

Problem A. Mate in 33

Input file: standard input
 Output file: standard output
 Time limit: 15 seconds
 Memory limit: 1024 megabytes

Chess (or Western Chess) is one of the world's most popular games. The bishop and knight checkmate in chess is a situation that, the checkmate of a lone king forced by a king, a bishop, and a knight.

Desprado2 is learning to play chess these days, and he finds that delivering the bishop and knight checkmate is too difficult for him. However, he thinks since computers can draw anime illustrations using AI technology, they must be able to solve a simple chess game quickly!

Formally, assume there is a **reachable** position on the chessboard, where White has a bishop, a knight and a king, while Black has a lone king, and White moves first. Your computer program should determine whether White can deliver the checkmate with the optimal play on both sides. And if there is a checkmate, you should report the number of moves for White it takes.

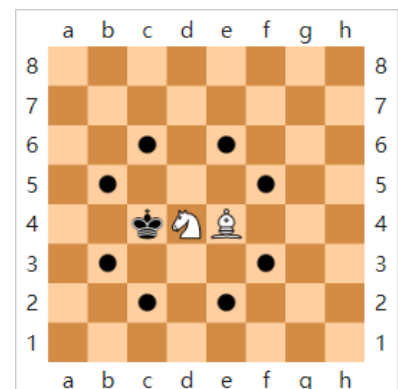
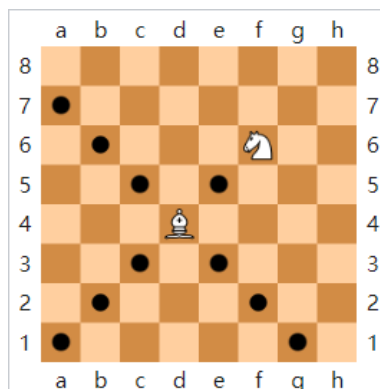
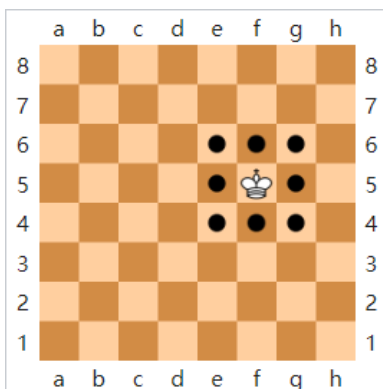
The chess rules related to this problem are as follows. **If you have already understood the basic rules of chess, you can skip the following statement and start reading the input format since we use the subset of standard rules.**

Movement

White moves first, after which players alternate turns, moving one piece per turn. A player can move a piece to either an unoccupied square or a square occupied by an opponent's piece. In the latter case, the opponent's piece is captured and removed from play.

Moving is compulsory: a player may not skip a turn. However, if one can not have a valid move in a turn, it could be a draw and we will explain it in section **End of the game**.

- The king can move one square in any of the eight directions to a square that shares at least a point with the current square. The king is considered with the highest value — attacks on the king must be immediately resolved. If a player can not counter an attack, the game ends immediately, and the attacker is declared the winner. Therefore, we have extra restrictions on moves related to such attack, see section **Check and checkmate** below.
- A bishop can move any number of squares diagonally under the constraint that it can not leap over other pieces.
- A knight moves to any of the closest squares that are not on the same rank, file, or diagonal, which forms an "L"-shape. In other words, a knight must move 1 step in one direction and 2 steps in the other. Knight can leap over other pieces without any limit on the path, and please notice that the rule is different from the rule "*Bie Ma Tui*" in Chinese chess.

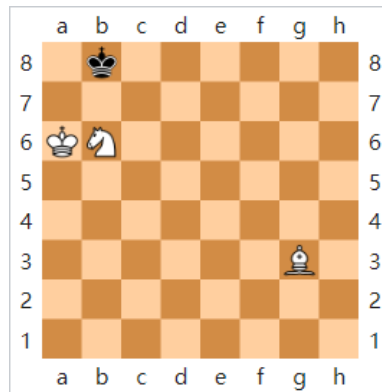


The moves of King (left), Bishop (middle) and Knight (right).

Check and checkmate

When a king is under immediate attack, it is said to be *in check*. A move in response to a check is legal only if it results in a position where the king is no longer in check.

The game's object is to *checkmate* the opponent, which occurs when the opponent's king is in check, and there is no legal way to get it out of check. It is **illegal** for a player to make a move that puts or leaves the player's king in check.



