VIII Maratón de Programación

2015

2016

2017

2018

2019

2020

2021

2022

Try, Learn, Enjoy

03 Sábado 03 de Septiembre de 2022

From 12:00 to 19:00

Aulas Sur Sb404

C++ python Java

Universidad Francisco de Paula Santander
Vigilada Mineducación

pragma

SILUX

Programa de Ingeniería de Sistemas
Acreditado de Alta Calidad
"Educación y Tecnología con Compromiso Social"

RED DE PROGRAMACIÓN COMPETITIVA
RPC
while (true) {
keepTraining();
}

Contents

1 VIII Maratón de Programación UFPS - 2022

Desde el año 2015, 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 computaciona, como resolución de problemas, programación de computadores, trabajo colaborativo y liderazgo.

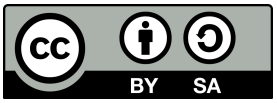
Los pioneros de dicho proyecto fueron dos estudiantes, quienes ya se graduaron y dejaron como herencia una gran motivación. Ahora siguen colaborando con el Semillero que es liderado directamente por los estudiantes, quienes ingresan desde primer semestre con el entusiasmo que trae el sueño de llegar 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 seis (6) años creciendo en Programación Competitiva junto a muchas universidades en varios países de toda Latinoamerica.

Para éste año 2022 nos complace presentar un conjunto de trece (13) problemas. Algunos son inéditos, escritos por estudiantes, profesores y graduados de varias universidades. Y otros corresponen al banco de problemas recopilados por RPC. Por la pandemia del COVID19, fueron dos años virtuales y este es el primer año nuevamente presenciales, algo muy importante, porque regresan los globo y las fotografías que registran la energía y alegría de los estudiantes.

Reconocimiento y agradecimiento especial al equipo de trabajo de todos los años:

- Hugo Humberto Morales Peña - Profesor Universidad Tecnológica de Pereira, Colombia
- Milton Jesús Vera Contreras - Profesor UFPS (desde Cúcuta)
- José Manuel Salazar Meza - Graduado UFPS (desde Cúcuta)
- Carlos Fernando Calderón Rivero - Estudiante UFPS (desde Cúcuta)
- Angie Melissa Delgado - Graduado UFPS (desde Cali)
- Gerson Yesid Lázaro - Graduado UFPS (desde Bogotá)
- Diana Espinosa y su semillero - Universidad de la Amazonía, Colombia
- Juan José Ortiz Plaza - Estudiante Universidad de la Amazonía, Colombia
- Equipo Red de Programación Competitiva (Diana Espinosa, Fabio Avellaneda, Johany Careño)
- Compañeros y Directivos de la UFPS Cúcuta



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 7 años consecutivos la clasificación a fase Regional Latinoamericana. En las dos últimas competencias de 2020 y 2021 la UFPS se ha logrado ubicar en el top 10.

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 trece (10) 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 unicarse en el top 3 de la competencia. Un quinto reto es lograr ubicarse en el top 5 o ser el mejor equipo novato ;)

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

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. Ninguna.

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 ocho años entre la UFPS y RPC, siempre con mucho entusiasmo y cariño para mantener viva la llama de la Programación Competitiva.

Problem A. Amazonía Univesity Tour

Source file name:	Amazonia.c, Amazonia.cpp, Amazonia.java, Amazonia.py
Input:	`standard input`
Output:	`standard output`
Author(s):	`Juan José Ortiz Plaza - Uni. Amazonía (Student)`

Amazonía Univesity has hired you to teach the top n sites new students should know about. To do this you must define a route in which each site is visited only once on the route and say what is the shortest possible distance.

Each site is numbered by a unique value and they are connected by m bidirectional roads. The tour must start at the university's principal door, identified with the number x. And the tour may end anywhere.

Input

The first line of input contains three numbers n, m and x. n is the number of sites, m is the number of roads and x is the identifier of the university's principal door. $2 \leq n \leq 16, 1 \leq m \leq n * (n-1)/2, 1 \leq x \leq 10^5$.

The next m lines contain the description of the roads, the $i-th$ line contains three numbers: $u_i, v_i, w_i (1 \leq u_i, v_i, w_i \leq 10^5, u_i \neq v_i)$ indicating that you can go from the site u_i to the site v_i and viceversa with a distance w_i.

Output

Show the minimum distance possible by visiting all sites once, starting from x. If it is impossible then to show -1.

Examples

Input	Output
5 6 1 1 4 50 4 3 2 3 50 1 50 2 100 50 4 1 4 2 1	152
5 6 2 1 4 50 4 3 2 3 50 1 50 2 100 50 4 1 4 2 1	153
5 6 50 1 4 50 4 3 2 3 50 1 50 2 100 50 4 1 4 2 1	-1

