Primera Fecha Gran Premio México 2018

Apr 28th 2018

Problems book

General information

This book contains a total of 10 problems; Pages are numbered from 1 to 12 not including this page. Verify your book is complete.

Problem A. Permuting and adding up

Source file name:	`A.c, A.cpp, A.java`
Input:	`Standard`
Output:	`Standard`
Author(s):	`Úscar Dávalos`

Given a positive integer number N, your task is to find how many different numbers exist that have the same digits as N and the sum of all these numbers.

As an example, if $N = 120$ then there are 6 numbers : 012, 021, 102, 120, 201 and 210. The sum of these numbers is $12 + 21 + 102 + 120 + 201 + 210 = 666$.

Input

The first line of input contains a single number T the number of test cases. The next T lines contain a single positive integer number N.

- $1 \leq T \leq 10^4$
- $0 \leq N \leq 10^{11}$

Output

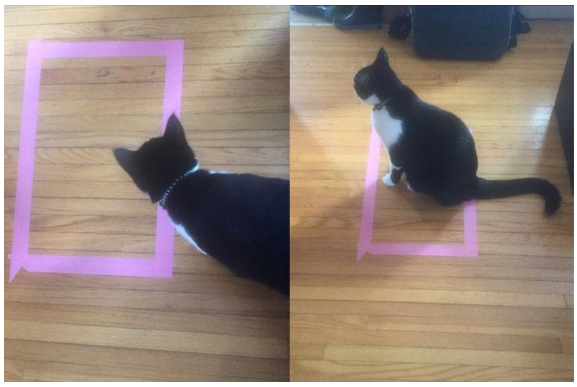
For each test case your program should print a single line with two integers separated by space. The first number represents the number of different numbers that exist with the same digits as N, the second number contains the sum of all these numbers.

Example

Input	Output
2 120 112	6 666 3 444

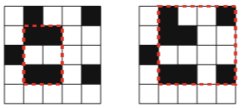
Problem B. Sleeping Baker

Source file name:	`B.c, B.cpp, B.java`
Input:	`Standard`
Output:	`Standard`
Author(s):	`Felipe Baquero, Lina Rosales`



Most cats love to sleep in well defined spots and Baker is not the exception, he loves to sleep in rectangular shape places. His owner decided to create for him a rectangular shape in her room. To do this, she will tie a piece of rope to different objects she has in the room to create rectangles with the rope.

In the following figure, the black sections are places in the room where the owner has an object where she can tie the piece of rope to create the rectangles. The red lines show possible rectangles she can create.



Help Baker's owner to discover how many different rectangles she can make for her beloved cat.

Input

There are multiple test cases. Each test case starts with a line with two integers separated by a space n and m describing the size of the room. The next n lines contain m characters each with the value 1 or 0, where 1 means that there is an object in that position were the rope can be tied of.

- $1 \leq n, m \leq 2500$

Output

For each test case print "case y: d" (without the quotes) where y is the number of test case starting with 1 and d is the number of different rectangles Baker's owner can make for him to sleep.

Example

Input	Output
5 5 01001 01100 10000 01101 00000 2 2 11 11 5 5 11111 11111 11111 11111 11111	case 1: 2 case 2: 1 case 3: 100

Problem C. Counting weak RNA sequences

Source file name:	`C.c, C.cpp, C.java`
Input:	`Standard`
Output:	`Standard`
Author(s):	`Araceli Velázquez`

An interesting problem in the production of proteins using bioengineering is to be able to determine when a protein can be efficiently created by a micro organism and when it is not. Some hypothesis lead that a protein will have low production levels when the transfer RNA sequence section has several bases of adenine (A) and thymine (T) chained together. This due to the fact that adenine is chained to thymine by 2 hydrogen links, while cytosine (C) is chained to guanine (G) by 3 links.

A section of the RNA sequence is any substring from the RNA sequence, this is, is formed of consecutive RNA bases from the RNA sequence. For example in the RNA sequence 'ATCTC', 'ATC' is a section of the RNA sequence while 'ACC' is not as these chars are not consecutive in the RNA sequence.

A complementary section of a RNA sequence exists only on those sequences that have only two types of bases and is defined as the sequence that contains the contrary base in the same position, examples of complementary RNA sequences are: 'TAAAA' and 'ATTTT', 'TATATA' and 'ATATAT'.

Normal ending transcription sections usually contain only bases of adenine and thymine, however it is not that usual that the RNA sequence contain also the complementary sequence of an ending transcription section, this is why we consider a weak RNA sequence as a section of 10 elements from the RNA sequence that contains only adenine and thymine and that its complementary section exists in the given RNA sequence.

Given a RNA sequence your task is to find how many weak RNA sequences exists in it.

Input

The first line of input contains a single number T the number of test cases to follow. The next T lines contain each a RNA sequence, the RNA sequence contains only the characters 'A' for adenine, 'C' for cytosine, 'T' for thymine, and 'G' for guanine and it's length is no longer than 10^6.

- $1 \leq T \leq 10$

Output

For each test case print in one line the number of weak RNA sequences that exist in the given RNA sequence.

Example

Input	Output
2 ATATATATATA GGCAAAGATATCGATCG	2 0

Explanation

In the first test case there are two weak sequences: 'atatatatat' and 'tatatatata'. For the second test case there are no weak sequences.

Problem D. Dividing Hexadecimal Numbers

Source file name:	`D.c, D.cpp, D.java`
Input:	`Standard`
Output:	`Standard`
Author(s):	`Gilberto Vargas Hernández`

Mr. Homft promised not to let homework to the kids that could accomplish a task in class. Today's lecture was about hex numbers. Hex numbers were invented by the lazy computer scientists from the last century who didn't want to write a lot of zeros and ones, so they synthesized binary into hex numbers. For a binary number it's possible to form groups of 4 bits and replace them by the hex digit. An easy way to convert from hex to binary! A hex digit is represented as a number from 0 to 9 or an uppercase letter from A to F which represents numbers from 10 to 15.

Mr. Homft's class is integrated only by smart kids, so don't get scared by their abilities. The last month they were studying Newton's laws, number theory and some other kind of sorcery tricks. Well, now that everything has been explained let's go to Mr. Homft's problem. He wrote a hex number N , actually a really huge one, and then he asked if the number was a multiple of 17. Passed no more than 5 minutes, all the kids had answered to the question correctly. You may be wondering if you must have taken the sorcery course last summer to calculate the result but maybe what you need is a different course. Can you answer correctly as the kids? Given a set of hex numbers your task is to say whether they are or not multiples of 17.

Input

You'll have to read until end of file. Each line of input will have the hex number N .

- N will have at most 10^5 hex digits.

Output

For each line of input, write "yes" if the hex number is divisible by 17, write "no" otherwise

Example

Input	Output
9999	yes
11	yes
AA	yes
0	yes
1	no

Problem E. Exact sum of squares

Source file name:	`E.c, E.cpp, E.java`
Input:	`Standard`
Output:	`Standard`
Author(s):	`Juan Pablo Marín`

Given the prime factors of a number N, can you find if there are two numbers a, b such that $a^2 + b^2 = N$?

Input

The first line of input contains a number K the number of prime numbers in the primer factorization of N. Each of the next K lines contain two numbers (P_i, B_i) separated by a space, the first number (P_i) is a prime number in the prime factorization of N the second number is the number of times that P_i appears in the prime factorization of N.

- $1 \leq K \leq 100$
- $1 \leq P_i \leq 10^6$
- $1 \leq B_i \leq 100$
- You can assume all values for P_i are prime numbers and none of the primer numbers repeat in the input.

Output

For each test case you must print a line with the string "Yes." if there are two numbers a, b such that $a^2 + b^2 = N$, print "No." otherwise.

Example

Input	Output
4 2 1 3 4 5 1 7 2	Yes.

Input	Output
2 3 3 11 4	No.

Explanation

In the first test case the given number $2^1 * 3^4 * 5^1 * 7^2 = 39690 = 189^2 + 63^2$. For the second test case $3^3 * 11^4 = 395307$ can not be represented as the sum of two squares.

Problem F. Finding the train

Source file name:	`F.c, F.cpp, F.java`
Input:	`Standard`
Output:	`Standard`
Author(s):	`Araceli Velázquez`

In a not so far future, trains are created totally using magnets. There are magnetic blocks that can be attached side by side regardless it's polarity, however they expose a polarity in the upper side of the block and on the lower side of the block, in such way that it allows an easier storage as if they are well designed you could break the train in two parts and stack one over the other.

As you can imagine by now a block can be stacked above another block if their polarity is not the same, but as these boxes are too heavy once the boxes are stacked in this way they can not be detached, however if a box is put above another box with the same polarity then the box will levitate over the other without moving.

Years, maybe centuries of work and effort have been put to create a machine that creates trains that can minimize the storage needed for them, a train is optimal for storage if it can be broken at the middle point so you have two equally sized parts that can be stacked one over the other in such way that the part you put above levitates, so that the length required to store the train is half the size of the train. As the half of the train that will be put above may be too heavy, the machine flips the two halves in such way that the last block of the first half is below the first block of the second half and the first block of the first half is below the last block of the second half.

Today the machine is not working properly and most of the trains turn out to not be optimal. You could easily disassemble all blocks from a train and return them to the machine to create a new train, but you risk to have another train that is not optimal for storage. So you take some initiative and decided to find from each train that the machine is creating what is the largest optimal storage train that you can get after removing an amount of blocks (maybe 0) from each of the sides of the train.

Input

The first line of input contains a number T, the number of trains to test. The following T lines contain a string S representing the specification of a train, the string contains only the symbols '+' and '-' and represents the polarity of each of the blocks that create the train.