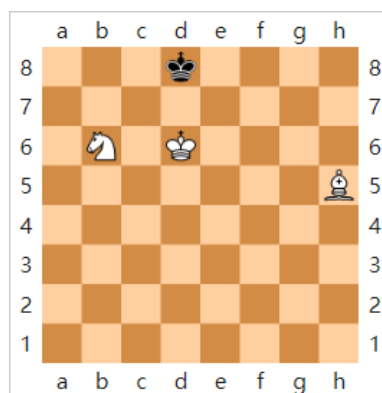
Black is in checkmate

End of the game

In this problem, the game will declare a winner only if a checkmate happens.

There are only two ways a game can end in a draw:

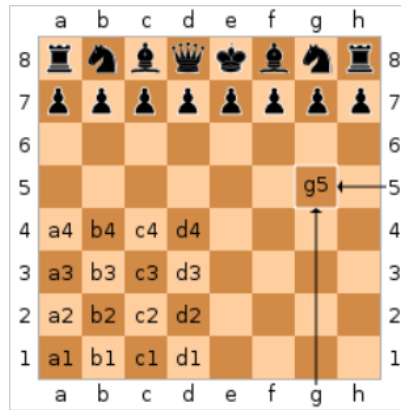
- *Stalemate*: If the player to move has no legal move, but is not in check, the position is a stalemate, and the game is drawn. Please note that this is a draw, and this rule is different from the rule “*Kun Bi*” in Chinese chess.
- *Fifty-move rule*: If during the previous 50 moves no capture has been made, the game ends in a draw. There is also an important conclusion: when there is a *dead position*, if both players keep playing, the game is bound to end in a draw by fifty-move rule. *Dead position*: a position where neither player is able to checkmate the other by any legal sequence of moves. In this problem, a dead position will happen if and only if White's bishop or knight is captured by Black's king. The game ends in a draw **immediately** when a dead position happens.



Black (to move) is not in check and has no legal move. The result is stalemate.

Notation

The standard notation system today is short-form algebraic notation. In this system, each square is uniquely identified by a set of coordinates, **a–h** for the files followed by **1–8** for the ranks.



Square names in algebraic chess notation.

Well, that's all. Refer to the sample for a better understanding.

Input

The first line contains one integer T ($1 \leq T \leq 10^5$), denoting the number of test cases.

For each test case, there is one line with four coordinates denoting the position of Black's king, White's king, White's knight, and White's bishop, respectively.

The position is guaranteed to be reachable, so notice that Black's king is guaranteed not to be under immediate attack because the White moves first.

Output

For each test case:

- If White can deliver the checkmate, print "win" (without quotes) and an integer in the following line, denoting the number of moves for White it takes if both sides play optimally.
- Otherwise, print "draw" (without quotes).

Example

standard input	standard output
5	win
a7 c7 b8 b7	1
d8 d6 b6 e8	draw
b1 d4 a2 c1	draw
b1 d1 a2 c1	win
c8 a8 h2 e8	6
	win
	33

Note

We still use short-form algebraic notation to describe a move. The usual format is:

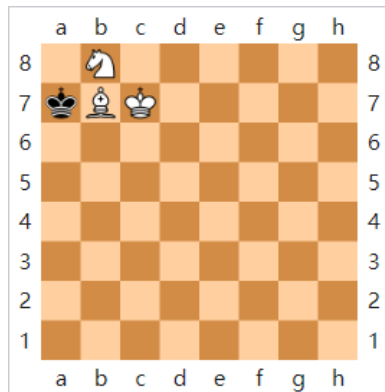
initial of the piece moved – file of destination square – rank of destination square

The pieces are identified by their initials of English names. In this problem, these are K (king), B (bishop), and N (knight; N is used to avoid confusion with king). For example, Kg5 means "king moves to g-file, 5-th rank (that is, to square g5)". If the piece makes a capture, "x" is inserted before the destination square. Thus Kxg5 means "king captures on g5". A move that places the opponent's king in check should add the

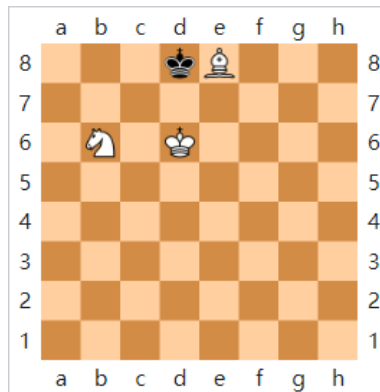
notation “+”. And if it is checkmate, “#” is used instead. For example, **Be3+** means “bishop moves to e3 with check”, and **Na6#** means “knight moves to a6 with checkmate”.

