu/>).

Desde entonces, cada año, el evento más importante es la Maratón Interna de Programación de la UFPS, que propone un conjunto de retos para poner a prueba las habilidades en algoritmia, programación y trabajo en grupo de los estudiantes. En dicho evento un actor fundamental siempre ha sido la Red de Programación Competitiva (RPC), quien apoya toda la logística de preparación de la competencia y además ofrece su plataforma tecnológica y su equipo de trabajo. El lema de RPC es “Aquí crecemos juntos” y la UFPS ya lleva diez (10) años creciendo en Programación Competitiva junto a muchas universidades en varios países de toda Latinoamerica.

El el año 2022, con el apoyo de la Empresa de Tecnología Pragma, realizamos un bonito evento llamado Primer Training Camp de los Santanderes, donde juntamos varias universidades de Santander (UIS y UDI) y Norte de Santander (Ocaña, Pamplona, UDES, Simón Bolívar, Remington), junto con UTP y U. Amazonía.

Y para este año 2023, nos juntamos UTP, UFPS y U. Amazonía para la primera Maratón Interuniversitaria. Estaremos en simultánea, presenciales, en Cúcuta, Pereira y Florencia y en la red estarán los equipos desde Mexico hasta Argentina para cerrar el año 2023 y soñar con el año 2024.

Este año nos complace presentar un conjunto de quince (15) problemas, todos inéditos, escritos por estudiantes, profesores y graduados de varias universidades. Por la pandemia del COVID19, fueron dos años virtuales y este es el segundo año nuevamente presenciales, algo muy importante, porque regresan los globos y las fotografías que registran la energía y alegría de nuestros estudiantes.

Reconocimiento y agradecimiento especial al equipo de trabajo de este año:

- Hugo Humberto Morales Peña - Profesor Universidad Tecnológica de Pereira, Colombia
- Milton Jesús Vera Contreras - Profesor UFPS (desde Cúcuta)
- Diana Espinosa y su semillero - Universidad de la Amazonía, Colombia
- José Manuel Salazar Meza - Graduado UFPS (desde Cúcuta)
- Carlos Alberto Salazar Meza - Estudiante UFPS (desde Cúcuta)
- María Alexandra Velásquez Jaimes - Estudiante UFPS (desde Cúcuta)
- Luis Daniel Carreño Pitalua - Estudiante UFPS (desde Cúcuta)
- Santiago Felipe Alferez Villamizar - Estudiante UFPS (desde Cúcuta)

- Eddy Miguel Ramirez - Profesor (desde Costa Rica)
- Equipo Red de Programación Competitiva (Fabio Avellaneda, Johany Careño)
- Estudiantes, Compañeros Profesores y Directivos de las universidades organizadoras UFPS, UTP y Universidad de la Amazonía.

Este trabajo se comparte bajo licencia Creative Commons Reconocimiento-Compartir Igual 4.0 Internacional (CC BY-SA 4.0)





## 2 Grupo de Estudio en Programación Competitiva

El grupo de estudio en Programación Competitiva hace parte del Semillero de Investigación SILUX (Linux, Software Libre y Licencias Abiertas) y tiene como fin preparar y fortalecer a los estudiantes de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander en competencias de resolución de problemas, algoritmia, programación de computadores, trabajo colaborativo y liderazgo. Se aprovecha el entorno competitivo o gamificación porque favorece el aprendizaje tanto de habilidades hard como habilidades soft.

Los integrantes del grupo de estudio han participado en la Maratón Nacional de Programación desde el año 2015, logrando por 9 años consecutivos la clasificación a fase Regional Latinoamericana. En las dos últimas competencias de 2022 y 2023 la UFPS se ha logrado ubicar en el top 3 y top 6.

## 3 Instrucciones

Puedes utilizar Java, C, C++ o Python, teniendo en cuenta:

1. Resuelve cada problema en un único archivo. Debes enviar a la plataforma únicamente el archivo .java, .c, .cpp, o .py que contiene la solución.
2. Todas las entradas y salidas deben ser leídas y escritas mediante la entrada estándar (Java: Scanner o BufferedReader) y salida estándar (Java: System.out o PrintWriter).
3. En java, el archivo con la solución debe llamarse tal como se indica en la linea "Source file name" que aparece debajo del título de cada ejercicio. En los otros lenguajes es opcional (puede tener cualquier nombre).
4. En java, la clase principal debe llamarse igual al archivo (Si el Source File Name o basename indica que el archivo debe llamarse example.java, la clase principal debe llamarse example).
5. En java, asegúrate de borrar la linea "package" antes de enviar.
6. Tu código debe leer la entrada tal cual se indica en el problema, e imprimir las respuestas de la misma forma. No imprimas líneas adicionales del tipo "La respuesta es..." si el problema no lo solicita explícitamente.
7. Si tu solución es correcta, en unos momentos la plataforma te lo indicará con el texto "YES". En caso contrario, obtendrá un mensaje de error.

## 4 Reglas

1. Gana el equipo que resuelve más problemas. Entre dos equipos que resuelvan el mismo número de problemas, gana el que los haya resuelto en menos tiempo (ver numeral 2).
2. El tiempo obtenido por un equipo es la suma de los tiempos transcurrido en minutos desde el inicio de la competencia hasta el momento en que se envió cada solución correcta. Habrá una penalización de 20 minutos que se suman al tiempo por cada envío erróneo (esta penalización solo se cuenta si al final el problema fue resuelto).
3. NO se permite el uso de internet durante la competencia. Únicamente se puede acceder a la plataforma en la cual se lleva a cabo la competencia.

4. NO se permite el uso de dispositivos electrónicos durante la competencia.
5. NO se permite la comunicación entre miembros de equipos diferentes. Cada integrante solo puede comunicarse con sus dos compañeros de equipo.
6. Se permite todo tipo de material impreso (libros, fotocopias, apuntes, cuadernos, guías) que el equipo desee utilizar.



## 5 Lista de Problemas

A continuación la lista de los quince (15) problemas a resolver. Un primer reto y logro es resolver alguno de los problemas. Un segundo reto es lograr resolver cualquiera de los problemas más rápido que todos los demás. Un tercer reto es lograr resolver todos los problemas. Un cuarto reto es lograr ubicarse en el top 3 de la competencia. Un quinto reto es lograr ubicarse en el top 5 o ser el mejor equipo novato o el mejor equipo femenino ;)

Pero recuerda que solo al competir ya se es ganador ;)

1. A Love Calculator (Page 8)
2. Betting on Luck (Page 9)
3. Conner's Towers (Pages 10-11)
4. Dictionary (Pages 12)
5. Emergency in Treenity (Pages 13-14)
6. For Honor Reasons (Pages 15-16)
7. Guardians (Pages 17-18)
8. Hemerology (Pages 19-20)
9. I Love Python (Page 21)
10. Judging Your Tetris Skills (Pages 22-24)
11. Kepler Barriers (Pages 25-26)
12. Longest Periodic Substring (Page 27)
13. Matchsticks Puzzle (Pages 28-29)
14. Night of the Little Candles (Pages 30-31)
15. OMG!!! A Rainy Road (Pages 32-33)

**Nota 1:** Los enunciados de los problemas están en idiomas inglés y español.

**Nota 2:** A continuación, en la siguiente página, encontrará el detalle de tiempos límite en segundos para C++, Java y Python, para cada caso de prueba, conforme a las pruebas realizadas por el equipo de trabajo de RPC y UFPS, usando el servidor de Boca de RPC y UFPS. Tenga en cuenta que los problemas son juzgados de manera automática con varios casos de prueba.