- $1 \leq T \leq 10$
- $1 \leq |S| \leq 10^6$

Output

For each train in the input your program should print a line with a single integer, the length of the largest optimal storage train that can be found on S removing blocks only from the sides of the train.

Example

Input	Output
3 ++-++++-++- +-++-+ -++	10 6 2

Problem G. Gambusines

Source file name:	`G.c, G.cpp, G.java`
Input:	`Standard`
Output:	`Standard`
Author(s):	`Sergio Ivvan Valdez`

A gambusine explores a zone of R x C acres to determine if they can be exploded or not. The explosions are made on deposits that are a set of acres that are exploitable and contiguous, each acre will generate a benefit of g units when exploded, and the explosion of a deposit will take a cost of Txg regardless of it's size. A deposit is then profitable if it contains at least T exploitable and contiguous acres as the total benefit of exploding the sub area will be $(n_y - T)g$ where n_y is the number of acres in the exploitable sub area.

Given the rectangular map of R x C acres of the zone the gambusine will explore. You need to find the number of deposits that are profitable and the total benefit of exploding such deposits. The exploration zone is defined by a matriz M, such that $0 \leq M_{i,j} \leq 100$. If $M_{i,j} = 1$ then the acre (i, j) is exploitable. A deposit is a set of acres $M_{i,j}$ such that $M_{i,j} = 1$ and at least one of the acres in it's neighborhood $V_{i,j} = M_{i-1,j}, M_{i,j-1}, M_{i+1,j}, M_{i-1,j}$ is also exploitable, for any (i, j) in the deposit.

Input

The input file contains in the first line two values T and g. The next line will contain R and C the number of rows and number of columns respectively for the zone that the gambusine will explore. The next R rows contains C numbers separated by space representing the values for the zone as explained above.

- $1 \leq R, C \leq 100$
- $2 \leq T \leq R * C$
- $1 \leq g \leq 100$
- $0 \leq M_{i,j} \leq 100$

Output

The output file contains a single line with two numbers separated by a space, the first one is the number of profitable deposits in the zone to be explored, the second one is the total benefit after exploding such deposits.

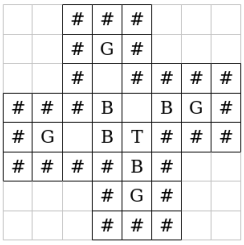
Example

Input	Output
4 2 10 7 82 83 84 85 86 87 88 83 84 85 1 1 88 1 1 85 86 87 88 1 1 85 1 87 88 1 90 91 86 1 1 89 1 91 92 87 1 89 1 1 92 93 1 1 90 91 92 93 94 89 90 1 92 1 94 95 90 91 92 93 94 95 96 91 1 93 1 95 1 1	2 4

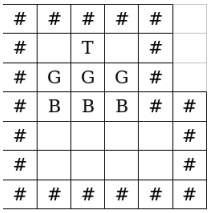
Problem H. Harder Sokoban

Source file name:	`H.c, H.cpp, H.java`
Input:	`Standard`
Output:	`Standard`
Author(s):	`Félix Arreola Rodríguez`

Toby is a terrible person, he likes to steal games from internet to publish them as own. So, for his next robbery Toby found a game called "Sokoban", a game where a player pushes boxes or crates around in a warehouse, trying to get them to storage locations. It doesn't matter which box ends in which storage location, the only requirement is that all boxes final positions must be a storage location. Also, the boxes can "pass" through storage locations freely as they transit to another storage location. Below is a example of the classical Sokoban game:



Where '#' represents a wall (non-walkable), 'G' represents a storage location, 'B' represents a box, and 'T' represents the initial position of the player. The game itself is pretty difficult, because the player cannot pass through boxes (or walls), and only can push a box from his position, because of this, sometimes, an "easy" looking puzzle can be very tricky, like this one:



Yes, indeed, this puzzle could be solved in only 3 moves if the player starting position was "below" the boxes. And because of this, Toby wants to "clone" this game in an different version, changing one simple rule: "The player can transport himself to any empty location". In this way, Toby can publish his "Harder Sokoban" game version.

Your task is help Toby to write a program to calculate the minimum number of box movements that are needed to put every box from his starting positions to a storage location (can be any storage location). Remember, two boxes cannot be pushed at the same time, and two boxes can't occupy the same space, a box can not be pushed through or over a wall or over another box.

Input

The first line contains two integer N and M indicating the size of the warehouse. The following N lines contains exactly M chars, a '#' represents a wall, a '.' (dot) represents a walkable tile, 'G' represents a storage location, 'B' represents a box. The map is always delimited by walls.

- $3 \leq N, M \leq 10$ There will be no more than 3 boxes in the input.

Output

You should output a single line with a integer with the minimum number of box movements to complete the level.

Example

Input	Output
7 6 ###### #...## #GGG## #BBB## #....# #....# ######	3

Problem I. Is a triangle

Source file name:	`I.c, I.cpp, I.java`
Input:	`Standard`
Output:	`Standard`
Author(s):	`Juan Pablo Marin - CUCEI Guadalajara`

JP has N coins in his pocket. While he waits for the contest to start he puts 1 coin in the table, then he puts 2 coins below the one he put before and makes a triangle. Then he does the following:

- Let K be the last amount of coins JP put and $K' = K + 1$. JP Adds K' coins in such way the coins are still arranged in a triangle.
- Repeat the previous step until JP can not add more coins

If at the end of the procedure JP has no coins then it is said that N is a triangular number as all the coins can be arranged in this way to create a triangle.

Your task is to find given the number N if it is a triangular number, in such case you need to find the last value of K' JP used to create the triangle.

Input

The first line of input contains a number T the number of test cases. Each of the next T lines contain a single number N.

- $1 \leq T \leq 100$
- $3 \leq N \leq 10^{18}$

Output

For each test case you must print the value of the last K' used by JP to create the triangle given that N is triangular, print -1 otherwise.

Example

Input	Output
3 55 90 10	10 -1 4

Problem J. Joining cells

Source file name:	`J.c, J.cpp, J.java`
Input:	`Standard`
Output:	`Standard`
Author(s):	`Juan Pablo Marín - CUCEI México`

In Quadradonia as you can imagine everything is a square, all their buildings, all the streets, all the tiles, even the people are squares. In the recent years people have started to arrange their square energy cells in a bigger square of size N in such way that you can maximize the amount of energy that you can take from all the energy cells in town.

An energy cell in Quadradonia allows to put some machine over it and then the machine gets the amount of energy that the energy cell has. The new arrangement Quadradonians are doing allows to put a machine that expands over more than one energy cell and then the machine will get the energy as the sum of all the cells it is overlapping, if the energy a machine gets is greater or equal to it's minimum energy requirement then the machine will power on.

To test the new arrangement a new machine with a minimum energy requirement of X will be built, this machine will have also a square shape, can you find all the different ways you could put a machine in the energy cells arrangements such that the machine will be able to power on?

Input

The first line of input contains two integers separated by a space N and X. The next N lines contain each N integers separated by space representing the amount of energy the cell in that position brings.

- $1 \leq N \leq 1000$
- $1 \leq X \leq 10^9$
- The value of each energy cell $C_{i,j}$ is in the range $1 \leq C_{i,j} \leq 100$

Output

Your program should print a line with an integer number. The number of ways you can put a machine with X minimum energy requirement to be able to power on into the energy cells arrangement.

Example

Input	Output
2 4 3 4 3 2	2

Explanation

In the example test case there you are asked to find the ways to put a machine that requires 4 of energy to power on. There are two ways to put such machine: One is to use all the 2x2 square, the other is to put the machine in the 1x1 square represented by the upper right cell which sums 4 and is the required to power on the machine.

Segunda Fecha Gran Premio México 2018

May 5th 2018

Problems book

General information

This book contains a total of 10 problems; Pages are numbered from 1 to 15 not including this page. Verify your book is complete.

Problem A. Amusement Park

Source file name:	`A.c, A.cpp, A.java, A.py2, A.py3`
Input:	`Standard`
Output:	`Standard`
Author(s):	`Juan Pablo Marín Rosas`

ohn is having his vacations out of town and he decided to visit an amusement park to ride all the exciting attractions that the park has, he is very brave and loves to have adventures from time to time so he can distress from all the work he has at home.

Vacations are finishing and he will have to return home, while he was spending some time in the amusement park he realized he has to bring some souvenirs home, however, as he already have made his baggage he can take only two small items. Being the adventurer man he is, John knows exactly what he will be getting home, the first item he is interested is a fridge magnet so he can remember his trip each time he will get some food from the fridge, the other item is a keychain so that he can remember the trip each time he is going to open a door. There are several souvenir stores in the park on some of them they sell the keychain, on others they sell the fridge magnet but none of them sell both items so John needs to go to two stores to get the souvenirs he wants to bring home.

John is standing near a map of the park, the map depicts the park as a rectangular matrix divided in N x M cells where a '.' represents a cell where John can walk, a 'K' is a place where a store sells keychains a 'F' is a place where a store sells fridge magnets, a 'J' represents where John is standing and 'E' represents the exit. The map shows the amusement park attractions with a '#' and John can not walk over them. John can walk freely on any of the stores, the exit, and the starting point, but he will not exit the amusement park until he reaches the exit having bought both of the souvenirs.

Time is flying and John needs to move fast to get the two items before walking to the exit of the park. Can you help John to find the smallest number of steps he needs to make to exit the amusement park? A step is a movement John makes from the current cell to another cell. John can move between cells only if them share a side.

Input

The first line contains an integer number T, the number of test cases. Each test case starts with a line containing two integer numbers N and M representing the dimensions of the park. The following N lines contain a string with exactly M characters these being one of '.', 'K', 'F', 'J', 'E', or '#'.