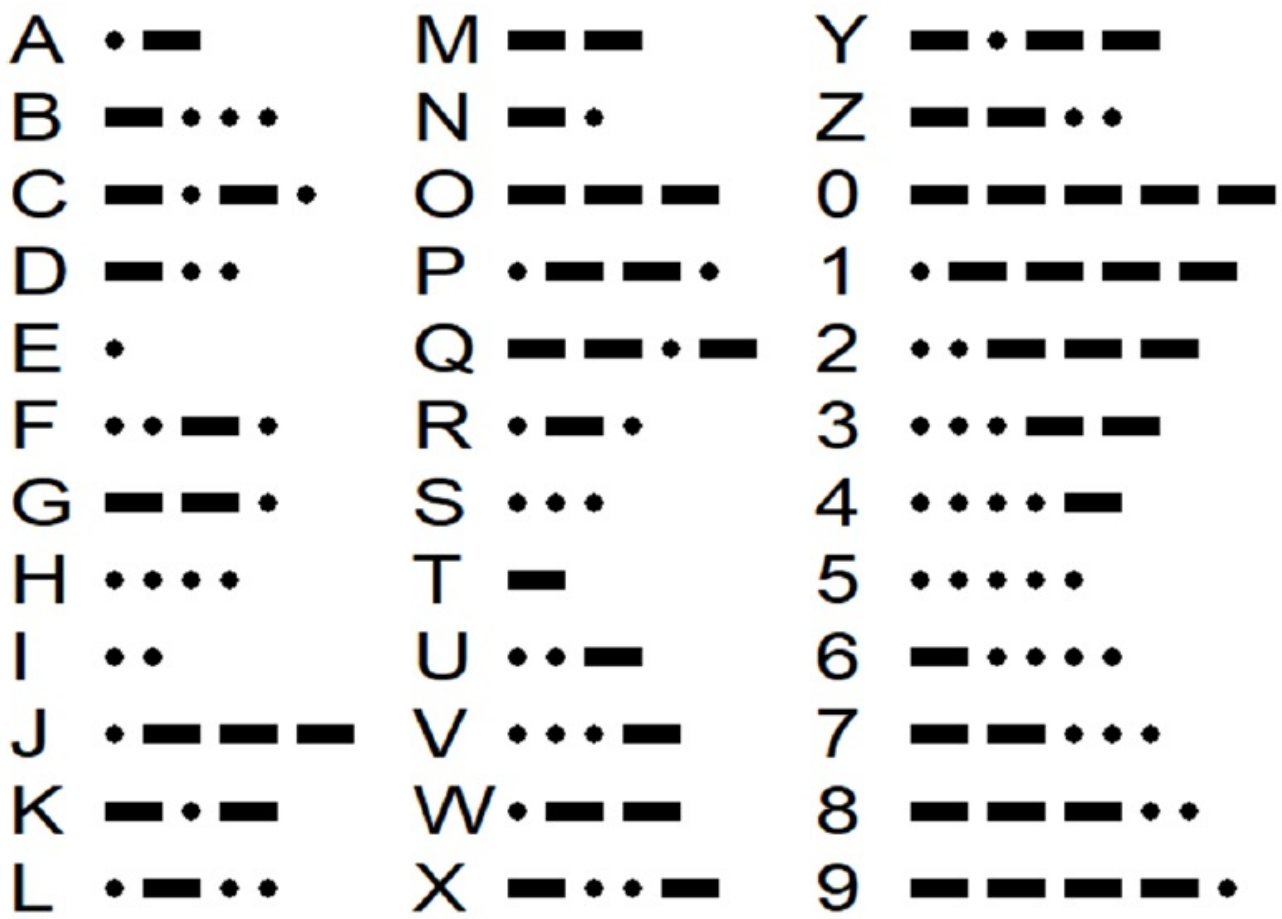
Problem B. [Building] Morse Code Palindromes

Source file name:	Building.c, Building.cpp, Building.java, Building.py
Input:	`standard input`
Output:	`standard output`
Author(s):	`Banco de Problemas RPC - Reto Externo Adaptado`

A **palindrome** is a word, number, phrase, or other sequence of characters which reads the same backward as forward, such as *madam* or *racecar* or *1881*. Phase palindromes ignore capitalization, punctuation, and word boundaries. For example:

`Madam I'm Adam.`

Morse code is a method used in telecommunication to encode text characters as standardized sequences of two different signal durations, called dots and dashes, or dits and dahs. Morse code is named after Samuel Morse, one of the inventors of the telegraph. The international morse code for letters and digits is (Note that the code for upper and lower case letters is the same.):



A word, number or phrase is a **Morse Code Palindrome** if the morse code for the letters and digits in the word, number or phrase reads the same backwards or forwards (ignoring spaces between character codes. For example:

159

Footstool

Write a program which takes as input a string and determines if the string is a *Morse Code Palindrome*.

Input

The single input line contains a string of up to 80 characters, possibly including spaces and other non-alphanumeric printable characters.

Output

The single output line consists of the string `YES` if the input string (ignoring everything but letters and digits) is a *Morse Code Palindrome*. Otherwise the output line consists of the string `NO`. If there are no letters or digits in the input string, the output should be `NO`.

Input	Output
hello	NO
159	YES
Madam I'm Adam	NO
footstool	YES
SOS	YES

Problem C. Counting songs

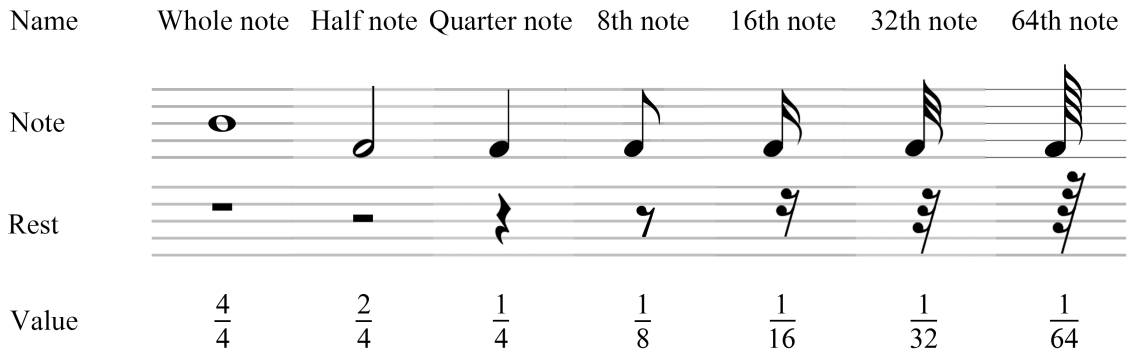
Source file name:	Counting.c, Counting.cpp, Counting.java, Counting.py
Input:	`standard input`
Output:	`standard output`
Author(s):	`José Manuel Salazar Meza (Graduate)`

"...No me cures, hermano, de delirio,
de aullido, desmesura o arrebato;
déjame arder en el amor ingrato
o en la inefable luz de otro martirio..."
(Silvio Rodriguez - Amigo Mayor)

Milton likes music, especially Silvio Rodriguez's music (Milton knows all his songs by heart). Silvio Rodriguez is a Cuban singer-songwriter, guitarist and poet who has composed more than 500 songs throughout his artistic career. Impressed by that, Milton wondered how many different songs Silvio Rodriguez will ever compose and your job is to help him find out.