TimeLimit				
Letra	Nombre	C, C++	Java	Python
A	A Love Calculator	1	1	1
B	Betting On Luck	2	3	5
C	Conner's Towers	1	7	3
D	Dictionary	3	6	7
E	Emergency in Treenity	2	5	5
F	For Honor Reasons	2	5	5
G	Guardians	1	5	5
H	Hemerology	1	1	1
I	I love Python	1	1	1
J	Judging Your Tetris Skills	1	1	1
K	Kepler Barriers	2	10	10
L	Longest Periodic Substring	2	5	5
M	Matchsticks Puzzle	3	3	1
N	Night of the little candles	1	1	1
O	OMG!!! A Rainy Road	1	5	2

## 6 Fe de Erratas

**Errare humanum est**, lo importante es reconocer y corregir el error. Si durante esta competencia se presentan errores, aparecerán aquí y se actualizarán en linea, informando a través de la plataforma:

1. En el problema G se corrigió la versión de inglés, que tenía una pequeña diferencia respecto a la versión en español. Este ajuste no afectó a los participantes.
2. En el problema F se ajustó el ejemplo de entradas y salidas, para consistencia con una aclaración del enunciado. Este ajuste no afectó a los participantes.
3. En el problema F el basename estaba Honor en el cuadernillo, pero era Fhonor en Boca. Este ajuste no afectó a los participantes.
4. En el problema E había un error en un caso de prueba, gracias a los amigos mundialistas de Argentina por la clarificación. Se juzgó de nuevo el reto y no se afectó el resultado final de la competencia.

Entendemos la inconformidad que estos errores generan, pero también confiamos en su comprensión. No es una tarea sencilla llevar a cabo una maratón de este tipo. Se requiere la colaboración voluntaria de muchas personas, quienes se esfuerzan mucho más allá de sus obligaciones. Y es algo que hemos logrado mantener durante diez años entre la UFPS y RPC, siempre con mucho entusiasmo y cariño para mantener viva la llama de la Programación Competitiva.

## Problem A. A Love Calculator

Source file name: Alove.c, Alove.cpp, Alove.java, Alove.py

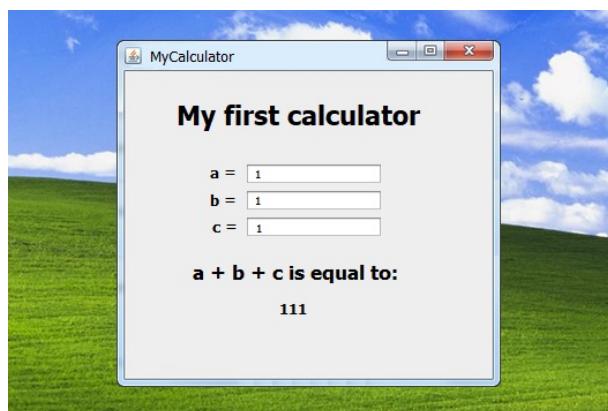
Input: standard input

Output: standard output

Author(s): Carlos Alberto Salazar Meza - UFPS (Estudiante)

María es una estudiante de la Universidad Francisco de Paula Santander conocida por ser la alumna favorita del profesor Eduardo, un académico de fama mundial con innumerables logros.

Por eso, cuando María reprobó su primer examen con Eduardo, este amable profesor se ofreció a darle una nota extra si ella hacía un trabajo especial: crear una calculadora que sume tres números  $a$ ,  $b$  y  $c$ . María aceptó la propuesta y se puso manos a la obra, la semana siguiente presentó el siguiente programa:



La suma de  $a = 1$ ,  $b = 1$  y  $c = 1$  es igual a 111.

Todos en la clase quedaron sorprendidos, pero se sorprendieron aún más al ver que el famoso profesor Eduardo felicitó a María por crear una calculadora de concatenación, y decidió plantear el siguiente problema para su clase:

Hay tres números enteros, debes asignar cada uno de estos valores a  $a$ ,  $b$  y  $c$  de tal manera que la calculadora de María genere el mayor resultado. Un resultado  $x$  es mayor que otro resultado  $y$  si el valor decimal de  $x$  es mayor que el valor decimal de  $y$ .

¿Puedes ayudar a la clase a dominar la calculadora de María y resolver el problema?

### Input

La única línea contiene tres números enteros  $x$ ,  $y$  y  $z$  ( $0 \leq x, y, z < 10^7$ ) sin ceros a la izquierda, los números que deben asignarse a  $a$ ,  $b$  y  $c$ .

Se garantiza que el resultado de la calculadora de María será inferior a 10 dígitos.

### Output

Imprime una línea con el mayor resultado posible obtenido al ingresar los datos en la calculadora.

### Examples

Input	Output
1 4 12415	4124151
21 789 12	7892112

## Problem B. Betting on Luck

Source file name: Betting.c, Betting.cpp, Betting.java, Betting.py  
 Input: standard input  
 Output: standard output  
 Author(s): Carlos Alberto Salazar Meza - UFPS (Estudiante)

Alexa y Beto son los dos apostadores más famosos de Barcelona y están buscando una nueva forma de malgastar su dinero y juventud. En esta búsqueda, encontraron el “juego de dados definitivo”, que consta de las siguientes reglas:

- El juego consta de  $n$  rondas.
- Alexa tiene un multiplicador  $m$  que inicialmente es igual a 1.
- En una ronda, cada jugador tira dos dados y suma los valores obtenidos,  $d$  es igual a la diferencia absoluta entre la suma de Alexa y la suma de Beto.
- Si la suma de Alexa es mayor que la suma de Beto, ella gana la ronda y debe elegir exactamente una de las siguientes acciones:
  1. Cambiar el valor de  $m$  por  $d$ .
  2. Sumar  $m * d$  a su puntuación y restablece el valor de  $m$  a 1.
- Si la suma de Beto es mayor o igual que la suma de Alexa, él gana la ronda, así que suma  $d$  a su puntuación y restablece el valor de  $m$  a 1.

Después de terminar la ronda número  $n$ , cada jugador recibe del otro una cantidad de dinero igual a su puntuación.

Alexa es una gambler que toma riesgos, por lo que cuando ella gana una ronda, cuanto mayor sea el valor de  $d$ , mayor será la probabilidad de que cambie el valor de  $m$  buscando obtener más puntos. Para ser más precisos, la probabilidad  $p$  de que Alexa cambie el valor de  $m$  está dada por  $p = \frac{d}{10}$ . Además, ella está tan concentrada en el juego que no tiene en cuenta el número de turno al momento de elegir su acción.

Alexa está pasando por una mala racha, por eso le gustaría saber el valor esperado de la puntuación final que tendrá cada jugador después de  $n$  rondas para saber si vale la pena apostar o no, ¡puedes ayudarla?

Se garantiza que, en los casos de prueba dados, los valores esperados requeridos se pueden representar como una fracción irreducible  $\frac{x}{y}$  donde  $x$  y  $y$  son números enteros y  $y \not\equiv 0 \pmod{10^9 + 7}$ .

### Input

La entrada contiene un número entero  $n$  ( $1 \leq n \leq 2 * 10^5$ ), el número de rondas que jugarán Alexa y Beto.

### Output

Imprime el valor esperado de la puntuación final de Alexa y Beto modulo  $1000000007$  ( $10^9 + 7$ ).

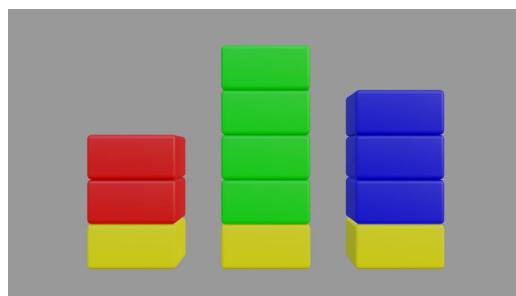
### Examples

Input	Output
3	872293200 782407417
6	164529281 564814827

## Problem C. Conner's Towers

Source file name: Conner.c, Conner.cpp, Conner.java, Conner.py  
Input: standard input  
Output: standard output  
Author(s): Eddy Ramírez - UNA & TEC Costa Rica (Profesor)

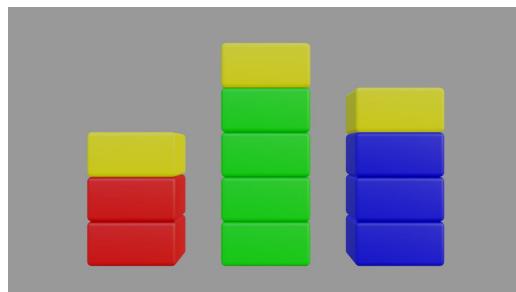
Conner es un bebé al que le gusta mucho jugar con piezas de madera. Para mantenerlo ocupado, su papá le ha puesto un reto, dada una cantidad  $A$  de piezas rojas,  $B$  de piezas verdes y  $C$  de piezas azules, va a apilar las piezas por color. Además en la base de cada pila pondrá una pieza amarilla. Por ejemplo, la posición inicial para  $A = 2$ ,  $B = 4$  y  $C = 3$  sería tal como se muestra en la imagen:



Posición inicial

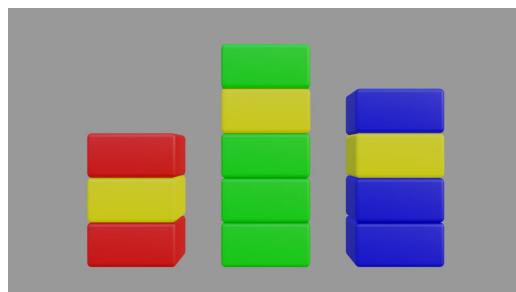
A partir de aquí, una “jugada” consiste en tomar la pieza que se encuentra en las tres bases y ponerlas arriba de cada pila de piezas.