- $1 \leq T \leq 10$
- $5 \leq N, M \leq 1000$
- The map has only a starting point for John and an exit
- There will be at least one store that sells keychains and one store that sells fridge magnets
- The map is surrounded by '#'

Output

For each test case your program should print a single line with an integer number representing the minimum number of steps John has to make in order to buy both souvenirs and go to the exit. The input maps have always a solution.

Example

Input	Output
2 5 5 ##### #F.K# #FJF# #..E# ##### 6 6 ###### #.J..# ###F.# #K#F## #..FE# ######	4 11

Problem B. Baker's Gang

Source file name:	`B.c, B.cpp, B.java, B.py2, B.py3`
Input:	`Standard`
Output:	`Standard`
Author(s):	`Eddy Ramírez`

Baker is a cat with a very delicate and exclusive taste for "cat nuggets", which are very expensive, though. Since Baker really wants to increase his share of cat nuggets, he with some other N cats, have planned to steal the nugget factory.

They executed their plan and it was very successful, they got quite a bunch of nuggets. Since the amount of nuggets was huge, Baker and his gang, decided to hide the nuggets in order to come sometime later and make the division in equal parts.

But ... since cats are very suspicious, specially from each other, starting with Baker, they cheated on each other.

Baker was the first, he arrived to the hideaway and counted every nugget, there were K nuggets, he took his share $\frac{K1}{N}$ of the nuggets and he realized that after dividing the amount of nuggets by N, the remainder was 1, so he took that extra nugget for his mother.

The second cat arrived a little later and he counted the nuggets and realized that the remainder after dividing by N was 1, he took his share and the one extra-nugget and left the rest.

The third cat arrived and did the same, with the same surprising fact that when dividing by N the remainder was still 1. He took "his share" and the extra-nugget.

After every single cat did the same thing with the same outcome, they reunited for the final division, and they counted the nuggets, no cat showed surprise for the lack of nuggets they noticed, but after dividing by N the number of remaining nuggets, the remainder was 1 again! So they agreed to give that extra-nugget to a street cat and problem solved.

Your task is to find the number of nuggets K that were originally stolen by Baker and his gang.

Input

The first line is a single integer T with the amount of test cases. The next T lines have a single number N which is the amount of members on Baker's gang.

- $1 \leq T \leq 10^3$
- $3 \leq N \leq 10^6$

Output

The output is the minimum natural number that fits with the description above. As this number can be big write the number modulo 1000000007 ($10^9 + 7$).

Example

Input	Output
1 4	1021

Input	Output
3 5 6 7	15621 279931 5764795

Problem C. Cables

Source file name:	`C.c, C.cpp, C.java, C.py2, C.py3`
Input:	`Standard`
Output:	`Standard`
Adapted by:	`Juan Pablo Marín Rosas From 2001 NEERC problem C`

Mexico regional programming contest are near, yes, they will be in like 6 months but with all the planning required to have success in the event 6 months is too little time.

The Mexico regional finals committee have decided they need to work in order to organize the most fair regional contest ever. It was decided that in order to have the most fair contest they need not to provide only computers with the same operative system and software and with exactly the same hardware but that it is also required that the connection to the judging system is exactly the same.

In order to have exactly the same connection to the judging system each of the K computers that contestants will be using during the contest will be connected directly with a cable to the central hub where the judging system is connected, also, to make sure everything is exactly the same on contestant computers the cable to each computer should have the same length.

To buy network cables, the committee has contacted a local network solutions provider with a request to sell for them a specified number of cables with equal lengths. The provider does not have available the required amount of cables but they have a stock of N cables with different sizes and they can cut these cables if they know the length of the pieces they must cut. After the committee heard this, some thoughts came to their mind about the size of the cable, it would be appropriate that the cable size is as long as possible, in this way they can sit contestants as far from each other as possible, making the contest even more fair.

As the committee is too busy solving more issues with the problem set, you are here to help them by writing a program that will determine the maximal possible length of a cable that can be cut from the cables in the providers stock, to get the required amount of cables.

Input

The input consists of several test cases. The first line of input contains a number T the number of test cases that follows. The following T test cases are as follow: The first line of each test case contains two integer numbers N and K, separated by a space. N is the number of cables in the providers stock, and K is the number of requested cables. The next N lines with one number per line, that specify the length of each cable in the stock in centimeters.

- $1 \leq T \leq 20$
- $1 \leq N \leq 10^4$
- $1 \leq K \leq 10^4$
- All cables in the stock are at least 1 centimeter and at most 10^9 centimeters in length.

Output

For each testcase write to the output the maximal length (in centimeters) of the pieces that network provider may cut from the cables in the stock to get the requested number of pieces.

If it is not possible to cut the requested number of pieces each one being at least one centimeter long, then the output must contain the single number "0" (without quotes).

Example

Input	Output
1 4 11 80 70 40 50	20

Problem D. Divisible repunit

Source file name:	`D.c, D.cpp, D.java, D.py2, D.py3`
Input:	`Standard`
Output:	`Standard`
Author(s):	`Juan Pablo Marín Rosas`

n recreational mathematics, a repunit is a number like 11, 111, or 1111 that contains only the digit 1. The term stands for repeated unit and was coined in 1966 by Albert H. Beiler in his book Recreations in the Theory of Numbers.

It is suspected that any number N ending with any of the digits 1, 3, 7 or 9 have at least one multiple that is a repunit, this is, there is at least one repunit R such that the result of diving R by N has no remainder.

Your task is to help confirm the previous statement. Given a number N that ends with any of the digits 1, 3, 7, or 9 can you find how many digits have the smallest repunit that N can divide?

Input

The first line of input contains a number T, the number of test cases. Each test case is described by a line with a single number N.

- $1 \leq T \leq 10$
- $1 \leq N \leq 10^6$
- It is guaranteed that the last digit of N will be 1, 3, 7 or 9

Output

For each test case print the number of digits that has the minimum repunit that can be divided by N without reminder if such repunit exists, print -1 otherwise.

Example

Input	Output
2 3 7	3 6

Problem E. Examining groups

Source file name:	`E.c, E.cpp, E.java, E.py2, E.py3`
Input:	`Standard`
Output:	`Standard`
Author(s):	`Juan Pablo Marín Rosas`

chools usually divide students in groups usually by age or ability, this has created some polarity on people opinion, some people think this is good so that students can interact with others that may have similar skills and likes in common. Others think this is one of the main things that are dividing society.

John's school does not like to be part of this kind of discussions that's why they will limit groups by the amount of students they can put on their classrooms. Today John's school has N students divided in 1000 groups, each of these groups is represented by an integer number between 1 and 1000 and are groups conformed of students who have common skills or common age.

In order to have more diversified groups John has decided to put all students in a line with N slots numbered from 0 to $N-1$, each student is wearing a shirt that represents which group the student belongs to. After all students are in the line then John ask them to move to a random position making sure that each position in the line has exactly one student. John will take now two positions X and Y and calculate the "diversity score" of the group that they can make by creating a group with students standing in the line from position X to position Y inclusive. The "diversity score" is the number of different groups that the students between X and Y belong to at this moment.

John will be selecting several values for X and Y trying to find good "diversity scores", but as there can be a lot of students it is complicated and time consuming. Can you help John to given the arrangement of the students to find what is the "diversity score" for each selection of X and Y that he makes?

Input

The first line contains a single number T, the number of test cases. Each test case starts with a line with two integer number N and Q being the number of students and the number of selections John will make for X and Y. The next line contains N numbers separated by a space, representing where the $i-th$ element of the line is the group the $i-th$ student in the line belongs to. The next Q lines contain two integer numbers separated by a space representing each of the selectiong for X and Y that John will make.

- $1 \leq N \leq 10^5$
- $1 \leq Q \leq 10^5$
- $0 \leq X \leq Y \leq N$
- Even when the school has 1000 groups, not necessarily all groups have students.

Output

For each test your program should print exactly Q lines each line contains the "diversity score" based on the definition above for each of the X and Y selections John has made on the test case.

Example

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

Problem F. Frogs secret meeting

Source file name:	`F.c, F.cpp, F.java, F.py2, F.py3`
Input:	`Standard`
Output:	`Standard`
Adapted by:	`Juan Pablo Marín Rosas From 2007 NWERC problem B`

Lately there have been a lot of problems about frogs. It seems this is not some kind of coincidence. Frogs have been meeting secretly to plan how to get a problem set with just frogs problems on "El Gran Premio".

These reunions can lead to have a very boring set, all talking about frogs, jumps and rocks. You want to stop it and have studied how frogs behave before their meetings so you are now ready to interrupt one of the meetings and avoid at all cost that the frogs take the problem set for the next competition.

The frogs plan their reunion in a pond where a number of frogs are standing on a number of rocks. The frogs can have the meeting in a rock where all the frogs can stand at the same time. The frogs do not like to get into the water because it can make a lot of noise and then their meeting would not be secret that's why they jump from one rock to another as long as their distance is less or equal to the maximum distance D a frog can jump.

Frogs put a lot of caution before their meetings, they know that too much jumps may look suspicious so they avoid jumping out too much from any rock because doing much jumps from the same rock could look suspicious to anyone looking at the pond. As frogs are experts hiding their plans they already know exactly how many times they can jump out of each rock before jumping from that rock looks suspicious, it is not suspicious that a frog lands on a rock, only when they jump out of the rocks.

Can you find all the rocks where the frogs can have their secret meeting?

Input

On the first line one positive number T the number of test cases. Each of the T test cases start with one line with the integer N and a floating-point number D, denoting the number of rocks in the pond and the maximum distance a frog can jump. Each of the next N lines contains X_i, Y_i, n_i and m_i, denoting for rock i its X and Y coordinate, the number of frogs on it and the maximum number of times a frog can jump off this rock before it looks suspicious.