The positions of the sample input are shown below:

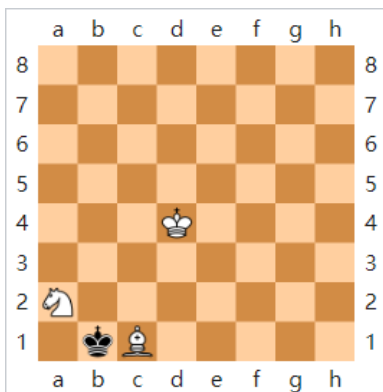
A.



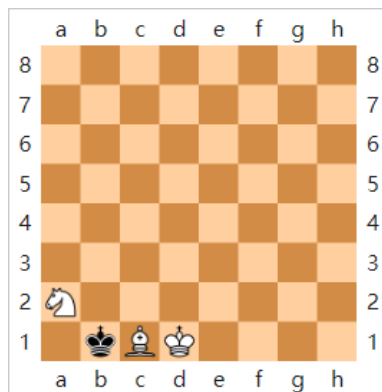
B.



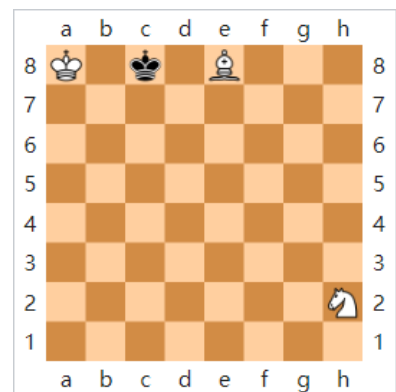
C.



D.



E.



- In position A, White can checkmate in one move:
 1. **Nc6#**
- In position B, White's bishop is under attack. If White just let Black capture the bishop, the game will end in a dead position. The only way for White to save their bishop is to move it, such as:
 1. **Bh5**

However, wherever White moves the bishop, the position will be a stalemate. Therefore White can't deliver the checkmate.
- In position C, both White's bishop and knight are under attack, but White can just save one piece. So Black can capture one of the two pieces, and then the game ends in a dead position. For example:
 1. **Bd2 K×a2**

or

 1. **Nc3+ K×c1**
- In position D, White's bishop is protected by White's king, so White just need to move the knight. We can prove that White can checkmate in 6 moves, and the optimal first move is
 1. **Nb4**
- In position E, White checkmate in 33 moves with optimal play on both sides, which is the longest distance to checkmate for these positions.