De modo que, para el caso descrito, la posición tras la jugada 1 sería como se muestra:



Posición jugada 1

Y tras la jugada 2 la posición sería:



Posición jugada 2

Conner además es un bebé que empieza a notar colores y le gusta mucho el amarillo, una jugada es considerada exitosa por Conner, si en alguna base queda una pieza amarilla.

Haga un programa en que dados los valores  $A$ ,  $B$ ,  $C$  y  $N$  (siendo  $N$  el número de jugadas) indique cuántas jugadas serían exitosas para Conner. La posición inicial no es considerada como exitosa.

## Input

Su entrada consiste en múltiples casos de prueba. La primer línea contiene un número  $T$  con  $1 \leq T \leq 10^5$  indicando el número de casos de prueba.

Las siguientes  $T$  líneas contienen 4 números separados por espacio:  $A, B, C, N$  con  $0 \leq A, B, C, N \leq 10^{18}$ . Indicando la cantidad de piezas rojas, verdes y azules respectivamente y siendo  $N$  la cantidad de jugadas que se van a realizar.

## Output

Por cada caso de prueba debe imprimir en una línea únicamente un número con la cantidad de jugadas que Conner consideraría exitosas.

## Examples

Input	Output
2	15
100 101 0 15	7
2 3 4 10	



## Problem D. Dictionary

Source file name: Dictionary.c, Dictionary.cpp, Dictionary.java, Dictionary.py

Input: standard input

Output: standard output

Author(s): Hugo Humberto Morales Peña - UTP (Profesor)

Se cuenta con un diccionario de palabras (donde obviamente no hay palabras repetidas) sobre el cual se quiere consultar la cantidad de palabras que se encuentran en el rango entre dos palabras, con base en el ordenamiento lexicográfico.

Por ejemplo, si se tiene un diccionario con el siguiente conjunto de palabras:

{camila, angel, margot, bambi, sabrina, mabel, celia}

y se pregunta por la cantidad de palabras que se encuentran en el rango entre las palabras: candy y patrick, con base en el ordenamiento lexicográfico, la respuesta sería 3 porque las palabras que están en el rango según su orden lexicográfico son: celia, mabel y margot.

### Input

La entrada consiste de un único caso de prueba. La primera línea contiene dos números enteros positivos  $n$  y  $q$  ( $1 \leq n \leq 10^6$ ,  $1 \leq q \leq 5 \cdot 10^5$ ) los cuales representan respectivamente la cantidad de palabras en el diccionario y el total de consultas a realizarse sobre el diccionario.

Luego se presentan  $n$  líneas cada una conteniendo una palabra de máximo 10 letras minúsculas en el alfabeto del idioma Inglés.

Por último, en el caso de prueba son presentadas  $q$  líneas, cada una de ellas presentando dos palabras distintas  $w_1$  y  $w_2$  (separadas por un espacio en blanco), cada palabra es de máximo longitud 10 y consisten únicamente de letras minúsculas del alfabeto del idioma Inglés. Se garantiza que la palabra  $w_1$  está primero que la palabra  $w_2$  en el orden lexicográfico.

### Output

La salida del problema debe contener  $q$  líneas cada una de ellas conteniendo un número entero no negativo que representa la cantidad de palabras que están en el rango  $[w_1, w_2]$  en el orden lexicográfico de las palabras del diccionario.

Input	Output
7 9	3
camila	7
angel	1
margot	7
bambi	3
sabrina	2
mabel	0
celia	4
candy patrick	2
angel sabrina	
candy conner	
angel vivian	
celia penny	
daisy margot	
daisy india	
ada lara	
lara penny	

## Problem E. Emergency in Treenity

Source file name: Emergency.c, Emergency.cpp, Emergency.java, Emergency.py

Input: standard input

Output: standard output

Author(s): María Alexandra Velásquez Jaimes - UFPS (Estudiante)

Lejos de cualquier civilización, en uno de los lugares más misteriosos de la tierra, se encuentra el reino de las hadas, Treenity. Este antiguo e inmenso reino tiene la particularidad de que puede ser representado como un árbol de  $n$  nodos, donde el nodo 1 corresponde a la capital y los restantes son ciudades mágicas. Las conexiones entre estas ciudades se establecen mediante senderos mágicos, específicamente hay  $n - 1$  senderos que interconectan todo el reino. El costo de usar un sendero  $i$  entre dos ciudades es de  $w_i$  unidades de polvo mágico.

Treenity había disfrutado de siglos de tranquilidad, pero esta paz se vio interrumpida por una emergencia que podría marcar el fin de su historia: un poderoso hechicero que desea la aniquilación total de este reino mágico. En respuesta, la reina de las hadas decidió trabajar incansablemente día y noche para crear un hechizo de protección, pero debido al poco tiempo disponible ella no pudo completarlo, por lo que solo surge efecto en las ciudades tales que el costo de polvo mágico para llegar a ellas desde la capital es par, (el costo de polvo mágico para ir desde la capital a una ciudad  $u$  es igual a la suma de los costos de polvo mágico en el camino de la capital a la ciudad  $u$ ).

Por suerte, la reina tiene un hechizo para cambiar la distribución de las ciudades en su reino, el cual es capaz de eliminar el sendero que une un par de ciudades  $u$  y  $p$  y crear uno nuevo entre la ciudad  $u$  y otra ciudad  $v$ , con un nuevo costo de unidades de polvo mágico, y sin afectar la particularidad del reino de ser representado como un árbol. Aquí  $p$  es el nodo más cercano a  $u$  en el camino entre  $u$  y la capital. En otras palabras,  $p$  es el padre directo de  $u$  en el árbol.

Para salvar a Treenity, la reina comienza a usar su hechizo de cambio de distribución y te pide a ti, su sirviente más leal, que cuentes el número de ciudades que no están protegidas por su hechizo de protección, es decir, aquellas en las que el costo de polvo mágico para llegar a ellas desde la capital es impar.

En esta situación pueden ocurrir dos tipos de eventos:

- Tipo 1: La reina usa su hechizo de cambio de distribución para eliminar el sendero entre el nodo  $u$  y  $p$  (su padre directo), y crear un nuevo sendero entre  $u$  y  $v$  con un nuevo costo de polvo mágico  $w$ .
- Tipo 2: La reina te solicita el número de ciudades que no están protegidas por su hechizo de protección.

### Input

La primera línea contiene un número entero  $n$  ( $3 \leq n \leq 10^5$ ), – el número de ciudades mágicas en el reino de Treenity.

Las siguientes  $n - 1$  líneas contienen tres números enteros  $u$ ,  $v$  y  $w$  ( $1 \leq u, v \leq n$ ,  $1 \leq w \leq 10^9$ ) indicando que hay un sendero con costo  $w$  de polvo mágico entre las ciudades  $u$  y  $v$ .

La siguiente línea contiene un número entero  $q$  ( $1 \leq q \leq 10^5$ ) – el número de eventos.

Cada una de las siguientes  $q$  líneas comienza con un número entero  $t_i$  ( $1 \leq t_i \leq 2$ ), que representa el tipo de evento.

Si  $t_i = 1$ , entonces el  $i$ -ésimo evento es un hechizo de cambio de distribución de la reina, y le siguen tres enteros  $u$ ,  $v$  y  $w$  ( $1 \leq u, v \leq n$ ,  $(u \neq 1)$ ,  $1 \leq w \leq 10^9$ ) representando un hechizo de cambio de distribución.

Si  $t_i = 2$ , entonces el  $i$ -ésimo evento es una solicitud de la reina.