- $1 \leq T \leq 100$
- $1 \leq N \leq 100$
- $1 \leq D \leq 10^5$
- $-10^4 \leq X_i, Y_i \leq 10^4$
- $0 \leq n_i \leq 10$
- $1 \leq m_i \leq 200$

Output

For each test case print one line containing a space-separated list of 0-based indices of the rocks on which the frogs can have their meeting this list should be in increasing order. If no such rock exists, output a line with the single number -1.

Example

Input	Output
2 5 3.5 1 1 1 1 2 3 0 1 3 5 1 1 5 1 1 1 5 4 0 1 3 1.1 -1 0 5 10 0 0 3 9 2 0 1 1	1 2 4 -1

Problem G. Generator strings

Source file name:	`G.c, G.cpp, G.java, G.py2, G.py3`
Input:	`Standard`
Output:	`Standard`
Author(s):	`Juan Pablo Marín Rosas`

string S of lower case characters is said to be a generator string of the string S' if S' has the same length of S and if each character that appears on S appears the same amount of times on S'.

As you can see the this definition leads that S is a generator string of S, also if S is a generator string of S' then S' is also a generator string of S.

Your task is given a list of N strings to determine how many pairs of indexes (A, B) you can take such that the $A-th$ element of the list is a generator of the $B-th$ element?

Input

The input consists of several test cases. The first line of input contains a number T the number of test cases. Each test case starts with a line containing a single integer N, followed by N lines where each line contains a string from the list, each string contains only lower case characters.

- $1 \leq N \leq 10^5$
- The length of a string L is in the range $1 \leq L \leq 100$

Output

For each test case your program should print an integer number, the number of pairs (A, B) that suffices the requirements listed above.

Example

Input	Output
2 3 abc bca cab 5 abc ab a b b	9 7

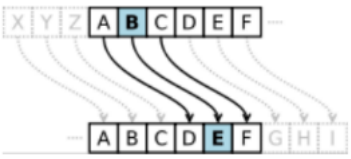
Problem H. Hiding messages

Source file name:	`H.c, H.cpp, H.java, H.py2, H.py3`
Input:	`Standard`
Output:	`Standard`
Author(s):	`Facultad de Informática Culiacán UAS`

n the ACM school students use to send message between each others. They don't want teachers to notice what these messages say. Students are really clever so they have created a coding mechanism that will allow them to send messages without teachers noticing what they say.

Students will code the message substituting each of the upper case letters in the message with the letter that is in a fixed number K of positions ahead in the alphabet.

As an example if $K = 3$ then the letter A will be substituted with the letter D, B will be substituted with the letter E, and so on. It is supposed that the alphabet is circular in such way that A is after Z, so the letter Z would be substituted with the letter C, you can see an idea of this in the following figure.



As mentioned above, students only substitute upper case letters all other characters in the message are not changed.

Your task is, given student messages to code each of those messages using the method described.

Input

The first line contains a number T the number of test cases. Each of the test cases start with two numbers N and K being the number of lines in the message and the number of positions that the letters will be shifted. The next N lines in the test case will contain a string that represents a line in the students message.

- $1 \leq T \leq 10$
- $1 \leq N \leq 100$
- $1 \leq K \leq 1000$
- Each line of the student message will contain any printable ASCII character excluding the new line character.

Output

For each line on each of the messages your program should print the encoded message using the method described above.

Example

Input	Output
2 1 1 UN TEXTO, y algo MAS 1 120 AlexAndeR	VO UFYUP, y algo NBT QlexQndeH

Problem I. Intergalactic tourism

Source file name:	`I.c, I.cpp, I.java, I.py2, I.py3`
Input:	`Standard`
Output:	`Standard`
Author(s):	`Juan Pablo Marín Rosas`

ow is year 4018 and we have a lot of technological advances, a lot of them related to space and time travelling. The main technology that enables space and time travelling are transporters, basically you can stand on a transporter and be transported in no time to any planet that you decide, this makes intergalactic tourism (travelling to planets) something really easy as you can go and travel to planets without wasting time.

Given all the commodities it is now a must to do intergalactic tourism, the complex thing is that the transporters have restrictions established that will not allow you to travel to planet X unless you have travelled before to all planets that restrict access to planet X. For example if the planets that restrict access to planet 2 are planets 1 and 3 you should visit planets 1 and 3 before the transporter gives you access to travel to planet 2.

You found based on the transporters restriction that there may be more than one way to visit all the N planets in the planetary system, you want to find all of them.

Can you find given the list of planets you must visit before visiting each planet all the different ways you can visit the N planets considering that on each way you will be visiting each planet only once?

Input

The first line contains a single integer T with the number of test cases. Each of the test cases start with a line with the number N the number of planets in the planetary system. Each of the next N start with an integer number R_i the number of planets that restrict access to this planet followed by R_i numbers representing the planets that restrict access to this planet, all numbers in the line are separated by a space.

- $1 \leq T \leq 10$
- $3 \leq N \leq 10$
- $0 \leq R_i \leq N - 1$
- Planets are numbered from 0 to $N - 1$
- It is guaranteed that there is always at least one way to visit all planets.

Output

For each test case print all the ways you could visit planets based on the restrictions mentioned above. Print them in lexicographical order this is: Given two different ways of visiting the planets, $a_1, a_2 \dots a_k$ and $b_1, b_2 \dots b_k$, the first one is smaller than the second one for the lexicographical order, if $a_i < b_i$, for the first i where a_i and b_i differ.

Example

Input	Output
3 3 0 0 2 0 1 3 0 0 1 1 6 2 4 5 2 3 4 1 5 1 2 0 0	0 1 2 1 0 2 0 1 2 1 0 2 1 2 0 4 5 0 2 3 1 4 5 2 0 3 1 4 5 2 3 0 1 4 5 2 3 1 0 5 2 3 4 0 1 5 2 3 4 1 0 5 2 4 0 3 1 5 2 4 3 0 1 5 2 4 3 1 0 5 4 0 2 3 1 5 4 2 0 3 1 5 4 2 3 0 1 5 4 2 3 1 0

Problem J. Joy and misery

Source file name:	`J.c, J.cpp, J.java, J.py2, J.py3`
Input:	`Standard`
Output:	`Standard`
Author(s):	`Juan Pablo Marín Rosas`

In a land far far away pirates say that being in union will bring joy, that's why pirates usually come in pairs and have put an attribute of good luck to even numbers. In the other hand, bad luck is represented by odd numbers that will bring misery as they represent for them pairs that have been divided.

That is why every time a pirate sees an even number he will feel joy, and every time he sees an odd number he will feel misery. John the most young pirate in the dark ship is having trouble to sleep so the other pirates have suggested him to start counting sheep at night. Oh! how unfortunate is John, as when he is counting the sheep he feels joy and misery all night until he falls asleep.

Can you count how many times will John feel joy and how many times he will feel misery if he starts counting the sheep at the number X and falls asleep when he reaches the sheep with the number Y

Input

The first line of input has a single number T, the number of test cases. Each test case will consist of a single line with two numbers separated by space X and Y.

- $1 \leq T \leq 1000$
- $1 \leq X \leq Y \leq 10^6$

Output

For each test case you should print two numbers separated by a space, the first of them being the number of times John will feel joy while counting sheep, the second number is the number of times John will feel misery while counting sheep.

Example

Input	Output
4 1 1 1 4 5 11 6 12	0 1 2 2 3 4 4 3

acm International Collegiate Programming Contest 2018

Maratón de programación de la SBC – ACM ICPC – 2018

El set de problemas de este cuaderno es utilizado simultáneamente en las siguientes competencias:
Maratona de Programação da SBC 2018
Tercera Fecha Gran Premio de México 2018
Tercera Fecha Gran Premio de Centroamérica 2018

15 de Septiembre de 2018

Cuaderno de Problemas

Información general

Este cuaderno contiene 13 problemas; Las páginas están numeradas de 1 a 22, sin contar esta página. Verifique que su cuaderno está completo.

A) Sobre los nombres de los programas
1) Para soluciones en C/C++ y Python, el nombre del archivo de código fuente no es significativo, puede ser cualquier nombre.
2) Si su solución es en Java, el archivo debe ser llamado: *codigo_de_problema*`.java` donde *codigo_de_problema* es la letra mayúscula que identifica al problema. Recuerde que en Java el nombre de la clase principal debe ser igual que el nombre del archivo.
3) Si su solución es en Kotlin, el archivo debe ser llamado: *codigo_de_problema*`.kt` donde *codigo_de_problema* es la letra mayúscula que identifica al problema. Recuerde que en Kotlin el nombre de la clase principal debe ser llamado igual que el nombre del archivo

B) Sobre la entrada
1) La entrada de su programa debe ser leída de *entrada standard*.
2) La entrada está compuesta de un único caso de prueba, descrito en un número de línea que depende del problema.
3) Cuando una línea de entrada contiene varios valores, estos están separados por un único espacio en blanco; la entrada no contiene ningún otro espacio en blanco.
4) Cada línea, incluyendo la última, contiene exactamente un caracter de final-de-línea.
5) El final de la entrada coincide con el final del archivo.

C) Sobre la salida
1) La salida de su programa debe ser escrita en *salida standard*.
2) Cuando una línea de salida contiene varios valores, estos deben ser separados por un único espacio en blanco; la salida no debe contener ningún otro espacio en blanco.
3) Cada línea, incluyendo la última, debe contener exactamente un caracter de final-de-línea.

Problema A

Aventurándose en Slackline

Beltrano recientemente se interesó en el *slackline*. Slackline es un deporte de equilibrio sobre una cinta elástica estirada entre dos puntos fijos, lo que le permite al practicante caminar y hacer maniobras sobre la cinta. Durante las vacaciones todo lo que Beltrano ha querido hacer es practicar y para eso se ha ido a la granja de un amigo, donde hay una plantación de eucaliptos.