First, Milton looked up the following definitions in his old music book:

- The music is subdivided into consecutive measures: measure 1, measure 2, measure 3, etc.
- A *measure* of $\frac{4}{4}$ has 4 beats available in it.
- Musical notes have a *value* that indicates the fraction of time they occupy within a measure. For example, in the image below we see that the value of a quarter note is $\frac{1}{4}$, so we can fill a measure of $\frac{4}{4}$ using 4 quarter notes, which total 4 beats.
- The *rest* symbols indicate that nothing should sound during the fraction of time indicated by their value.
- Adding a *dot* to a note or rest increases its value by 50%.
- Adding a *tie* between 2 or more consecutive notes (not rests) makes the notes join with a duration equal to the sum of their values.



After reading his old music book, Milton wants to know how many different songs of n measures of $\frac{4}{4}$ can be created? taking into account the following conditions:

- Any of the notes or rests in the image above can be used.

- Dots can be added to any note or rest except 64th notes.
- Ties can be added between notes, even from different measures.
- Only the rhythm (length of notes and rests) matters, not the sound of the note. That is, it does not matter if it is a C, D, E, F, etc.
- Two songs are different if there is an i such that the i-th measure of a song is different from the i-th measure of the other.
- Two measures are different if the rhythm of a measure is different from the other.

Input

The input consists of a single line with an integer n ($1 \leq n < 2^{31}$) indicating the number of measures.

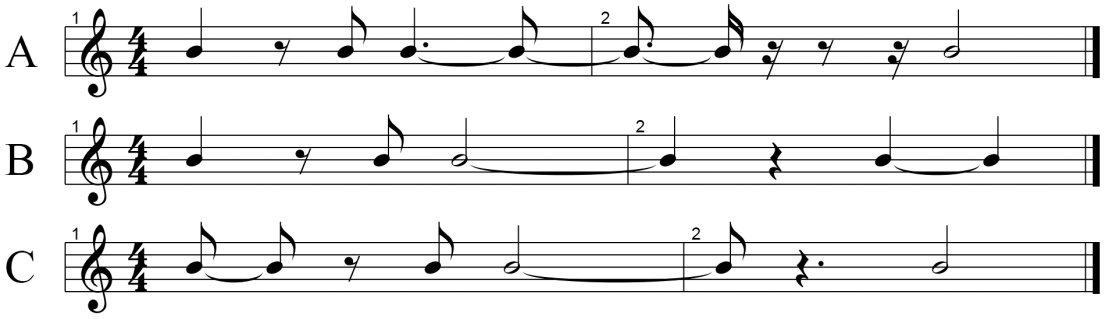
Output

The output consists of a single line with an integer indicating the answer to the problem modulo 1000000007 ($10^9 + 7$).

Examples

Input	Output
1	461493940
2	361271126
3	999820470
2022	879184852

Note



3 examples for $n = 2$. Songs A and B are the same, but songs A and C are different because the 2th measure has a different rhythm.

Problem D. Dividing and the double data type is a bad idea

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

Carlos is a student who use several division and double data type in his computer programs. His professor has warned that division results in precision errors, for example when the division is not exact and a repeating decimal number is generated. But Carlos insists on using division and double data type. Then his professor created this simple problem and challenged Carlos to solve it.

Input

The input consists of multiple test cases. Each test case consists of two integers a and n ($1 \leq a \leq 100, \quad 0 \leq n \leq 10^4$).

Output

For each test case two integer m and x must be printed if the division $\frac{1}{a^n}$ is exact and m is the smallest integer that satisfies this equality (in the set of integers):