## Output

Por cada evento de tipo 2 debes imprimir una única línea con el número de ciudades que en ese momento no están protegidas por el hechizo de protección de la reina.

## Examples

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

## Problem F. For Honor Reasons

Source file name: Forhonor.c, Forhonor.cpp, Forhonor.java, Forhonor.py  
Input: standard input  
Output: standard output  
Author(s): Carlos Alberto Salazar Meza - UFPS (Estudiante)

Los reinos del continente de Endor, famosos por vivir en constante conflicto, finalmente desean una era de paz y están haciendo todos los preparativos para que esto suceda, por lo que contrataron a un consejo de sabios que determinarán solicitudes para establecer alianzas entre los reinos con el fin de lograr la paz. Además te contrataron a ti, el mejor mensajero del continente, para entregar las solicitudes entre ellos.

A pesar de desear una era de paz, los habitantes del continente son los más orgullosos y arrogantes del planeta y nunca aceptarán crear una alianza con un reino con el cual estuvieron en guerra en el pasado, ni con un reino que sea aliado de un reino enemigo. Si les llegara una solicitud de este tipo, ellos lo verán como una burla en contra de su honor y empezarán una guerra en el acto.

Los sabios insisten en continuar con el proceso sin tener en cuenta el orgullo de los habitantes. Así que para evitar un desastre, tú decides no entregar aquellas solicitudes que causarán una guerra, y contactas al famoso historiador Selram para saber exactamente en qué casos ocurriría esto.

Selram te da un listado de las guerras que ocurrieron en el pasado y te dice lo siguiente:

Existen  $n$  reinos en Endor, enumerados de 1 a  $n$ . Sean  $A$  y  $B$  los reinos seleccionados por el consejo para llevar a cabo una solicitud de alianza,  $X$  un reino que pertenece a la misma alianza que  $A$  y  $Y$  un reino que pertenece a la misma alianza que  $B$ , se desatará una guerra si al menos una de las siguientes condiciones se cumple:

- $A$  y  $B$  estuvieron en guerra en el pasado
- $A$  y  $Y$  estuvieron en guerra en el pasado
- $X$  y  $B$  estuvieron en guerra en el pasado
- $X$  y  $Y$  estuvieron en guerra en el pasado

Con esta información en mano puedes comenzar con tu tarea de entregar las solicitudes de alianza determinadas por el consejo de sabios. Pero antes de eso, Selram necesita que lo ayudes con su registro de hechos históricos importantes, por lo que para cada solicitud te pide que le digas si no fue entregado para evitar una guerra, si es innecesario porque los reinos ya pertenecen a la misma alianza, o si debería entregarse para fortalecer la paz en el continente.

### Input

La primera línea consiste de dos enteros  $n$  y  $m$  ( $2 \leq n \leq 10^5$ ,  $0 \leq m \leq 2 * 10^5$ ), que indican la cantidad de reinos en el continente y la cantidad de guerras en la lista de Selram.

Las siguientes  $m$  líneas contienen dos enteros  $u$  y  $v$  – los identificadores de los reinos que estuvieron en guerra en el pasado.

La siguiente línea consta de un entero  $q$  ( $1 \leq q \leq \min(2 * 10^5, \frac{n*(n-1)}{2})$ ), la cantidad de solicitudes de alianza que determinó el consejo de sabios.

Las siguientes  $q$  líneas contienen dos enteros  $u$  y  $v$  – los identificadores de los reinos implicados en la solicitud de alianza determinada por el consejo de sabios. Las solicitudes de paz están dadas en orden cronológico. Se garantiza que el consejo de sabios no determinará más de una solicitud de alianza entre un mismo par de reinos.



## Output

Por cada solicitud de alianza determinada por los sabios, imprime una línea que contenga la respuesta según los siguientes casos:

- “**AVOID WAR**“ si esa solicitud de alianza causará una guerra.
- “**NEXT**“ si esa solicitud de alianza es innecesaria.
- “**FOR THE PEACE**“ si esa solicitud de alianza debe ser entregada.

## Examples

Input	Output
5 1	FOR THE PEACE
1 2	FOR THE PEACE
6	AVOID WAR
1 3	AVOID WAR
2 4	FOR THE PEACE
3 4	NEXT
1 2	
1 5	
3 5	

## Problem G. Guardians

Source file name: Guardians.c, Guardians.cpp, Guardians.java, Guardians.py

Input: standard input

Output: standard output

Author(s): Juan José Ortiz Plaza - UNIAMAZONIA (Student)

En un reino distante, los héroes Yari y Flora se embarcan en una aventura épica a través de un misterioso laberinto lleno de mazmorras. El laberinto se puede representar como un árbol con  $n$  nodos, donde cada nodo  $i$  es una mazmorra que está protegida por un guardián cuya vitalidad está representada por un número entero  $h_i$ . Además, sólo hay una manera de entrar y salir del laberinto, esto es a través de dos portales especiales ubicados en los nodos  $a$  y  $b$ , que inicialmente son desconocidos por ellos.

Yari y Flora son héroes extremadamente capaces, pero tienen un defecto: son muy cobardes. Por ello, crearon un plan especial para evitar ser emboscados en el camino o quedar atrapados dentro de una mazmorra, el cual tiene dos reglas:



- Solo visitar mazmorras en el camino de  $a$  a  $b$ .
- Solo se puede explorar una mazmorra si la vitalidad del guardián que la protege es menor o igual que su habilidad combinada  $x$ , en caso contrario, ellos no exploran esa mazmorra y continúan su camino.

Además, como medida preventiva con el fin de preparar suficientes suministros para su aventura, quieren hacer algunas simulaciones para encontrar el número máximo de mazmorras que podrían explorar si el nodo de entrada fuera  $a$  y el nodo de salida fuera  $b$ . ¿Puedes ayudarlos?

### Input

La primera línea contiene dos números enteros  $n$  y  $q$  ( $1 \leq n, q \leq 10^5$ , el número de mazmorras y el número de simulaciones que Yari y Flora harán, respectivamente).

La segunda línea contiene  $n$  enteros  $h_1, h_2, \dots, h_n$  ( $1 \leq h_i \leq 10^9$ ), que representan la vitalidad del guardián que protege la  $i$ ésima mazmorra.

Las siguientes  $n - 1$  líneas describen las conexiones del laberinto. Cada línea contiene dos números enteros  $u$  y  $v$  ( $1 \leq u, v \leq n, u \neq v$ ), indicando que hay una ruta entre las mazmorras  $u$  y  $v$ .

Las siguientes  $q$  líneas contienen tres números enteros  $a$ ,  $b$  y  $x$  ( $1 \leq a, b \leq n, 1 \leq x \leq 10^9$ ), que representan los nodos de entrada y salida del laberinto en la simulación, y la habilidad combinada de Yari y Flora, respectivamente.

### Output

Para cada simulación imprime el número máximo de mazmorras que pueden ser exploradas por Yari y Flora.



## Examples

Input	Output
10 3	1
10 5 4 9 2 3 8 7 9 4	1
1 4	2
1 6	
1 5	
4 10	
5 2	
5 8	
4 9	
1 3	
5 7	
6 2 2	
10 8 2	
2 5 8	

## Problem H. Hemerology

Source file name: Hemero.c, Hemero.cpp, Hemero.java, Hemero.py  
Input: standard input  
Output: standard output  
Author(s): Milton Jesús Vera Contreras - UFPS (Professor)

La Hemerología es el “estudio de los calendarios, tanto en sus aspectos astronómicos, técnicos, históricos o religiosos” (Wikipedia). Es abrumadora la variedad de calendarios: juliano, musulmán, gregoriano, turco, chino, ruso, egipcio, azteca, inca, maya y muchos más. Y es maravilloso todo lo que se aprende al estudiar hemerología, como la siguiente rima en español para nuestro calendario:

Treinta días tiene noviembre  
con abril, junio y septiembre,  
los demás de treinta y uno,  
excepto febrero mocho,  
que sólo tiene veintiocho  
y veintinueve funesto  
cuando es un año bisiesto.

Incluso se puede aprender sobre Ciencias de la Computación con hemerología. Por ejemplo, el profesor Donald Knuth menciona y explica varios algoritmos de calendarios, ojalá los hayas leído o los leas alguna vez.