La plantación está muy bien organizada. Los eucaliptos están acomodados en N filas con M árboles en cada una. Hay un espacio de un metro entre cada fila y los árboles en cada fila están todos perfectamente alineados con un espacio de un metro entre ellos.

Beltrano va a montar el slackline uniendo dos árboles. Al montar el slackline a Beltrano no le gusta que la distancia entre los dos árboles sea muy pequeña debido a que las mejores maniobras requieren que la cinta tenga al menos L metros de longitud. Tampoco es posible estirar la cinta demasiado ya que tiene una longitud máxima de R metros. Tenga en cuenta que al estirar la cinta entre los dos árboles elegidos no debería haber ningún otro árbol en la línea formada, de lo contrario no sería posible utilizar toda la cinta para realizar las maniobras.

A Beltrano le gustaría saber de cuantas formas diferentes es posible montar el slackline usando los árboles de la granja. Dos formas se consideran diferentes si por lo menos uno de los árboles donde se amarra la cinta es diferente.

Entrada

La entrada consiste de una única linea que contiene cuatro números enteros, N, M, L, R, representando respectivamente el número de filas y columnas de la plantación y las longitudes mínima y máxima del slackline ($1 \leq N, M \leq 10^5$; $1 \leq L \leq R \leq 10^5$).

Salida

Su programa debe producir una sola línea con un número entero que representan de cuantas maneras diferentes se puede poner el slackline. Como el resultado puede ser grande, la respuesta debe ser ese número módulo $10^9 + 7$.

Ejemplo de entrada 1	**Ejemplo de salida 1**
`2 2 1 1`	`4`

Ejemplo de entrada 2	**Ejemplo de salida 2**
`2 3 1 4`	`13`

Ejemplo de entrada 3	**Ejemplo de salida 3**
`3 4 1 4`	`49`

Problema B

Buscando ganar

Usar canicas como moneda no dio buenos resultados en Cubiconia. En el intento de redimirse con sus amigos, después de robar sus canicas, el emperador decidió invitar a todos a su palacio para una noche de juegos.

Naturalmente, los juegos utilizan canicas, después de todo el emperador necesita encontrarles alguna utilidad. N canicas son extendidas en un tablero cuyas filas son enumeradas de 0 a L y sus columnas enumeradas de 0 a C.

Los jugadores alternan turnos y en cada turno el jugador debe escoger una de las canicas para moverla. El primer jugador que logre mover una canica a la posición $(0, 0)$ es el vencedor. Para que el juego sea interesante, los movimientos son limitados; de lo contrario, el primer jugador siempre movería la canica a la posición $(0, 0)$ y ganaría. Un movimiento consiste en escoger un entero u mayor que 0 y una canica, cuya posición denotaremos (l, c), que puede ser movida a una de las siguientes posiciones, siempre y cuando estén dentro del tablero:

- $(l - u, c)$;
- $(l, c - u)$;
- $(l - u, c - u)$.

Note que más de una canica puede ocupar la misma posición en el tablero.

Como al emperador no le gusta perder, usted debe ayudar a determinar en cuales partidas él debe participar. Como es de esperar, siempre que el emperador juega, tiene el primer turno. Suponiendo que todos juegan de manera óptima, su programa debe analizar la distribución inicial de las canicas en el tablero y determinar si es posible que el emperador gane en caso de que juegue.

Entrada

La primera línea contiene un entero N $(1 \leq N \leq 1000)$. Cada una de las siguientes N líneas contiene dos enteros l_i y c_i indicando en cual fila y columna está la i-ésima canica en el tablero. $(1 \leq l_i, c_i \leq 100)$.

Salida

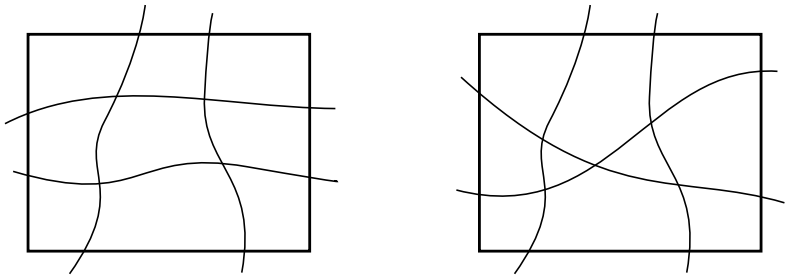
El programa debe imprimir una única línea con el caracter `Y` en caso de que sea posible que el emperador gane el juego o `N` en caso contrario.

Ejemplo de entrada 1	**Ejemplo de salida 1**
2 1 3 2 3	Y

Ejemplo de entrada 2	**Ejemplo de salida 2**
1 1 2	N

Problema C
Cortador de Pizza

El abuelo Giusseppe ganó de regalo un cortador profesional de pizza, de aquellos tipo carretilla, y, para celebrar, ¡ Ha hecho una pizza rectangular gigante para sus nietos!. Él siempre ha cortado sus pizzas en pedazos haciendo líneas continuas como cortes las cuales no necesariamente son rectilíneas, y son de dos tipos: Unas las empieza en el borde izquierdo de la pizza, siguen de manera monótona hacia la derecha hasta terminar en el borde derecho; las otras comienzan en el borde inferior, siguen de manera monótona hacia arriba para terminar en el borde superior. Pero el abuelo Giusseppe siempre seguía una regla: dos cortes del mismo tipo nunca se podían interceptar. Vea un ejemplo con 4 cortes, dos de cada tipo, en la parte izquierda de la imagen, que dividen una pizza en 9 pedazos.



Resulta que el abuelo Giusseppe simplemente ama la geometría, topología, combinatoria y cosas así; es por eso que ha decidido mostrar a los niños que puede obtener más pedazos con el mismo número de cortes si fuera permitido que cortes del mismo tipo se interceptaran. La parte derecha de la imagen muestra, por ejemplo, que si los dos cortes del tipo que van de izquierda a derecha se pudieran interceptar entonces la pizza sería dividida en 10 pedazos.

El abuelo Giusseppe ha descartado la regla, pero no hará cortes aleatorios. Los cortes, además de ser de uno de los dos tipos deberán obedecer las siguientes restricciones:

- Dos cortes tienen como máximo un punto de intersección y, si lo tienen, es porque los cortes se cruzan en ese punto;
- Tres cortes no se interceptan en un mismo punto;
- Dos cortes no se interceptan en el borde de la pizza;
- Un corte no intercepta una esquina de la pizza.

Dados los puntos de inicio y fin de cada corte, su programa debe contar el número de pedazos resultantes de los cortes del abuelo Giusseppe.

Entrada

La primera línea de entrada contiene dos números enteros X y Y, $(1 \leq X, Y \leq 10^9)$, representando las coordenadas (X, Y) de la esquina superior derecha de la pizza. La esquina inferior izquierda tiene siempre coordenadas $(0, 0)$. La segunda línea contiene dos números enteros H y V, $(1 \leq H, V \leq 10^5)$, indicando, respectivamente, el número de cortes que van de izquieda a derecha, y el número de cortes que van de abajo hacia arriba. Cada una de las siguientes H líneas contiene dos números enteros Y_1 y Y_2 definiendo las ordenadas de encuentro de los lados verticales de la pizza con un corte que va del lado izquierdo en la ordenada Y_1 hasta el lado derecho en la ordenada Y_2. Cada una de las siguientes V líneas contiene dos números enteros X_1 y X_2 definiendo las abscisas en las que se encuentran los lados horizontales de la pizza con un corte que va del lado inferior en la abscisa X_1 hasta el lado superior en la abscisa X_2.

Salida

Imprima una línea que contiene un número entero representando el número de pedazos resultantes.

Ejemplo de entrada 1	**Ejemplo de salida 1**
3 4 3 2 1 2 2 1 3 3 1 1 2 2	13

Ejemplo de entrada 2	**Ejemplo de salida 2**
5 5 3 3 2 1 3 2 1 3 3 4 4 3 2 2	19

Ejemplo de entrada 3	**Ejemplo de salida 3**
10000 10000 1 2 321 3455 10 2347 543 8765	6

Problema D

Desvelando Monty Hall

En el escenario de un programa de auditorio hay tres puertas cerradas: puerta 1, puerta 2 y puerta 3. Detrás de una de esas puertas hay un coche, detrás de cada una de las otras dos puertas hay un chivo. La producción del programa sortea aleatoriamente la puerta donde va a estar el coche, sin hacer trampa. Solamente el presentador del programa sabe dónde está el coche. Él pide al jugador elegir una de las puertas. Ahora, como sólo hay un coche, detrás de una de las dos puertas que el jugador no eligió, ¡tiene que haber un chivo!

Por lo tanto, el presentador siempre puede hacer lo siguiente: entre las dos puertas que el jugador no eligió, abre la que tiene el chivo, de modo que el jugador y los espectadores pueden ver el chivo. El presentador entonces pregunta al jugador "¿Desea cambiar su puerta por la otra que aún está cerrada?" ¿Es ventajoso cambiar o no? El jugador quiere quedarse con la puerta que tiene el coche, ¡claro!.

Paulino vio una demostración rigorosa de que la probabilidad de que el carro esté detrás de la puerta que el jugador eligió inicialmente es 1/3 y que la probabilidad de que el carro esté detrás de la otra puerta, que aún está cerrada y que el jugador no eligió inicialmente es 2/3, y que por lo tanto el intercambio es ventajoso. Paulino no se conforma, su intuición le dice que la probabilidad es 1/2 para ambas puertas que aún están cerradas.

En este problema, para acabar con la duda de Paulino, vamos a simular este juego miles de veces y contar cuántas veces el jugador ganó el coche. Vamos a suponer que:

- El jugador siempre elige inicialmente la puerta 1;
- El jugador siempre cambia de puerta después de que el presentador revela al chivo abriendo una de las dos puertas que no fueron elegidas inicialmente.