$$x = \frac{1}{a^n} \cdot 10^m$$

If the division $\frac{1}{a^n}$ is inexact then print: `Precision Error`

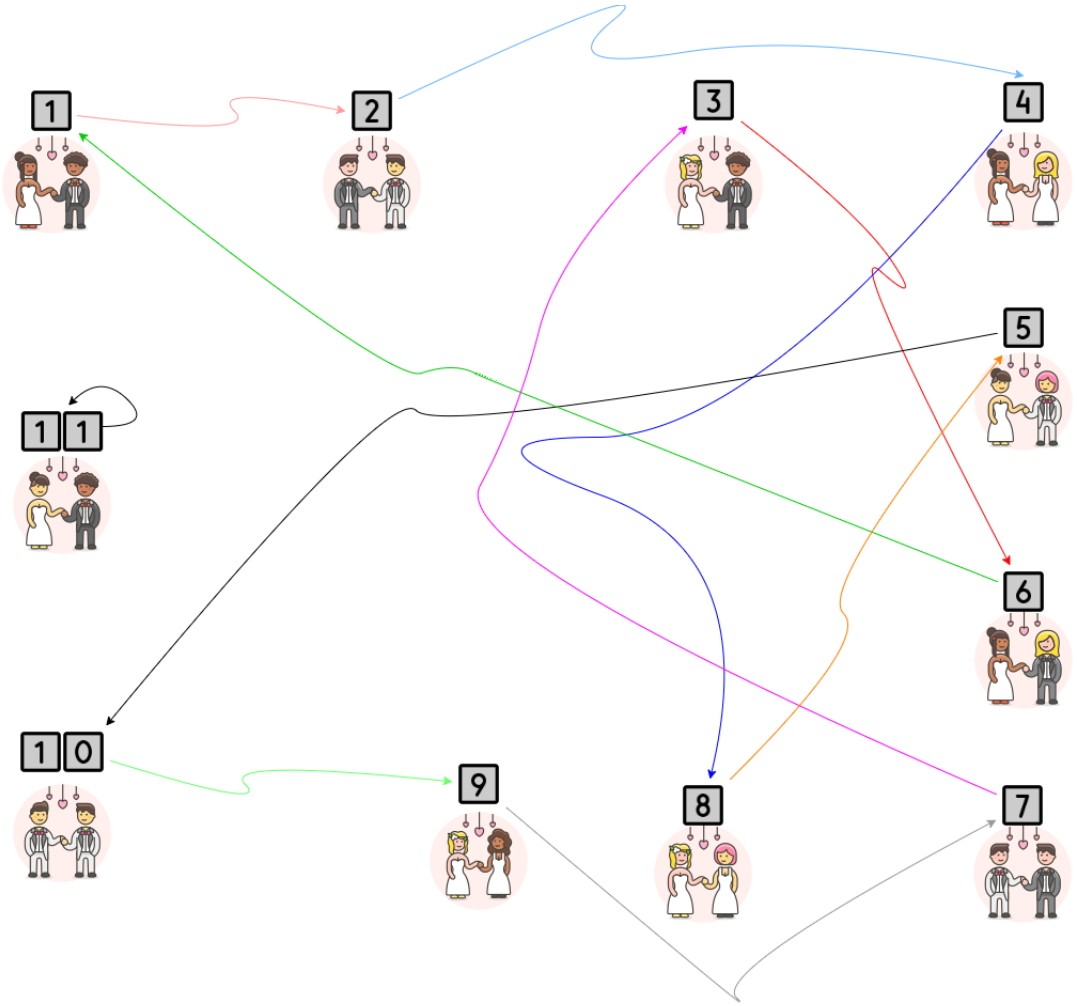
Input	Output
1 9999	0 1
2 6	6 15625
5 11	11 2048
10 15	15 1
80 15	60 28421709430404007434844970703125
7 19	Precision Error
99 23	Precision Error

A division is exact when the remainder is zero. The dividend equals the divisor multiplied by the quotient. The quotient is an integer number or a floating point number no periodic, for example: $\left(\frac{1}{16} = 0.0625\right)$, $\left(\frac{1}{125} = 0.008\right)$.

A division is inexact when there is a leftover remainder. The dividend equals the divisor multiplied by the quotient plus the remainder. The quotient is a floating point number periodic, for example: $\left(\frac{1}{7} = 0, \underbrace{142857}\underbrace{142857}\underbrace{142857}\ldots\right)$ with period (repeating decimal) 142857, $\left(\frac{1}{14} = 0, 0\underbrace{714285}\underbrace{714285}\underbrace{714285}\ldots\right)$ with period (repeating decimal) 714285.

Problem E. Expansión: Manuel's music group

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



Manuel is an engineer and musician and when he was a student he led the competitive programming teams. He is a member of a famous musical group named Expansión. In his concerts he organizes the public to dance following a simple algorithm:

First, Manuel count the number of dance couples n. Then divide the dance room into n zones and mark each zone with numbers from 1 to n, in a circular way, like a clock (the zone 2 is in front of the 1, the zone 3 is in front of the 2, zone 1 is in front of the n, etc). Then, he associates the zone i with the zone than is i positions after, as explained in the picture.

Once the zones and their associations are ready, each couple is located in a different zone and Expansion begins the party. Manuel randomly selects two zones x and y. The two dance couples that are in those selected zones must begin to move from zone to zone, following the previous associations. Finally, Manuel indicates the number of times a and b that the dance couples must move from zone to zone in each song that is played.

While playing his guitar, Manuel calculates how many songs must be played for the two dance couples

meet for the first time in the same zone and what area it will be. Manuel also think if the two dance couples will never meet. When the dance couples are in the same zone, they receive a prize and must leave. The concert ends when there are not dance couples.

Because Manuel is a lover of prime numbers, in his concerts the number n of dance couples is always a **prime number**. Also, because Manuel is a great musician, the number of dance couples is sometimes a very large number ($3 \leq n \leq 10^{10}$.

Carlos, Manuel's brother, has been thinking about his brother's algorithm for nine months. He know that if $x = n$ or $y = n$ the solution to your problem is trivial. But there are other cases where the problem is really hard. Will you be able to solve it before him?

Input

The input consists of a single line containing five integers n, x, a, y and b ($3 \leq n \leq 10^{10}$ and **is prime**, $1 \leq x, y < n$, $1 \leq a, b \leq 10^{10}$) that indicate the total of dance couples and the numbers chosen by Manuel, described above.

Output

The output consists of a single line. If the two dance couples never meet, print -1. Otherwise, print two integers k and z indicating how many songs must be played for the two dance couples to meet for the first time on the same zone and what that zone will be, respectively.

The answer is guaranteed to fit into a 64-bit integer.

Examples

Input	Output
5 1 2 2 3	3 4
7 4 2 3 1	-1
100000007 67108864 2 1048576 3	6 77887708

Note

A number n is **prime** if it's greater than 1 and has only two different positive divisors: 1 and n.

Note

This problem is an unauthorized adaptation of an old problem. I hope you like it and solve it.