En la práctica, un calendario se puede simplificar a tres conceptos: días, meses y años. Los días se agrupan formando meses, los meses se agrupan formando años, y los años avanzan sin límites hasta el fin de los tiempos y sin los raros años bisiestos. Siguiendo esta simplificación, es sencillo calcular el tiempo transcurrido entre dos fechas dadas.

¿Podrías escribir, lo más rápido posible, un programa eficiente que calcule la diferencia entre dos fechas (día, mes y año) de cualquier calendario ficticio?

### Input

La entrada consiste en tres líneas: la primera línea tiene  $n + 1$  números (separados por un espacio en blanco) que describen un calendario ficticio: el primer número  $1 \leq n \leq 10^6$  es la cantidad de meses de un año, y después  $n$  números  $1 \leq m_1, m_2 \dots m_{n-1}, m_n \leq 10^6$ , con la cantidad de días de cada uno de los meses del año de ese calendario ficticio.

La segunda línea tiene la fecha más antigua, que consiste en tres números (separados por un espacio en blanco):  $1 \leq day_1 \leq 10^6$ ,  $1 \leq month_1 \leq 10^6$ , y  $1 \leq year_1 \leq 10^6$ .

Y la tercera línea tiene la fecha más reciente, que también consiste en tres números (separados por un espacio en blanco):  $1 \leq day_2 \leq 10^6$ ,  $1 \leq month_2 \leq 10^6$ , y  $1 \leq year_2 \leq 10^6$ .

Las dos fechas siempre serán válidas en ese calendario ficticio.

### Output

Un número  $t$  con el total de días transcurridos desde la fecha más antigua hasta la fecha más reciente, según el calendario ficticio.



## Examples

Input	Output
12 31 28 31 30 31 30 31 31 30 31 30 31 25 11 2023 25 11 2023	0
12 31 28 31 30 31 30 31 31 30 31 30 31 25 11 2023 2 12 2023	7
12 31 28 31 30 31 30 31 31 30 31 30 31 1 1 2023 31 12 2023	364
12 31 29 31 30 31 30 31 31 30 31 30 31 1 1 2024 31 12 2024	365
12 31 28 31 30 31 30 31 31 30 31 30 31 1 1 2000 31 12 2010	4014
3 1000000 1000000 1000000 1 1 1000000 1000000 3 1000000	2999999
1000000 "1000000 times 1000000" 1 1 1 1000000 1000000 1000000	99999999999999999999

## Note

Use Fast I/O

## Problem I. I Love Python

Source file name: Ilove.c, Ilove.cpp, Ilove.java, Ilove.py

Input: standard input

Output: standard output

Author(s): Milton Jesús Vera Contreras - UFPS (Professor)

Algunos estudiantes aman Python, porque pueden resolver problemas con una sola línea de código. Y odian Java y C++, porque deben escribir mucho código y a veces mal código. Ellos son todo lo contrario a este meme:



$$\begin{aligned}
 a^n &= \sum_{k=0}^n f(x)^k = a_0 + \sum_{k=1}^n a_k x^k \\
 &= \pi r^2 \cdot f(x) = mc^2 \quad e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \dots + \frac{x^n}{n!} \\
 &\quad - 2ab + b^2 \quad E = mc^2 \quad e^{-x} = 1 - \frac{x}{1!} + \frac{x^2}{2!} - \dots - \frac{x^n}{n!} \\
 &\quad \frac{\partial^2}{\partial t^2} - \frac{\partial^2}{\partial x^2} \\
 &E = mc^2 \sin a \pm \sin b = 2 \sin \frac{a+b}{2} \cos \frac{a-b}{2} \quad (1+x)^n = 1 + nx + \frac{n(n-1)}{2!}x^2 + \dots + \frac{n(n-1)(n-2)\dots(1)}{n!}x^n \\
 &\quad A = \pi r^2 \cdot \cos^2 a + \cos^2 b - (1-x)^2 = 1 - 2x + x^2 \\
 &\beta = a^2 + 2ab + b^2 \quad A = \pi r^2 (1-a^2 - b^2) + \dots + \frac{(-1)^n}{n!}x^n = 1 - 2x + x^2 + \dots + (-1)^n x^n \\
 &\quad + \cos \beta = 2 \cos^2 \left( \frac{a+b}{2} \right) = 1 + \frac{nx}{1!} + \frac{n(n-1)x^2}{2!} + \dots + \frac{n(n-1)(n-2)\dots(1)}{n!}x^n \\
 &\quad \text{etc.} \quad (1+x)^n = 1 + \frac{-2x}{1!} + \frac{-2x+2x^2}{2!} + \dots + \frac{(-1)^n(1-n)}{n!}x^n = 1 - 2x + x^2 + \dots + (-1)^n x^n \\
 &\quad x = \frac{-2x \pm \sqrt{4x^2 - 4a^2}}{2a} = \frac{x \pm \sqrt{x^2 - a^2}}{a} \\
 &\quad a(\alpha_{xx} + \alpha_{yy}) \quad (a-b)^2 = a^2 - 2ab + b^2 \\
 &\quad -2x^2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \\
 &\quad b^2 = c
 \end{aligned}$$

## Solve with Math Equation

```
if a > b:  
    case = 1  
else:  
    if a < b:  
        case = 2  
    else:  
        if a == b:  
            case = 3  
        else:  
            if a != b:  
                case = 4  
            else:  
                if otherCase:  
                    case = 5  
                else:  
                    #OMG!!!
```

## Hard-coding IF/ELSE

Pensando en esos estudiantes, el profesor propuso este reto como el más sencillo de la competencia:

Se tiene un número entero  $n$  ( $0 \leq n \leq 10^{17}$ ) y un dígito  $d$  ( $0 \leq d \leq 9$ ) y se quiere determinar el número  $m > n$  más cercano a  $n$  cuyo último dígito (dígito más a la derecha) sea  $d$ .

Demuestra al profesor tu amor por Python, resolviendo este problema con la menor cantidad de líneas ;)

## Input

Dos números separados por un espacio:  $n$  ( $0 \leq n \leq 10^{17}$ ) y  $d$  ( $0 \leq d \leq 9$ ).

## Output

El número  $m > n$  más cercano a  $n$  cuyo último dígito (dígito más a la derecha) sea  $d$ .

## Examples

Input	Output
15 4	24
30 1	31
151 1	161
922337203685477581 0	922337203685477590

## Problem J. Judging Your Tetris Skills

Source file name: Tetris.c, Tetris.cpp, Tetris.java, Tetris.py  
Input: standard input  
Output: standard output  
Author(s): Milton Jesús Vera Contreras - UFPS (Professor)

El Tetris es un videojuego muy importante en la historia de la tecnología, al punto que aparece en la película Pixels, protagonizada por Adam Sandler, Kevin James, Peter Dinklage, Josh Gad y la hermosa Michelle Monaghan. En dicha película se plantea un caso hipotético de invasión alienígena, en la que los videojuegos cobran vida dentro del mundo humano.

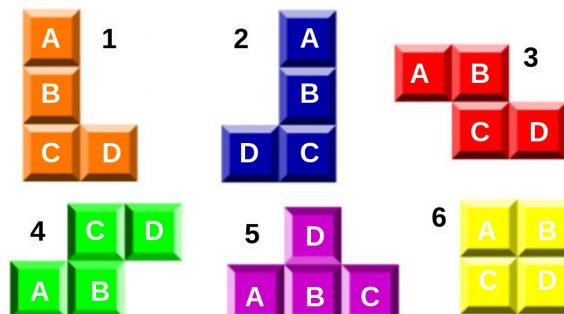


Pensando en dicha película, en el auge del Metaverso, en las ideas apocalípticas de la Inteligencia Artificial y en lo difícil que resulta convencer a los estudiantes de abandonar Windows y el Mouse para usar Linux y las consolas de comandos basadas en texto, el profesor buscó un examen de programación de computadores del año 2004, ya casi 20 años atrás, y lo adaptó para la X Maratón UFPS+RPC 2023.