Con estas condiciones, en un juego, dado el número de la puerta que contiene el coche, podemos saber exactamente si el jugador va a ganar o no el coche.

Entrada

La primera línea de entrada contiene un número entero N ($1 \leq N \leq 10^4$), que representa el número de juegos a simular. Cada una de las siguientes N líneas contiene un número entero: 1, 2 o 3; representando el número de puerta que tiene el coche en ese juego.

Salida

Su programa debe imprimir una única línea conteniendo un número entero que representa el número de veces que el jugador ganó el coche en la simulación, suponiendo que el jugador siempre elige inicialmente la puerta 1 y que siempre cambia de puerta después que el presentador revela un chivo abriendo una de las puertas que no fueron elegidas inicialmente.

Ejemplo de entrada 1	**Ejemplo de salida 1**
5 1 3 2 2 1	3

Ejemplo de entrada 2	**Ejemplo de salida 2**
1 1	0

Ejemplo de entrada 3	**Ejemplo de salida 3**
15 3 2 3 1 1 3 3 2 2 1 2 3 2 1 1	10

Problema E

Enigma

Dada una configuración inicial, la máquina criptográfica alemana Enigma, de la segunda guerra mundial, sustituía cada letra digitada en el teclado por otra letra. La sustitución era bastante compleja, pero la máquina tenía una vulnerabilidad: ¡una letra nunca sería sustituida por ella misma! esa vulnerabilidad fue explorada por Alan Turing, quien trabajo en el criptoanálisis de Enigma durante la guerra. El objetivo era encontrar una configuración inicial de la máquina usando la suposición de que el mensaje contenía cierta expresión usual de comunicación, como por ejemplo la palabra `ARMADA`. Esas expresiones eran llamadas *cribs*. Si el mensaje cifrado era, por ejemplo, `FDMLCRDMRALF`, el trabajo de probar las posibles configuraciones de la máquina era simple porque la palabra `ARMADA`, si estuviese en ese mensaje cifrado, solo podría estar en dos posiciones, ilustradas en la tabla de abajo con una flecha. Las otras cinco posiciones no podrían corresponder al mismo *crib* `ARMADA` porque al menos una letra del *crib*, subrayada en la tabla de abajo, es emparejada con su correspondiente en el mensaje cifrado; como Enigma nunca sustituirá una letra por ella misma, esas cinco posiciones son descartadas por las pruebas.

F	D	M	L	C	R	D	M	R	A	L	F
A	R	M	A	D	A						
	A	R	M	A	D	A	←				
		A	R	M	A	D	A				
			A	R	M	A	D	A	←		
				A	R	M	A	D	A		
					A	R	M	A	D	A	
						A	R	M	A	D	A

En este problema, dado un mensaje cifrado en un *crib*, la tarea es calcular el número de posiciones posibles para el *crib* en el mensaje cifrado.

Entrada

La primera linea de entrada contiene el mensaje cifrado, que es una secuencia de por lo menos una letra y máximo 10^4 letras. La segunda línea contiene el *crib*, que es una secuencia de por lo menos una letra y máximo el mismo número de letras que el mensaje. El *crib* y el mensaje únicamente incluyen las 26 letras del abecedario, mayúsculas y sin tildes.

Salida

Imprima una línea con un entero, indicando el número de posiciones posibles para un *crib* en el mensaje cifrado.

Ejemplo de entrada 1	**Ejemplo de salida 1**
FDMLCRDMRALF ARMADA	2

Ejemplo de entrada 2	**Ejemplo de salida 2**
AAAAABABABABABABABA ABA	7

Problema F

Festival

Los festivales de música deberían ser pura diversión, sin embargo algunos se tornan tan grandes que terminan causando dolores de cabeza a los asistentes. El problema es que son tantas presentaciones buenas en varios escenarios que la simple tarea de escoger a qué presentaciones asistir se torna compleja.

Para ayudar a los asistentes de los festivales de música, Fulano decidió crear una aplicación que, después de evaluar las canciones escuchadas en los servicios de streaming favoritos del usuario, sugiere a cuáles presentaciones asistir de modo que no exista otra combinación de presentaciones mejor de acuerdo a los siguientes criterios:

- Para aprovechar la experiencia al máximo, es importante asistir a cada una de las presentaciones hasta el final de la misma;
- Ir al festival y no asistir a alguno de los escenarios está fuera de consideración;
- Para garantizar que la selección de los artistas será compatible con el usuario, se deben contar cuántas canciones de cada artista el usuario conoce por haberlas escuchado en los servicios de streaming. El total de las canciones conocidas de los artistas elegidos debe ser lo más grande posible.

Infelizmente, la versión beta de la aplicación recibió varias críticas, porque los usuarios podían pensar en selecciones mejores que las sugeridas. La tarea en este problema es ayudar a Fulano a escribir un programa que, dadas las descripciones de los shows que ocurren en cada escenario, calcule la lista ideal para cada usuario.

El tiempo de desplazamiento entre cada escenario es ignorado; por lo tanto desde que no haya intersección entre el horario de dos presentaciones escogidas, se considera posible asistir a ambos. En particular, si un show acaba exactamente cuando otro show empieza, es posible asistir a ambos.

Entrada

La primera línea contiene un número entero $1 \leq N \leq 10$ representando el número de escenarios. Las siguientes N líneas describen las presentaciones que ocurren en cada escenario. La i-ésima de ellas está compuesta por un entero $M_i \geq 1$, representando el número de presentaciones marcados para el i-ésimo escenario seguido por M_i descripciones de las presentaciones. Cada descripción de la presentación contiene 3 enteros i_j, f_j y o_j ($1 \leq i_j < f_j \leq 86400$ y $1 \leq o_j \leq 1000$), representando respectivamente los horarios de inicio y fin de la presentación y el número de canciones del cantante presentándose que fueron previamente escuchadas por el usuario.La suma de los M_i no excederá 1000.

Salida

Su programa debe producir una sola línea con un entero que representan el total de canciones previamente escuchadas de los artistas elegidos o -1 en caso de que no haya solución valida.

Ejemplo de entrada 1	**Ejemplo de salida 1**
3 4 1 10 100 20 30 90 40 50 95 80 100 90 1 40 50 13 2 9 29 231 30 40 525	859

Ejemplo de entrada 2	**Ejemplo de salida 2**
3 2 13 17 99 18 19 99 2 13 14 99 15 20 99 2 13 15 99 18 20 99	-1

Problema G

Gasolina

Terminada la huelga de los camioneros, usted y los demás especialistas en logística de Nlogonia tienen la tarea de planear el reabastecimiento de los puestos de servicio de la ciudad. Para ello, se recogió información sobre las reservas de R refinerías y sobre la demanda de los P puestos de servicio de gasolina. Además, hay restricciones contractuales que hacen que algunas refinerías no puedan atender algunos puestos de servicio; cuando una refinería puede reabastecer un puesto, se sabe el menor tiempo del recorrido para transportar el combustible de un lugar a otro.

La tarea de los especialistas es minimizar el tiempo de abastecimiento de todos los puestos, satisfaciendo completamente sus demandas. Las refinerías tienen una cantidad suficientemente grande de camiones, de modo que es posible asumir que cada camión hará máximo un viaje, de una refinería a un puesto de gasolina. La capacidad de cada camión es mayor que la demanda de cualquier puesto, pero puede ser necesario usar más de una refinería para atender la demanda de un puesto.

Su tarea es encontrar el tiempo mínimo en el que es posible abastecer totalmente todos los puestos de servicio, respetando las restricciones de las refinerías.

Entrada

La primera línea de entrada tiene tres enteros, P, R y C, el número de puestos de servicio, el número de refinerías y el número de pares de refinerías y puestos cuyo tiempo de recorrido será dado, ($1 \leq P, R \leq 1000$ e $1 \leq C \leq 20000$) respectivamente. La segunda línea contiene P enteros D_i ($1 \leq D_i \leq 10^4$), representando las demandas, en litros de gasolina, de los puestos $i = 1, 2, \ldots, P$, en ese orden. La tercera línea contiene R enteros E_i ($1 \leq E_i \leq 10^4$), representando las reservas, en litros de gasolina, de las refinerías $i = 1, 2, \ldots, R$, en ese orden. Finalmente, las últimas C líneas describen los tiempos de recorrido, en minutos, entre puestos y refinerías. Cada una de esas líneas contiene tres enteros, I, J e T ($1 \leq I \leq P$ y $1 \leq J \leq R$ e $1 \leq T \leq 10^6$), donde I identifica un puesto, J identifica una refinería y T el tiempo de recorrido de un camión de la refinería J a un puesto I. No habrá pares (J, I) repetidos. No todos los pares serán informados; en caso de que un par no sea informado, hay restricciones contractuales que impiden a la refinería atender el puesto.

Salida

Imprima un entero T que indique el tiempo mínimo en minutos para que todos los puestos de servicio sea reabastecidos totalmente. En caso de que no sea posible, imprima -1.

Ejemplo de entrada 1	**Ejemplo de salida 1**
3 2 5 20 10 10 30 20 1 1 2 2 1 1 2 2 3 3 1 4 3 2 5	4

Ejemplo de entrada 2	**Ejemplo de salida 2**
3 2 5 20 10 10 25 30 1 1 3 2 1 1 2 2 4 3 1 2 3 2 5	5

Ejemplo de entrada 3	**Ejemplo de salida 3**
4 3 9 10 10 10 20 10 15 30 1 1 1 1 2 1 2 1 3 2 2 2 3 1 10 3 2 10 4 1 1 4 2 2 4 3 30	-1

Ejemplo de entrada 4	**Ejemplo de salida 4**
1 2 2 40 30 10 1 1 100 1 2 200	200

Problema H

Hipótesis Policial

El sistema de transporte público de Nlogonia cuenta con una red express que conecta los principales puntos turísticos del país. Se utilizan $N - 1$ trenes bala para conectar N atracciones de modo que a partir de cualquiera de los puntos turísticos es posible llegar a cualquier otro punto usando la red.