Problem F. Forsyth–Edwards Notation

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

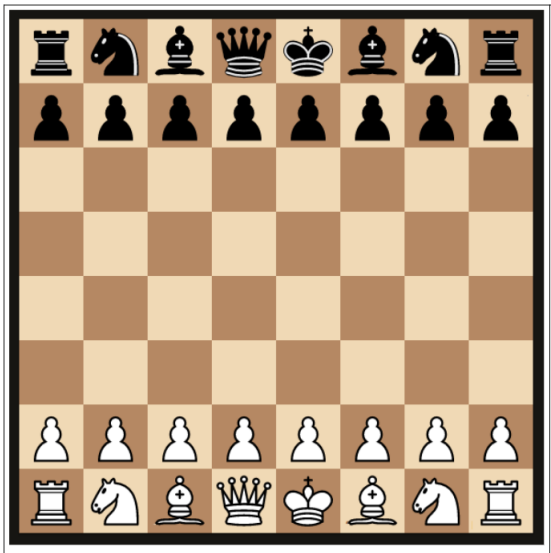
FEN (Forsyth–Edwards Notation) is a notation proposed in 1883 to save the state of a Chess Board. The rules of this notation are:

- White tiles are represented by Capital Letters
- Black tiles are represented by lowercase letters
- The letters used for tokens are:
  - Pawn P o p
  - Rook R o r
  - Knight N o n
  - Bishop B o b
  - Queen Q o q
  - King K o k
- If there arse blank spaces, consecutive spaces are counted and represented by a single digit from 1 to 8
- Each row is separated by a slash character "/"
- The board is stored from top to bottom and from left to right, located on the side with the white pieces.

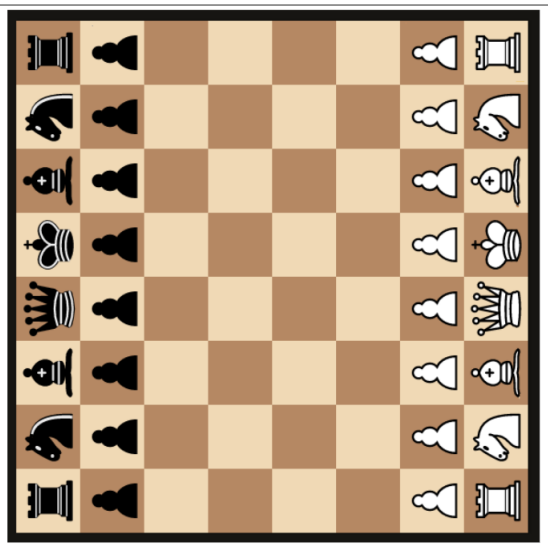
Professor Eduard really likes chess and he found this notation in a book, with a lot of pictures about chess endgames. The professor asked for help to student: to record the games using this notation. But, the student made a mistake: when recording the boards he turned the book 90° counterclockwise.

When the student discovered the mistake, he only had half a day left to turn in the assigned homework to the Professor. The student decided to solve the situation using a computer program. May you write a computer program to solve it?

The following figure show the chessboard in its initial position and its 90° counterclockwise rotation, both with FEN notation.



`rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR`



rp4PR/np4PN/bp4PB/kp4PK/qp4PQ/bp4PB/np4PN/rp4PR

Input

Several test cases. Each test case is a FEN notation *string*, but taken, by mistake, with the board rotated 90° counterclockwise.

Output

The correct chessboard, printing 0 where there are blanks and the letters corresponding to the pieces according to the FEN notation. A line break should be printed at the end of the board.

Examples

Input
rp4PR/np4PN/bp4PB/kp4PK/q2pP2Q/bp2P2B/np4PN/rp4PR
Output
r n b q k b n r p p p 0 p p p p 0 0 0 0 0 0 0 0 0 0 0 p 0 0 0 0 0 0 P P 0 0 0 0 0 0 0 0 0 0 0 0 P P 0 0 P P P P R N B Q K B N R

The purpose of this challenge is than first or second semester students to win the award for best beginner team. Do you want to win the prizes of the best beginner team? Solve this challenge, you can do it!

Problem G. [Game] ~~Football~~ Scoring

Source file name:	Game.c, Game.cpp, Game.java, Game.py
Input:	`standard input`
Output:	`standard output`
Author(s):	`Banco de Problemas RPC - Reto Externo Adaptado`

There are five ways to score points in american professional football:

1. Touchdown - 6 points
2. Field Goal - 3 points
3. Safety - 2 points
4. After touchdown
   (a) 1 point (Field Goal or Safety) - Typically called the "Point after"
   (b) 2 points (touchdown) - Typically called the "Two-point conversion"

(https://operations.nfl.com/the-rules/nfl-video-rulebook/scoring-plays/)

Given the box score values for two competing teams, calculate the point total for each team.

Input

There are two lines of input each containing five space-separated non-negative integers, T, F, S, P and C representing the number of **T**ouchdowns, **F**ield goals, **S**afeties, **P**oints-after-touchdown and two-point **C**onversions after touchdown respectively. $(0 \leq T \leq 10)$, $(0 \leq F \leq 10)$, $(0 \leq S \leq 10)$, $(0 \leq (P+C) \leq T)$. The first line represents the box score for the visiting team, and the second line represents the box score for the home team.

Output

The single output line consists of two space-separated integers showing the visiting score and the home score respectively.