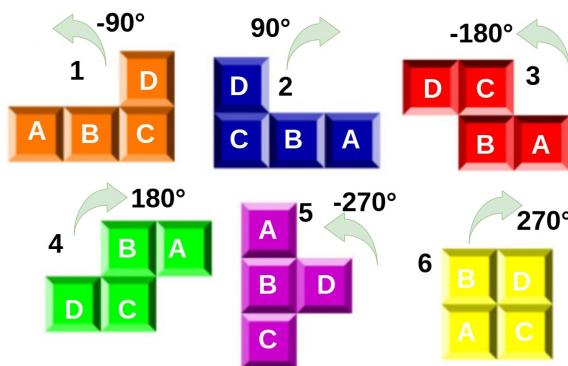
En aquella época los exámenes eran en papel y el cerebro del profesor funcionaba como compilador e intérprete. La web y HTML eran toda una novedad, no había computación en la nube, ni IDE en la nube (como nuestro RPC IDE <https://www.rpcide.cloud/>). En cambio, en esta época, los avances tecnológicos son maravillosos, y se requiere toda la creatividad y agilidad de los amantes de la programación competitiva.

¿Aceptas el reto de resolver en 2023 este examen de hace casi 20 años?

El Tetris consiste en varias figuras conformadas por recuadros. La siguiente imagen muestra seis de las muchas figuras que puede tener un Tetris, enumeradas de 1 a 6 y etiquetando los recuadros con las letras A, B, C y D.



El juego permite rotar las figuras a la izquierda o a la derecha 90°, 180° y 270°. La siguiente imagen muestra ejemplos de rotaciones de las seis figuras.



Imagina que por alguna invasión alienígena los computadores pierden la multimedia y solo se puede usar texto. En ese caso, las figuras del Tetris se podrían pensar como una pequeña matriz de 3x3 y las rotaciones se harían tomando como punto de rotación el centro de la matriz. Las dos imágenes anteriores se verían de la siguiente forma (imprimiendo guion “-” en lugar de espacio en blanco):

<b>1</b>	<b>2</b>	<b>3</b>	<b>1</b>	<b>2</b>	<b>3</b>
A - -	- - A	- - -	- - -	- - -	D C -
B - -	- - B	A B -	- - D	D - -	- B A
C D -	- D C	- C D	A B C	C B A	- - -
<b>4</b>	<b>5</b>	<b>6</b>	<b>4</b>	<b>5</b>	<b>6</b>
- - -	- - -	- - -	- B A	A - -	- - -
- C D	- D -	A B -	D C -	B D -	- B D
A B -	A B C	C D -	- - -	C - -	- A C

Siguiendo esta representación se quiere simular un juego de Tetris, donde se recibe el número de la figura 1, 2, 3, 4, 5 o 6 y un conjunto de rotaciones a la izquierda o a la derecha de 90°, 180° y 270°. La simulación debe imprimir la figura en su estado final, después de aplicar todas las rotaciones.

## Input

La entrada consiste en una única línea de  $n$  números (separados por un espacio en blanco) que describen el escenario a simular. El primer número  $1 \leq f \leq 6$  es el identificador de la figura. El segundo número  $0 \leq r \leq 100$  es la cantidad de rotaciones aplicadas a la figura en su estado original (según la imagen del enunciado). Después  $r$  números  $c_1, c_2, \dots, c_{r-1}, c_r$ , con las rotaciones aplicadas a la figura, cuyos valores pueden ser 90, 180 y 270 para rotaciones hacia la derecha y -90, -180 y -270 para rotaciones a la izquierda. Se garantiza que solo serán esos valores.

## Output

Imprime tres líneas, cada línea con tres caracteres “A”, “B”, “C”, “D” ó guion “-”, sin dejar espacios en blanco, representando la figura del Tetris, después de haber aplicado las rotaciones.



## Examples

Input	Output
1 6 90 180 90 -90 90 -90	--- --D ABC
2 5 270 90 -90 90 90 90	--- D-- CBA
3 7 270 90 -90 -90 90 90 -180	DC- -BA ---
4 6 90 180 90 -90 90 180	-BA DC- ---
5 7 270 90 -90 -90 90 90 -270	A-- BD- C--
6 5 270 90 -90 90 270	--- -BD -AC

## Note

Use Fast I/O

## Problem K. Kepler Barriers

Source file name: Kepler.c, Kepler.cpp, Kepler.java, Kepler.py

Input: standard input

Output: standard output

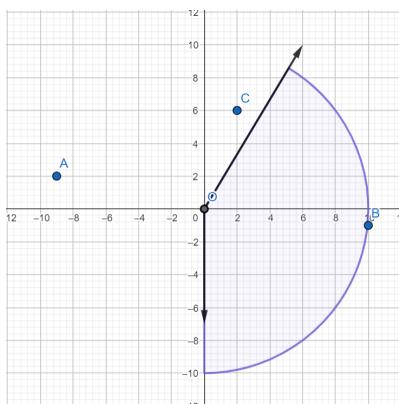
Author(s): Luis Daniel Carreño Pitalua - UFPS (Estudiante)

En el dominio de Teleportia, la tecnología ha alcanzado nuevos horizontes gracias a la invención de las Torres de Teletransportación. Estas estructuras, alimentadas por energía cuántica, han revolucionado por completo la forma en que las personas se desplazan y se conectan en todo el dominio.

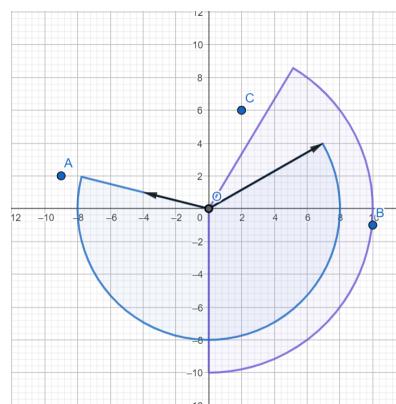
Para que las Torres de Teletransportación funcionen a la perfección, es necesario crear un Núcleo teleportador que sirva como punto de conexión central entre todas las torres. Basta con que una torre sea visible desde el Núcleo teleportador para que exista una conexión entre ambos.

Sin embargo, debido a las características tan particulares de este dominio, todas las noches se forman una serie de barreras cuánticas, llamadas Barreras de Kepler. Estas barreras se disponen formando un arco de circunferencia con respecto al Núcleo teleportador y se generan una después de la otra a lo largo de la noche (por algún motivo que se desconoce). Durante su existencia, las barreras son capaces de bloquear la visión la torre hacia el Núcleo teleportador, lo que inhabilita temporalmente su funcionamiento. Afortunadamente, las barreras desaparecen al amanecer y las torres vuelven a operar con normalidad.

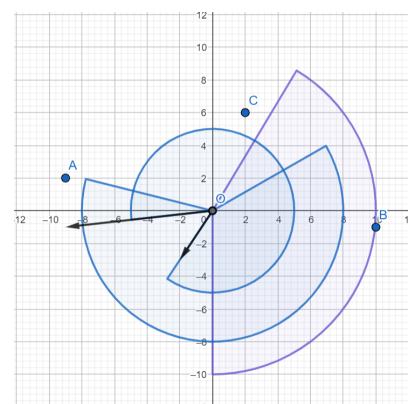
Teleportia se puede ver como un plano de dos dimensiones donde en la coordenada  $(0, 0)$  se encuentra el Núcleo teleportador. Cada una de las Torres de Teletransportación se encuentra ubicada en una coordenada  $(x, y)$  del plano. Por su parte, cada una de las barreras de Kepler está definida por dos vectores,  $\vec{v}$  y  $\vec{u}$ , los cuales determinan el ángulo que abarca la barrera. Este ángulo es igual al formado desde el vector  $\vec{v}$  hasta el vector  $\vec{u}$  en sentido antihorario. También se define un radio  $r$ , el cual indica a qué distancia se generará la barrera de Kepler con respecto al Núcleo teleportador.



(a) Aparición de la primera barrera



(b) Aparición de la segunda barrera



(c) Aparición de la tercera barrera

Visualización del primer caso de prueba.

Dado el orden en que aparecen las barreras de Kepler durante una noche y la posición de las Torres de Teletransportación existentes en el dominio, se requiere determinar cuántas torres quedan inhabilitadas después de la aparición de cada una de las barreras de Kepler. Esto permitirá tener un mayor conocimiento sobre el impacto que tienen estas barreras en el funcionamiento de las torres y así poder tomar medidas adecuadas para optimizar su rendimiento.