Como en cualquier otro lugar del mundo, es común que haya graffiti en las estaciones del tren. Lo que llamó la atención de la policía del país es el hecho de que en cada una de las estaciones es posible encontrar exactamente una letra pintada con un estilo en específico. La hipótesis es que los criminales pueden estar alterando los graffitis como medio de comunicación y por lo tanto se ha decidido crear un sistema capaz de monitorear los graffitis y sus alteraciones.

Dado un patrón `P`, la descripción de las conexiones entre las estaciones y las letras sospechosas en cada una de las estaciones, tu tarea es escribir un programa capaz de realizar las siguientes operaciones:

- `1` u v: imprime cuantas ocurrencias del patrón `P` existen en el camino de u a v si miramos las letras sospechosas asociadas a los vértices consecutivos en el camino;
- `2` u `x`: Cambia la letra del graffiti de la estación u por la letra `x`.

Entrada

La primera línea contiene dos números enteros N y Q ($1 \leq N, Q \leq 10^5$), representando el número de estaciones y la cantidad de operaciones que deben ser procesadas. La segunda línea contiene el patrón P ($1 \leq |P| \leq 100$). La tercera línea contiene una cadena S con N caracteres representando las letras que se encuentran inicialmente asociadas a cada una de las N estaciones. Cada una de las siguientes $N - 1$ líneas contiene dos números enteros u y v indicando que existe un tren bala que conecta a las estaciones u y v. Las Q líneas siguientes describen las operaciones que deben ser procesadas conforme a las operaciones descritas previamente.

Salida

El programa debe imprimir una línea para cada operación de tipo `1` conteniendo un número entero que representa el número de ocurrencias del patrón `P` en el camino analizado.

Ejemplo de entrada 1	**Ejemplo de salida 1**
4 4 xtc xtzy 1 2 2 3 3 4 1 1 3 2 3 c 1 1 3 1 3 1	0 1 0

Ejemplo de entrada 2	**Ejemplo de salida 2**
6 7 lol dlorlx 1 2 1 3 3 4 3 5 5 6 1 2 6 2 3 l 2 6 l 2 5 o 1 2 6 2 1 o 1 6 2	0 1 2

Ejemplo de entrada 3	**Ejemplo de salida 3**
5 2 aba ababa 1 2 2 3 3 4 4 5 1 1 5 1 5 1	2 2

Problema I

Interruptores

En el panel de control de un gran anfiteatro existen N interruptores, numerados de 1 a N. que controlan las M lámparas del local, enumeradas de 1 a M. Note que el número de interruptores y lámparas no es necesariamente el mismo porque cada interruptor está asociado a un conjunto de lámparas y no sólo a una lámpara. Cuando un interruptor es accionado, el estado de cada una de las lámparas asociadas es invertido. Es decir, aquellas apagadas se encienden y las encendidas se apagan.

Algunas lámparas están encendidas inicialmente y el encargado de seguridad del anfiteatro necesita apagar todas las lámparas. Él comenzó intentando accionar los interruptores manera aleatoria, pero como no estaba consiguiendo apagar todas las lámparas al mismo tiempo decidió seguir una estrategia fija. Él va a accionar los interruptores en la secuencia $1, 2, 3, \ldots, N, 1, 2, 3, \ldots$ es decir, cada vez después de accionar el interruptor con número N, él reanuda la secuencia a partir del interruptor 1. Él pretende accionar los interruptores, siguiendo esa estrategia, hasta que todas las lámparas estén apagadas al mismo tiempo (momento en el que parará de accionar los interruptores). ¿Será que esa estrategia le va a funcionar?

En este problema, dadas las lámparas encendidas inicialmente y dados los conjuntos de lámparas que están asociados a cada interruptor, la tarea es calcular el número de veces que el encargado de seguridad va accionar los interruptores. En caso de que la estrategia nunca apague todas las lámparas al mismo tiempo, el programa debe imprimir -1.

Entrada

La primera línea contiene dos enteros N e M ($1 \leq N, M \leq 1000$) representando, respectivamente, el número de interruptores y el número de lámparas. La segunda línea contiene un entero L ($1 \leq L \leq M$) seguido por L enteros diferentes X_i ($1 \leq X_i \leq M$), representando las lámparas encendidas inicialmente. Cada una de las N líneas siguientes contienen un entero K_i ($1 \leq K_i \leq M$) seguido por K_i enteros diferentes Y_i ($1 \leq Y_i \leq M$),representando las lámparas asociadas al interruptor i ($1 \leq i \leq N$).

Salida

El programa debe imprimir una única línea con un entero representando el número de veces que el encargado de seguridad va accionar los interruptores, siguiendo la estrategia descrita, hasta que todas las lámparas estén apagadas al mismo tiempo. Si ese caso nunca pasa, imprimir -1.

Ejemplo de entrada 1	**Ejemplo de salida 1**
6 3 2 1 3 3 1 2 3 2 1 3 2 1 2 2 2 3 1 2 3 1 2 3	5

Ejemplo de entrada 2	**Ejemplo de salida 2**
3 3 2 2 3 1 3 2 1 2 1 2	-1

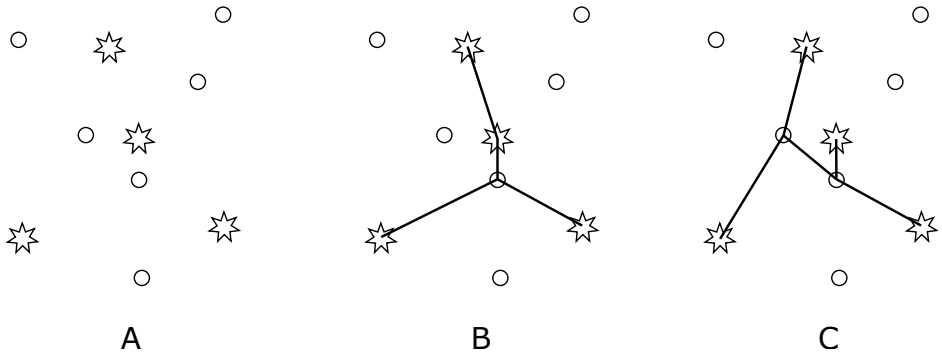
Problema J

Juntando capitales

Un reino lejano posee N ciudades, dentro de las cuales K son capitales. El rey Richard quiere construir líneas de transmisión, cada una de ellas uniendo dos ciudades. Es necesario que haya un camino, es decir, una secuencia de líneas de transmisión entre cualquier par de capitales.

Cada línea de transmisión posee un costo asociado, que es la distancia euclidiana entre las ciudades que la línea de transmisión conecta. Como el rey es un poco ávaro, el desea que las líneas de transmisión sean creadas de modo tal que el costo total (suma de los costos de las líneas) sea lo menor posible.

La figura, en la parte A, muestra un ejemplo de reino con $N = 10$ ciudades, siendo $K = 4$ capitales. El jefe de ingeniería del reino presentó al rey la solución mostrada en la parte B, el cual minimiza el costo total. Pero al rey no le gusto ver una capital con más de una línea de transmisión. Entonces, determino una nueva regla: una capital solo puede estar unida únicamente a otra ciudad. De esa manera y después de mucho trabajar, el jefe de ingeniería ha presentado una nueva solución, ilustrada en la parte C de la figura. Pero ¡ no está seguro de que esa sea la solución óptima y necesita de tu ayuda!



Dadas las coordenadas de las ciudades, tu programa debe calcular el costo total mínimo posible para construir las líneas de transmisión de modo que todo par de capitales esté unida por un camino y que cada capital este unida solo a una ciudad.

Entrada

La primera línea de entrada contiene dos números enteros N y K, $4 \leq N \leq 100$ y $3 \leq K < \min(10, N)$, representando, respectivamente el número de ciudades y el número de capitales. Las siguientes N líneas tienen, cada una, dos enteros X y Y, $-1000 \leq X, Y \leq 1000$, representando las coordenadas de una ciudad. Las primeras K ciudades son las capitales. No hay dos ciudades con las mismas coordenadas.

Salida

El programa debe imprimir una línea con un número decimal con 5 posiciones decimales, indicando el costo total mínimo para construir las líneas de transmisión de acuerdo a las restricciones descritas.

Ejemplo de entrada 1	**Ejemplo de salida 1**
6 4 -20 10 -20 -10 20 10 20 -10 -10 0 10 0	76.56854

Ejemplo de entrada 2	**Ejemplo de salida 2**
22 9 -3 -25 0 -6 -1 -9 2 -21 -5 -19 0 -23 -2 24 -4 37 -3 33 -3 -12 2 39 3 -49 -3 -26 2 24 5 3 -4 -9 -2 -9 -4 8 3 -33 -2 31 -1 -13 0 2	95.09318

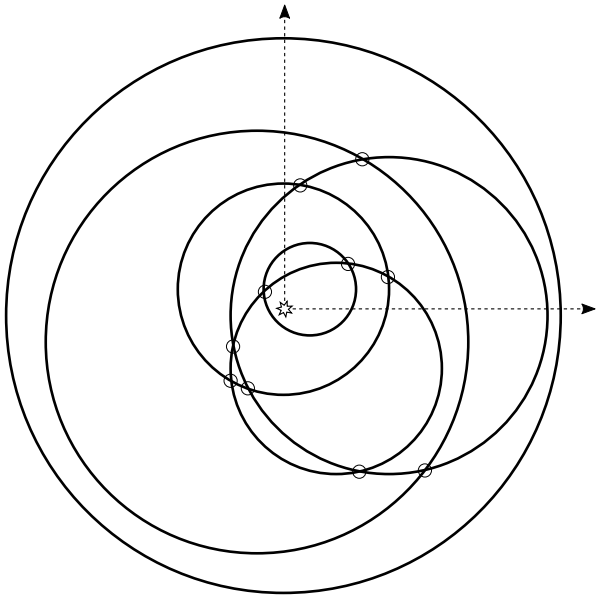
Problema K

Kepler

En este extraño sistema planetario, N planetas siguen órbitas circulares al rededor de una estrella que está posicionada en las coordenadas $(0, 0)$ del sistema. La estrella está dentro del interior de todos los círculos que definen las órbitas, sin embargo, el centro de las órbitas no es necesariamente la coordenada $(0, 0)$.