Input	Output
1 3 0 0 1 3 1 1 1 1	17 26

Image source Wikimedia

Problem H. Honeycomb Havoc

Source file name:	Honey.c, Honey.cpp, Honey.java, Honey.py
Input:	`standard input`
Output:	`standard output`
Author(s):	`Carlos Fernando Calderón Rivero - UFPS (Student)`

Honeycomb Havoc is a Minigame found in Mario Party 2. In this game, players queue up and take turns below a tree with various types of fruits and honeycombs. On their turn, each player selects one of the two options: Take the next 1 or 2 items from the Tree. If the player gets only fruits, then keeps playing and go back to the end of the queue. Otherwise, if the player gets a honeycomb, it will be eliminated and wait outside up the end of the game. The player who remains in the queue when all others are eliminated wins the game.

Some rules:

- If the player selects two items, but the first one is a honeycomb, then it will be eliminated immediately and won't get the extra item.
- Before the minigame starts, a random number of fruits and honeycombs are generated and arranged in a line. The players will get the items in the order they were generated. If, for some reason, there isn't a winner after the items over, only honeycombs will be delivered from the tree to finish the game.
- The initial queue of players is ordered by the number of each player. Player 1 is first, player 2 is second and so on.
- A player controlled by the CPU will always select a number of items each turn depending on the player's number:
  - Player 2 will always take 1 item
  - Player 3 will always take 2 items
  - Player 4 will alternate. In its first turn it will take 1 item, then 2 items, then 1 again and so on.

Long story short, Bob is about to win the whole game, but he needs to win this last minigame and he will make anything for achieve it. He is playing against other three players controlled by the CPU. All the players can see the arranged line of items, so Bob would like to know if he can win the game choosing efficiently the numbers of items he must select in each of his turns.

Input

The first line of input contains a number T ($T <= 100$) with the number of test cases. Each case starts with a line containing two integers: N ($1 \leq N \leq 1000$), the number of items from the tree. The second line will contain a string of size N containing only letters "F" and "H" where F is a fruit and H is honeycomb. The order of each letter is the order of the items. Item i will be the $i-th$ delivered by the tree.

Output

Print a single line with word $POSSIBLE$ if Bob can win the game. Otherwise print $IMPOSSIBLE$

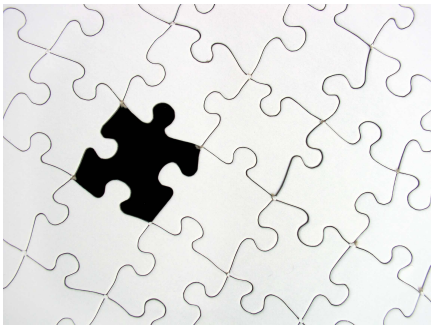
Examples

Input	Output
6 5 FHHFH 6 FFFFFF 5 FFFFF 7 FFFFFFF 8 FFHHFFHF 5 HFFFF	IMPOSSIBLE POSSIBLE IMPOSSIBLE POSSIBLE POSSIBLE IMPOSSIBLE

Image source https://www.mariowiki.com/Honeycomb_Havoc

Problem I. Impossible

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



Two mathematicians and one engineer are auditing bank ATM transactions. Each transaction is identified with a unique ID number that is consecutive. They have the unordered transactions list, but one ID number is missing. Mathematicians say that it is not possible find the missing ID number. The engineer says that there are cases that can be solved. May you help them?

Input

Several test cases. Each test case starts with a number $10^3 \leq n \leq 10^6$, which corresponds to the number of transactions. Then follows a list of $n-1$ ID numbers x_i, $10^5 \leq x_i \leq 10^9$. The ID numbers are consecutive and unordered, there are not duplicate numbers and one number is missing.

Output

For each test case print the ID number of transaction x_i that is missing or $IMPOSSIBLE$ if the mathematicians are correct and "it is not possible find the missing ID number".