### Input

La primera línea contiene dos enteros  $N, M$  ( $1 \leq N, M \leq 10^5$ ), indicando el número de Torres de Teletransportación en el dominio y el número de barreras de Kepler que aparecerán en la noche, respec-



tivamente.

Cada una de las siguientes  $N$  líneas contiene dos enteros  $x, y$  ( $-10^9 \leq x, y \leq 10^9$ ); los cuales indican la posición en el dominio de cada una de las torres. Está garantizado que dos torres distintas no comparten la misma ubicación.

Las siguientes  $M$  líneas describen cada una de las barreras de Kepler, cada una con cinco enteros. Los primeros dos enteros  $x_1, y_1$  ( $-10^9 \leq x_1, y_1 \leq 10^9$ ) definen el vector desde el cual inicia la barrera. Los siguientes dos enteros  $x_2, y_2$  ( $-10^9 \leq x_2, y_2 \leq 10^9$ ) definen el vector donde acaba la barrera, y el último entero,  $r$  ( $0 < r \leq 10^9$ ), indicando la distancia a la que se generará la barrera con respecto al Núcleo teleportador. Las barreras son dadas en orden cronológico de aparición.

Está garantizado que ninguna torre o barrera coincide con el punto  $(0, 0)$ .

## Output

Imprime una única línea con  $M$  enteros donde el  $i$ -ésimo entero indica la cantidad de torres inhabilitadas luego de la aparición de la  $i$ -ésima barrera.

## Examples

Input	Output
3 3 -9 2 10 -1 2 6 0 -7 6 10 10 -4 1 7 4 8 -2 -3 -9 -1 5	1 2 3
5 3 6 -6 -3 10 -10 -9 -1 10 -5 2 7 10 -10 0 9 -7 -5 3 -5 9 5 8 1 0 7	2 3 4
2 2 3 3 -3 3 2 1 -1 2 3 1 2 -5 4 5	1 1

## Problem L. Longest Periodic Substring

Source file name: Longest.c, Longest.cpp, Longest.java, Longest.py

Input: standard input

Output: standard output

Author(s): Jose Manuel Salazar Meza - UFPS (Graduado)

Una cadena  $s$  de longitud  $n$  es llamada *periódica* si existe una cadena  $p$  de longitud  $k$  ( $k < n$ ) tal que  $s$  se puede formar concatenando dos o más repeticiones de  $p$ . En ese caso, se dice que  $s$  tiene un *periodo* igual a  $k$ . Por ejemplo, las cadenas “abababab” y “abcabc” son *periódicas* porque “abababab” puede ser formada por 4 repeticiones de la cadena “ab” o 2 repeticiones de la cadena “abab”, y “abcabc” puede ser formada por 2 repeticiones de la cadena “abc”, mientras que las cadenas “ababa” y “abcdefg” no son *periódicas*.

Una subcadena de  $s$  es una cadena  $x$  no vacía que consta de caracteres continuos de  $s$ . Por ejemplo, “lucho”, “luchoneta” y “choneta” son subcademas de “luchoneta”, mientras que “ufps” no lo es.

Dada una cadena  $s$ , tu tarea es encontrar la subcadena *periódica* más larga de  $s$  o informar que no hay ninguna. Si hay múltiples subcademas que cumplen la condición, indica cuantas **diferentes** hay y elige la lexicográficamente menor.

**Ver las notas para aclaraciones.**

### Input

La entrada consiste de una sola línea que contiene la cadena  $s$  ( $1 \leq |s| \leq 2 * 10^5$ ) – una secuencia de letras minúsculas en inglés.

### Output

Si no hay subcademas *periódicas* en  $s$ , entonces imprime 0.

De lo contrario, imprime el número de subcademas **diferentes** que cumplen la condición y, en la siguiente línea, imprime la subcadena *periódica* más larga de  $s$  lexicográficamente menor.

### Examples

Input	Output
aaaaa	1 aaaaa
aababcbcabab	2 abab
xmaratoninternaufps	0

### Note

Dos subcademas  $x$  y  $y$  de una cadena  $s$  son diferentes si  $|x| \neq |y|$ , o si existe un  $i$  ( $1 \leq i \leq |x|$ ), tal que  $x_i \neq y_i$  (y  $|x| = |y|$ ). Aquí  $|a|$  denota la longitud de la cadena  $a$ , y  $a_i$  denota el carácter en la posición  $i$  de la cadena  $a$ .

Una cadena  $x$  es lexicográficamente menor que una cadena  $y$ , si  $x$  es un prefijo de  $y$  (y  $x \neq y$ ), o si existe un  $i$  ( $1 \leq i \leq \min(|x|, |y|)$ ), tal que  $x_i < y_i$ , y para cualquier  $j$  ( $1 \leq j \leq i$ )  $x_j = y_j$ .

## Problem M. Matchsticks Puzzle

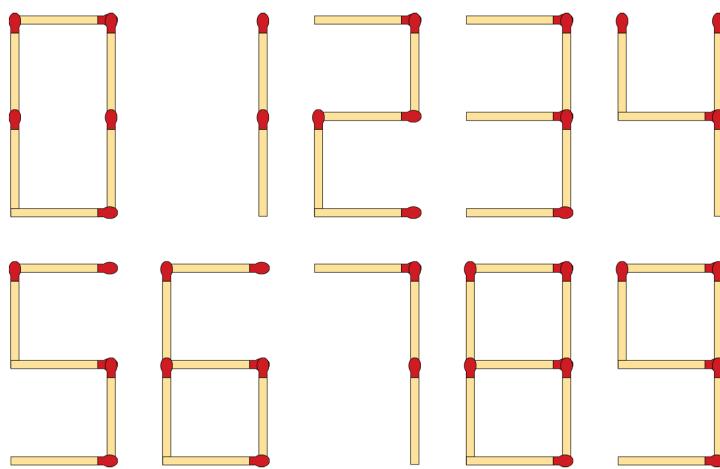
Source file name: Matchsticks.c, Matchsticks.cpp, Matchsticks.java, Matchsticks.py

Input: standard input

Output: standard output

Author(s): Jose Manuel Salazar Meza - UFPS (Graduado)

Desde que era un niño, Manuel se ha interesado por los rompecabezas, los acertijos, y cualquier juego que lo desafíe a resolver un problema pensando y siendo creativo. El otro día, se encontró con un rompecabezas de cerillas que decía: Dado el número 508 hecho con cerillas, ¿cuál es el mayor número que se puede obtener moviendo exactamente 2 cerillas? ¿Podrías resolverlo?



La imagen muestra cómo representar los diez dígitos decimales con cerillas.

Algunas respuestas correctas podrían ser 999, 5051 o 51181, dependiendo de las reglas del juego. Para Manuel, este problema no supuso ningún desafío y logró resolverlo correctamente en poco tiempo. Sin embargo, después de resolverlo, pensó en un problema más desafiante: Dado un número  $s$  con  $n$  dígitos hecho con cerillas, ¿cuál es el menor y el mayor número que se puede obtener moviendo exactamente  $k$  cerillas, para cada  $k$  desde 1 hasta  $M$ , donde  $M$  es igual a la cantidad total de cerillas en el número  $s$ ?

Para este problema, definió algunas reglas:

- Mover una cerilla consiste en levantarla y ponerla en otro lugar donde no había una cerilla inicialmente. Una cerilla solo se puede mover una única vez.
- Está permitido remover dígitos o agregar nuevos tanto a la izquierda como a la derecha del número  $s$ , siempre y cuando, al final, el número formado no tenga espacios vacíos entre dígitos. No está permitido agregar nuevos dígitos en medio de los  $n$  dígitos iniciales.
- Si deseas convertir un dígito existente en otro diferente, solo puedes convertirlo en **un solo** nuevo dígito. Por ejemplo, 4 puede ser convertido en 3 moviendo una cerilla y agregando otra, pero 0 no puede ser convertido en 11 levantando dos cerillas ni en 111 moviendo dos cerillas.
- Mover exactamente  $M$  cerillas es equivalente a reorganizar todas las cerillas como se deseé.
- En todos los casos, las  $M$  cerillas deben formar parte de algún dígito del número formado.

Este problema sí que es mucho más desafiante para Manuel, y ahora tanto él como tú están listos para enfrentarlo. ¡Buena suerte!

## Input

La primera línea contiene un entero  $n$  ( $1 \leq n \leq 12$ ) que indica la cantidad de dígitos del número  $s$ .