Las órbitas circulares están en posición general: si dos órbitas se interceptan, entonces, se interceptan en dos puntos distintos; además, nunca se interceptan tres órbitas en un mismo punto.

El científico Juan Kepler está interesado en probar una nueva teoría y, para eso, ha pedido tu ayuda para calcular el número de puntos de intersección entre las órbitas en caso de que ese número sea menor o igual a $2N$. En caso contrario, solo necesita saber que el número es mayor que $2N$.



Entrada

La primer línea de entrada contiene un número entero N ($2 \leq N \leq 10^5$), representando el número de órbitas. Cada una de las siguientes N líneas contienen tres números reales, con exactamente 3 dígitos decimales, X, Y y R, definiendo las coordenadas del centro y el radio de las órbitas.

Salida

Imprima una línea con un entero, representando el número de puntos de intersección entre las órbitas siempre que ese número sea menor o igual a $2N$, en caso contrario, imprima "`greater`".

Ejemplo de entrada 1	**Ejemplo de salida 1**
6 0.000 1.000 4.000 0.000 0.000 10.500 4.000 0.000 6.000 1.000 1.000 1.750 -1.000 -1.000 8.000 2.000 -2.000 4.000	10

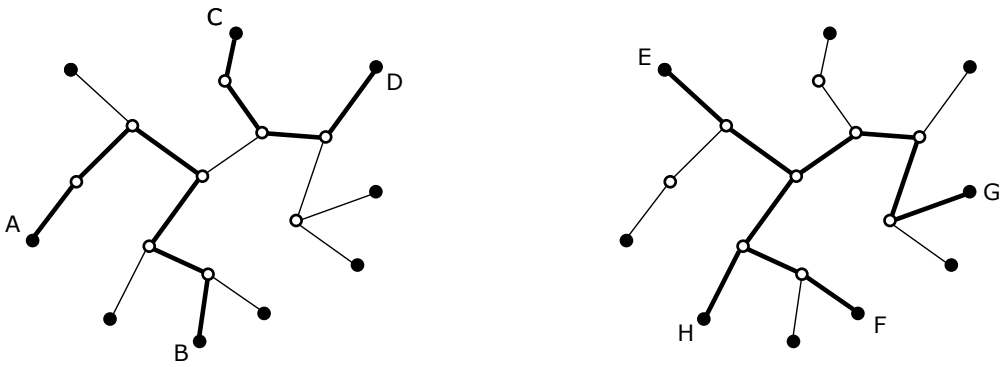
Ejemplo de entrada 2	**Ejemplo de salida 2**
4 -1.000 -1.000 3.000 1.000 -1.000 3.001 -3.004 3.003 5.002 1.000 1.000 3.005	greater

Problema L

Líneas del metro

El sistema de metro de una gran ciudad está formado por un conjunto de estaciones y por túneles que conectan algunos pares de estaciones. El sistema fue diseñado de tal forma que existe exactamente una secuencia de túneles conectando cualquier par de estaciones. Hay varias líneas de trenes que hacen viajes de ida y de vuelta entre dos estaciones terminales, transitando por el camino único entre ellas. La población se está quejando de las líneas actuales, por eso, el alcalde de la ciudad ordenó una reformulación total de las líneas. Como el sistema tiene muchas estaciones, necesitamos ayudar a los ingenieros que están intentando decidir cuales pares de estaciones pasarán a definir una línea.

La figura ilustra un sistema donde las estaciones terminales son mostradas como círculos rellenados y las estaciones no terminales como círculos vacíos. En la parte izquierda note que el par (A,B) define una línea y el par (C,D) define otra y no tienen ninguna estación en común. Sin embargo, en la parte derecha, podemos ver que los pares (E,F) y (G,H) definen dos líneas con dos estaciones en común.



Dada la descripción del sistemas de túneles y una secuencia de Q consultas compuestas por dos pares de terminales, su programa debe calcular, para cada consulta, cuántas estaciones en común tendrían esas líneas definidas por esos dos pares de estaciones.

Entrada

La primera línea de entrada contiene dos enteros N ($5 \leq N \leq 10^5$) y Q ($1 \leq Q \leq 20000$), representando respectivamente el número de estaciones y el número de consultas. Las estaciones están enumeradas de 1 hasta N. Cada una de las $N-1$ líneas siguientes contiene dos enteros distintos U y V, $1 \leq U, V \leq N$, indicando que existe un túnel entre las estaciones U y V. Cada una de las Q líneas siguiente contiene cuatro enteros distintos A, B, C y D ($1 \leq A, B, C, D \leq N$), representando una consulta: las dos líneas de tren son definidas por los pares (A, B) y (C, D).

Salida

Para cada consulta, su programa debe imprimir una línea con un entero representando cuantas estaciones en común tendrían las dos líneas de tren definidas en la consulta.

Ejemplo de entrada 1	**Ejemplo de salida 1**
10 4 1 4 4 5 3 4 3 2 7 3 6 7 7 8 10 8 8 9 6 10 2 5 1 9 5 10 9 10 2 1 5 10 2 9	0 4 0 3

Ejemplo de entrada 2	**Ejemplo de salida 2**
5 1 1 5 2 5 5 3 5 4 1 2 3 4	1

Problema M
Modificando el SAT

El problema de satisfacibilidad booleana (conocido como SAT) consiste en decidir, dada una fórmula booleana en la forma normal conjuntiva, si existe alguna asignación de valores "verdadero" o "'falso" a sus variables de modo que la fórmula entera sea verdadera.

En la forma normal conjuntiva, la fórmula se da en un formato específico. En primer lugar, las únicas operaciones lógicas utilizadas son "Y" , "O", y la negación, denotadas por $\wedge, \vee$ y $\neg$, respectivamente. Una fórmula se forma a través de la operación "Y" de diferentes partes, llamadas cláusulas, $C_1, \ldots, C_m$. De esta forma, una fórmula φ tendrá el siguiente formato:

$$\varphi = C_1 \wedge \cdots \wedge C_m.$$

Además, cada una de las cláusulas también tiene un formato específico. En particular, cada una de las cláusulas es compuesta por la operación "O" de literales que son variables o negaciones de variables, encerrados por paréntesis. Assi, $(x_1 \vee \neg x_2)$ es una cláusula válida, mientras que $(x_1 \wedge \neg x_2)$ no lo es porque usa el operador "Y". Un ejemplo de una fórmula completa es:

$$\varphi = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1) \wedge (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3).$$

Una variación al problema SAT es conocido como k-SAT, donde cada cláusula puede tener máximo k literales. La fórmula de arriba sería un ejemplo de una instancia del problema 3-SAT, pero no del 2-SAT. Note que, en todos estos problemas, para que una fórmula sea verdadera, cada una de las cláusulas debe ser verdadera y, por lo tanto, por lo menos uno de los literales (de la forma x_i o $\neg x_i$) de cada cláusula debe ser verdadero.

Una *asignación* es un modo de definir las variables como verdaderas o falsas. En este problema estamos interesados en una variación del problema 3-SAT, en el que una asignación válida debe tener exactamente 1 o exactamente 3 literales verdaderos en cada cláusula. Dada una fórmula, su tarea es decidir si existe una atribución válida, tomando en cuenta la restricción mencionada. En caso de que haya al menos una atribución válida, debe imprimir la mayor lexicográficamente. El orden lexicográfico es definido de la siguiente manera: dadas dos atribuciones diferentes, podemos compararlas buscando la variable de menor índice que difiere en las dos asignaciones; de las dos, la mayor asignación es la que da valor verdadero para dicha variable.

Entrada

La primera línea de entrada contiene dos números enteros M y N ($1 \leq M, N \leq 2000$), representando, respectivamente, el número de cláusulas y el número de variables. Las siguientes M líneas, cada una describe una cláusula (véase el ejemplo para detalles del formato). Cláusulas consecutivas son separadas por la cadena " `and`". Cada cláusula contiene un máximo de 3 literales. Las variables son denotadas por "`x`" seguido de un número entre 1 y N. No habrá dos espacios consecutivos, ni habrá espacio en blanco al final de las líneas.

El primer ejemplo muestra la fórmula φ mencionada anteriormente.

Salida

Su programa debe imprimir una única línea con N caracteres correspondientes a la asignación lexicográficamente mayor que es válida, o el texto `impossible` en caso de que no exista alguna asignación válida. El i-ésimo caracter debe ser `T` si la variable es verdadera en la asignación y `F` en caso contrario.

Ejemplo de entrada 1	**Ejemplo de salida 1**
4 3 (x1 or x2 or x3) and (not x1) and (x1 or not x2 or x3) and (x2 or not x3)	impossible

Ejemplo de entrada 2	**Ejemplo de salida 2**
5 6 (not x1) and (x1 or x2 or x4) and (x1 or x3 or x5) and (not x2 or x3 or x5) and (x2 or x3 or not x4)	FTTFFT

Ejemplo de entrada 3	**Ejemplo de salida 3**
1 1 (x1 or x1 or not x1)	F