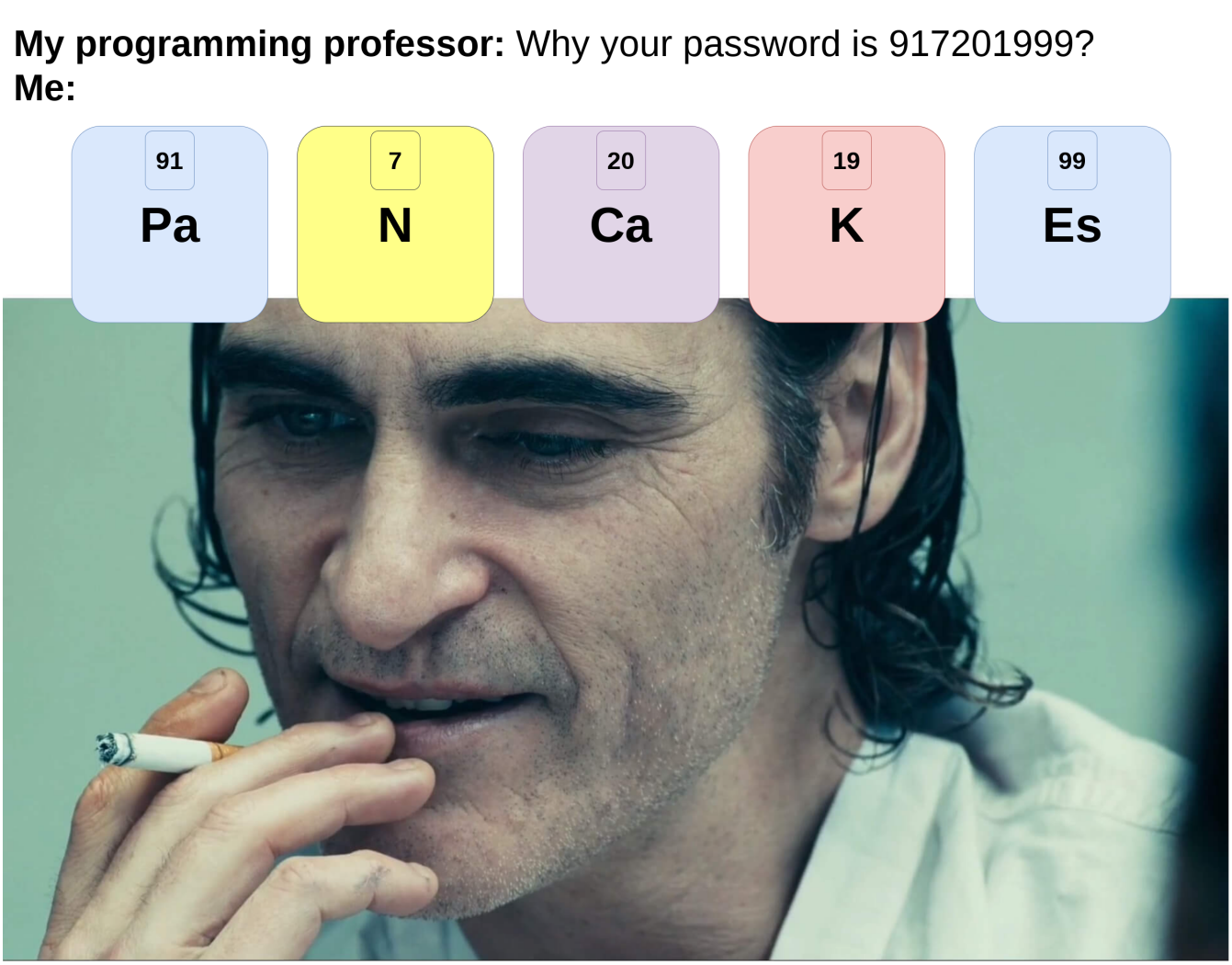
Examples

Input	Output
10 123465 123460 123459 123457 123458 123463 123461 123464 123456	123462
11 789065 789060 789059 789057 789058 789062 789063 789061 789064 789056	IMPOSSIBLE

Note: Due to the amount of data, an example is used with a number n smaller.

Problem J. Julian's Chemical Cryptography

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



Julián Vera is a clever student from Chinácota - Norte de Santander. He got "Generación E" scholarhip and studies chemistry at the Universidad de La Sabana. He is my cousin and sometimes he asks me for help with programming. Because we use the internet to communicate, the internet gossip bubble suggested me this meme for chemists, which inspired me to write this problem:

An interesting way to generate strong passwords is to take the atomic weight of various elements from the Periodic Table and concatenate them. If the symbol of such elements is concatenated, the resulting string sometimes corresponds to valid words, as in the meme: PaNCaKes.

The strength of the password lies in the fact that a number formed by the concatenation of atomic weights can correspond to various combinations of the Periodic Table symbols. In the meme example there are fifteen (15) combinations.

According to what Julián has learned about chemistry and programming, the password must be interpreted from left to right, in groups of 1, 2 or 3 digits, considering only those cases that correspond to the atomic

weight of a chemical element (from 1 to 118). Interpreting the meme example:

- The initial string is 917201999.
- There are two groups: F (9) or Pa (91). 917 is not possible.
- If F (9) is taken, the number 17201999 remains and the process is repeated.
- Then there are two groups: H(1) or Cl (17). 172 is not possible.
- If H (1) is taken, the number 7201999 is left and the process is repeated.
- Then there are two groups: N(7) or Hf (72). 720 is not possible.
- If N (7) is taken, the number 201999 is left and the process is repeated.
- Then there are two groups: He (2) or Ca (20). 201 is not possible.
- If He (2) is taken, the number 01999 is left and the process is repeated.
- Then there are not groups, because 0 is not valid atomic weight. Is necessary return to previous step and select the second group, Ca (20).
- If Ca (20) is taken, the number 1999 is left and the process is repeated.
- Then there are two groups: H (1) or K (19). 199 is not possible.
- If H (1) is taken, the number 999 is left and the process is repeated until the entire string is interpreted. In this case, by repeating the previous process, the group F (9) is obtained three times FFF. Then the concatenation of the symbols is $FHNCaHFFF$.
- Also, if H (1) is taken, you can get group F (9) and then again group Es (99). Then the concatenation of the symbols is $FHNCaHFEs$.
- Similarly, if H (1) is taken, you can get group Es (99) and then again group F (9). Then the concatenation of the symbols is $FHNCaHEsF$.
- In this example there are ten (10) combinations that start with F (9) and $five$ (5) that start with Pa (91). In total there are $fifteen$ (15) and one of them is $PaNCaKEs$;) and it is the last one that appears according to Julián's algorithm.
- There are cases like 10101000 and 9990009 that do not allow to generate any combination of symbols of the Periodic Table ;(

Input

The input consists of several test cases. Each case is a string S of minimum ten (10) and maximum twenty (20) characters, $10 \leq |S| \leq 20$. All characters S_i are digits from zero (0) to nine (9), $0 \leq S_i \leq 9$.