La segunda línea contiene el número  $s$ , que consta de  $n$  dígitos hechos con cerillas.

## Output

Imprime una línea por cada  $k$  desde 1 hasta  $M$  indicando el menor y el mayor número que se puede obtener moviendo exactamente  $k$  cerillas, separados por un espacio. Si no hay respuesta para un  $k$  en particular, imprime un “\*”.

## Examples

Input	Output
1 2	3 3 5 5 17 71 17 71 2 71
2 74	* 13 711 8 711 8 117 12 711 * 8 711
3 013	012 913 047 1017 031 9171 021 11311 08 77711 08 711111 08 711111 047 171111 08 711111 012 711111 021 711111 * 08 711111

## Problem N. Night of the Little Candles

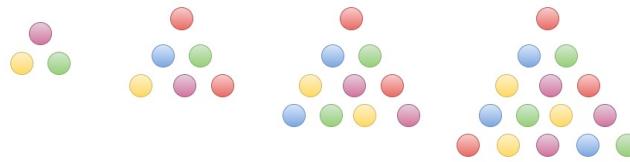
Source file name: Night.c, Night.cpp, Night.java, Night.py  
 Input: standard input  
 Output: standard output  
 Author(s): Milton Jesús Vera Contreras - UFPS (Professor)

Una de las tradiciones colombianas más importantes es **La Noche de las Velitas**, que se celebra el 7 de diciembre, en la víspera de la Inmaculada Concepción de la Virgen María, una fiesta de la religión católica. En cada familia se reúnen y encienden velitas de colores, agradeciendo o pidiendo deseos o favores a la Virgen, quien pasará esa noche por los hogares de aquellos que encienden las velas.



En la Universidad Francisco de Paula Santander (UFPS) Cúcuta, esa fecha suele coincidir con el inicio de exámenes finales, por lo que algunos estudiantes ponen velitas rogando salvar el semestre. Este año 2023, los estudiantes reunieron dinero y compraron muchísimas velitas,  $n \leq 10^{18}$ , y las van a poner en la UFPS.

Los estudiantes decidieron poner las velitas formando triángulos de lado  $L \geq 2$  velitas, porque tienen el agüero de que esa figura les ayudará a salvar el semestre. Comienzan con un triángulo de lado  $L = 2$  velitas, después otro de lado  $L = 3$  y así sucesivamente, creciendo siempre  $L$  en 1, como en la siguiente figura:



Cuando las velitas no son suficientes para armar un nuevo triángulo creciendo  $L$  en 1, comienzan nuevamente desde  $L = 2$  y repiten el procedimiento tantas veces como sea necesario, hasta que ya no haya más velitas o no sea posible armar un triángulo.

¿Podrías ayudar a los estudiantes a determinar cuántos triángulos de velitas se podrían armar con  $0 \leq n \leq 10^{18}$  velitas, y si sobran velitas, cuántas sobran?

### Input

La entrada consiste en un número  $0 \leq n \leq 10^{18}$ , el total de velitas compradas por los estudiantes.

### Output

Imprime dos números separados por espacio en blanco. El primer número  $t$  es el total de triángulos de velitas que armaron los estudiantes y el segundo número  $s$  es la cantidad de velitas que sobraron.

## Examples

Input	Output
10	2 1
100	10 2
1000	22 1
10000000000000000000	1836092 0
999999999999999999	1836091 2

## Problem O. OMG!!! A Rainy Road

Source file name: Rainy.c, Rainy.cpp, Rainy.java, Rainy.py  
Input: standard input  
Output: standard output  
Author(s): Jorge Andres Marles Florez - UFPS (Estudiante)



El mejor equipo de Fornique, UFPechugonaS, necesita ir a la Final Regional del ICPC, pero, debido a cortes de presupuesto, ellos solo tienen dinero para ir por carretera.

Fornique es un país con  $N$  ciudades, numeradas de 1 a  $N$ , conectadas por  $m$  vías de ida y vuelta entre ellas, cada vía conecta un par de ciudades  $u$  y  $v$ , y toma  $t$  tiempo cruzar esa vía. UFPechugonaS está en la ciudad 1, y la Final Regional del ICPC se hará en la ciudad  $N$ , ellos quisieran saber el tiempo mínimo para ir desde la ciudad 1 a la ciudad  $N$ .

Sin embargo, el gobierno de Fornique ha declarado una emergencia por las lluvias, y algunas vías podrían quedar bloqueadas por derrumbes, con el fin de no quedar atrapados en una vía derrumbada o incluso peor, UFPechugonaS viajará en el vehículo más inteligente del mundo, La Luchoneta, éste vehículo viene con una poderosa IA llamada MagicalGirl, que puede predecir en qué vía ocurrirá un derrumbe, cuándo ocurrirá y cuándo despejarán la vía.

Para evitar quedar enterrados en algún derrumbe, si el equipo quiere cruzar por una vía, y esa vía está bloqueada por un derrumbe en ese momento, deben esperar hasta que la vía sea despejada, además, no pueden entrar a una vía si no tienen suficiente tiempo para salir antes de que un derrumbe ocurra.

Desafortunadamente, todos los integrantes del equipo de UFPechugonaS están ocupados estudiando para la competencia, así que te pidieron ayuda. Dadas las ciudades, caminos, y los derrumbes que ocurrirán, ¿Cuál es el tiempo mínimo que le tomará a La Luchoneta completar el viaje?

### Input

La primera línea contiene 3 números enteros,  $N$  ( $1 \leq N \leq 10^5$ ),  $m$  ( $n - 1 \leq m \leq \min(10^5, \frac{n(n-1)}{2})$ ) y  $d$  ( $0 \leq d \leq 10^5$ ) – el número de ciudades, el número de vías y el número de derrumbes predichos, respectivamente. Es garantizado que hay una ruta entre las ciudades 1 y  $N$ .

Luego, siguen  $m$  líneas que contienen 3 enteros,  $u$ ,  $v$  ( $1 \leq u, v \leq n, u \neq v$ ) y  $t$  ( $1 \leq t \leq 10^8$ ) indicando que existe una vía de ida y vuelta entre las ciudades  $u$  y  $v$  que toma  $t$  unidades de tiempo cruzarla. Es garantizado que no existe más de una vía entre dos ciudades  $u$  y  $v$ .

Luego, siguen  $d$  líneas que contienen 4 enteros,  $u$ ,  $v$  ( $1 \leq u, v \leq n, u \neq v$ ),  $l$  y  $r$  ( $0 \leq l < r \leq 10^{10}$ ), indicando que la vía entre las ciudades  $u$  y  $v$  estará bloqueada desde el tiempo  $l$  hasta el tiempo  $r$ . Está garantizado que existe una vía entre  $u$  y  $v$ .

## Output

Imprime un número entero, el tiempo mínimo requerido para ir de la ciudad 1 a la ciudad  $N$ .

## Examples

Input	Output
10 14 5 1 2 1 1 4 1 2 3 2 3 4 1 4 5 3 4 6 15 4 8 2 5 6 1 5 7 6 6 7 7 6 9 1 7 9 9 8 9 1 9 10 3 2 3 1 2 5 6 0 8 4 8 0 8 6 7 0 10000 7 9 10 12	13
2 1 2 1 2 1 1 2 0 2 1 2 1 3	4

## Note

En el primer caso de prueba, el orden de las acciones será:

1. Ir de la ciudad 1 a la ciudad 4 (después de eso,  $t = 1$ )
2. Ir de la ciudad 4 a la ciudad 5 (después de eso,  $t = 4$ )
3. Esperar en la ciudad 5, 4 unidades de tiempo (después de eso,  $t = 8$ )
4. Ir de la ciudad 5 a la ciudad 6 (después de eso,  $t = 9$ )
5. Ir de la ciudad 6 a la ciudad 9 (después de eso,  $t = 10$ )
6. Ir de la ciudad 9 a la ciudad 10 (después de eso,  $t = 13$ )

En el segundo caso de prueba, la vía de 1 a 2 estará bloqueada desde el tiempo  $t = 0$  hasta  $t = 3$ . Por lo tanto, solo se podrá usar en tiempos  $t \geq 3$ . Luego de esperar, y tras otra unidad de tiempo cruzando la vía, el tiempo total será  $t = 4$  unidades de tiempo.