Output

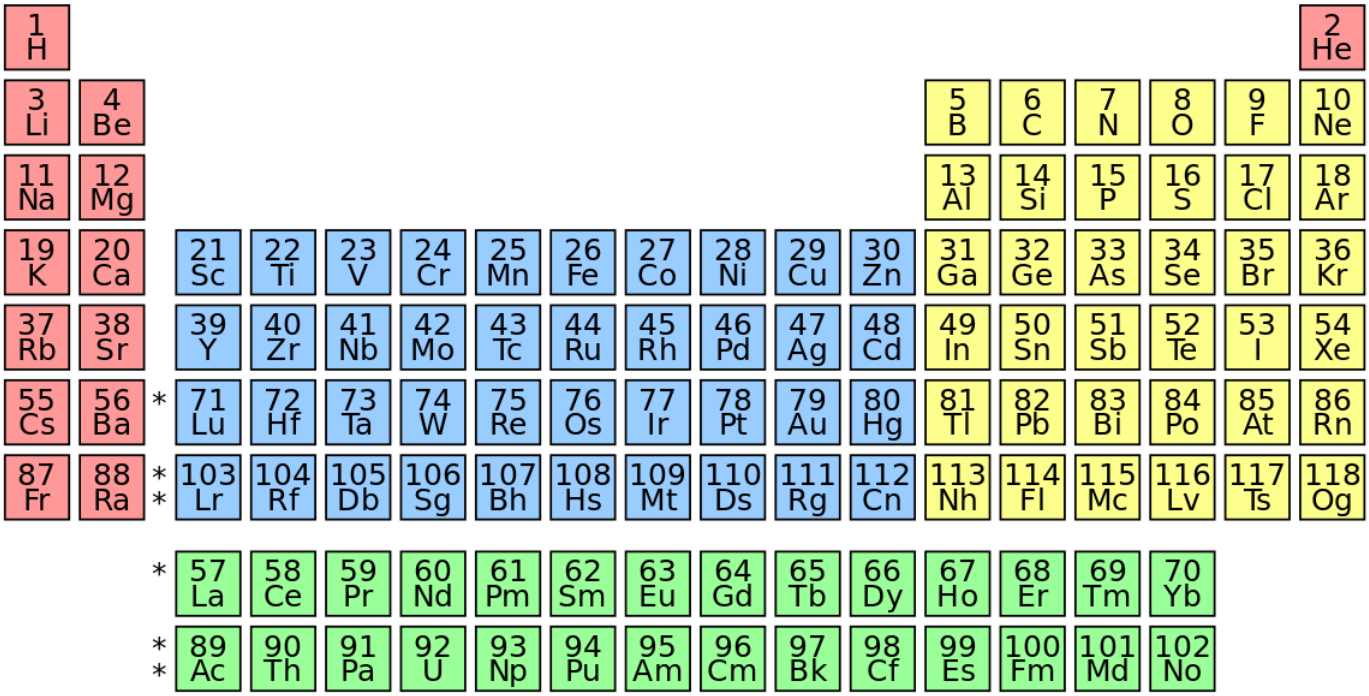
The output consists of all possible combinations of strings formed by concatenating the chemical element symbols, as described in the statement. Each combination must be printed on a separate line. If there is no combination, print the $ symbol.

Examples

Input	Output
917201999	FHNCaHFFF FHNCaHFEs FHNCaHEsF FHNCaKFF FHNCaKEs FClCaHFFF FClCaHFEs FClCaHEsF FClCaKFF FClCaKEs PaNCaHFFF PaNCaHFEs PaNCaHEsF PaNCaKFF PaNCaKEs
10101000	$

Use fast I/O

If you forgot the periodic table, here it is:



Note

I made the meme using draw.io, because the official meme has low resolution ;)

Problem K. Do you know who wrote this problem?

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

The leader students of the UFPS competitive programming study group won the ICPC Regional Contest and will go to the ICPC World Finals in Europe. The graduates of that group now have a lot of money, so they donated money for the students to travel. Then the leader students will be able to visit n different tourist places numbered from 1 to n.

Although the graduates have a lot of money, their donation only covers food, hotel and admission ticket, but does not include the cost of transportation. The students found on a website some cheap transport alternatives:

- Basic alternatives: that allow you to travel from a tourist place u to a tourist place v for a cost of w, but they are dangerous.
- Alternatives with fixed origin and variable destination: they allow traveling from a fixed origin tourist place u to any destination in the range $[l, r]$.
- Alternatives with fixed destination and variable origin: they allow traveling from any tourist place in the range $[l, r]$ to a fixed destination u.

However, the limitation of the the cheap transport alternatives is that they can be used only once.

The three students are excellent programmers and solve various types of problems. So they decided to write a computer program to estimate the minimum amount of money they will have to spend and make a budget that would allow them to visit as many tourist places as possible in Europe.

Do you want to qualify for the ICPC World Finals? If you solve this problem, you will be in the ICPC World Finals.

Input

The first line of the input contains three 3 integers n, p and g ($1 \leq n, p \leq 10^5; 1 \leq g \leq n$):

- n: Number of tourist places.
- p: Number of cheap transport alternatives.
- g: Initial position of the students.

The following p lines contain the description of the cheap transport alternatives. Each line begins with a string s, which defines the type of alternative ($basic$, $fixed{-}origin$, $fixed{-}destination$). If the alternative is of type $basic$, then three 3 integers v, u and w will follow where w is the cost ($1 \leq v, u \leq n, 1 \leq w \leq 10^9$). Otherwise four integer 4 will follow u, l, r and w where w is the cost ($1 \leq u \leq n, 1 \leq l \leq r \leq n, 1w10^9$).

Output

Print a single line with n integers separated by a space. The ith number is the cost to go from the students initial position to the ith destination, or -1 if it is impossible to get there.

Examples

Input	Output
6 8 1 fixed-origin 4 3 6 95 fixed-destination 3 1 3 24 fixed-destination 6 4 4 12 fixed-origin 5 1 2 9 basic 3 3 20 fixed-destination 4 2 6 32 basic 1 6 25 fixed-origin 6 1 5 6	0 31 24 31 31 25
5 4 1 fixed-origin 2 3 4 10 basic 2 4 16 fixed-destination 4 1 3 12 basic 2 5 25	0 -1 -1 12 -1

Note

This problem is an unauthorized adaptation of an old problem. I hope you like it and solve it.

Image source Wikipedia