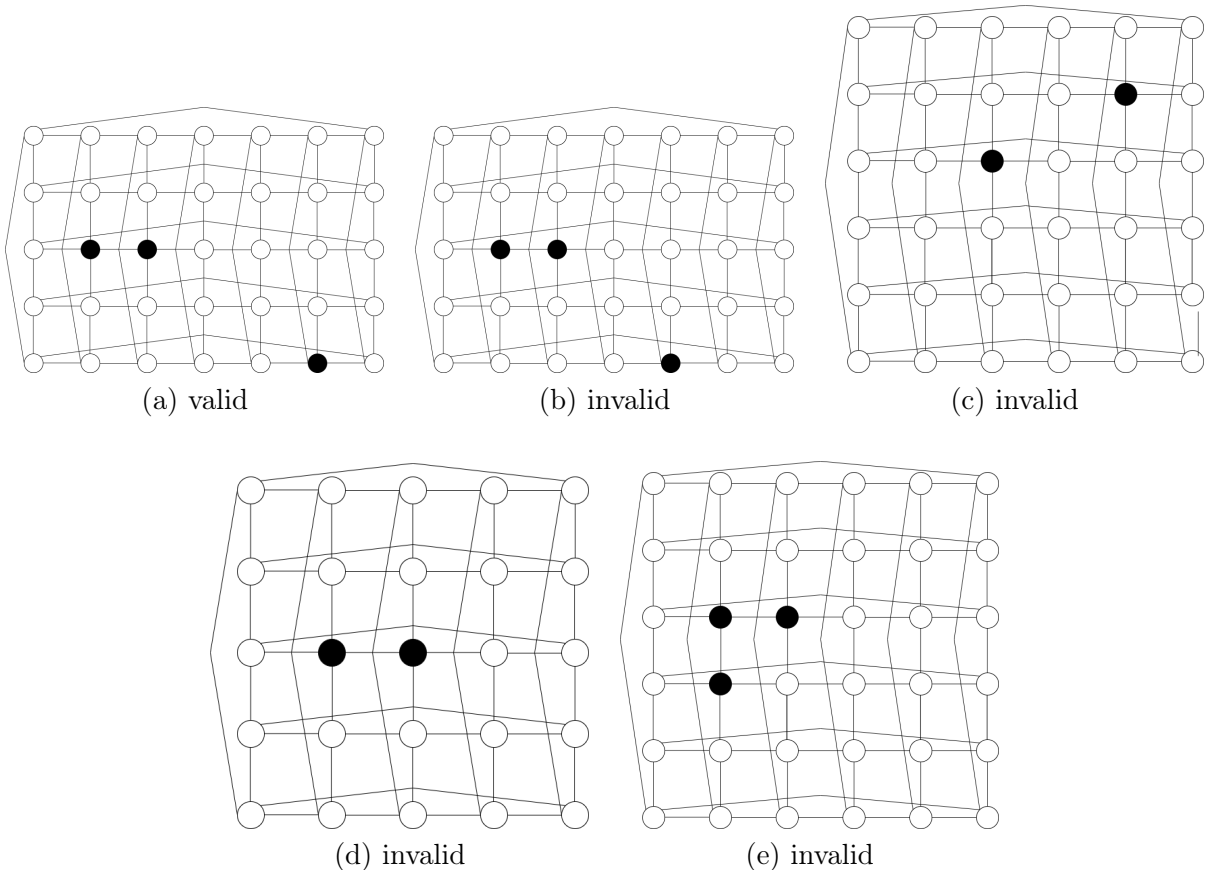
Problem B. Grid World

Input file: standard input
 Output file: standard output
 Time limit: 8 seconds
 Memory limit: 1024 megabytes

An $r \times c$ **grid graph** is a graph whose vertices correspond to the points in the plane with integer coordinates, x -coordinates being in the range $1, \dots, r$, y -coordinates being in the range $1, \dots, c$, and two vertices are connected by an edge whenever the corresponding points are at distance 1. An $r \times c$ **cyclic grid graph** is a graph formed by connecting the first row with the last row and the first column with the last column on the basis of the $r \times c$ grid graph. Formally, in a $r \times c$ cyclic grid graph, (x, y) and $(x, y \bmod c + 1)$ is adjacent, and (x, y) and $(x \bmod r + 1, y)$ is adjacent.

Desprado2 constructed an $r \times c$ grid graph, but unfortunately, there are some broken vertices into the graph. This graph has some properties:

1. $r, c \geq 5$.
2. Each unbroken vertex is adjacent to at least three unbroken vertices.
3. Every row and column has at least four unbroken vertices.
4. For any broken vertices u and v , if they are not in the same connected component, then the shortest distance between u and v is at least 5.



Some examples are shown above, where white denotes unbroken vertices while black denotes broken vertices.

- The graph in (a) is valid.

- The graph in (b) is invalid, because the shortest distance between broken vertices $(3, 3)$ and $(5, 5)$ is only 4, which breaks property 4.
- The graph in (c) is invalid, because the shortest distance between two broken vertices is 3, which breaks property 4.
- The graph in (d) is invalid, because the row 3 only contains 3 unbroken vertices, which breaks property 3.
- The graph in (e) is invalid, because the unbroken vertex $(4, 3)$ is adjacent to only two unbroken vertices, which breaks property 2.

After removing all the broken vertices in the graph and numbering the unbroken vertices **arbitrarily**, he got an undirected graph of n vertices and m edges. There is an edge between u and v in the undirected graph if and only if u and v are adjacent in the original cyclic grid graph. Unfortunately again, after some days, he only remembered the undirected graph and forgot the coordinates of vertices in the original cyclic grid graph.

Please help Desprado2 to restore the original cyclic grid graph.

Input

The first line contains four integers n , m , r and c ($n \leq 10^5$, $m \leq 2n$, $n \leq r \cdot c \leq 10^6$), denoting the number of vertices and edges in the undirected graph, and the number of rows and columns in the original cyclic grid graph.

Then follows m lines. Each line contains two integers u and v ($1 \leq u, v \leq n$, $u \neq v$), denoting an edge in the undirected graph.

It is guaranteed that the undirected graph is converted from a valid cyclic grid graph satisfying all the properties in problem description.

Output

Print an $r \times c$ matrix denoting the original cyclic grid graph. The number $a_{i,j}$ denotes the vertex $a_{i,j}$ in the undirected graph locates at the i -th row and j -th column in the original cyclic grid graph. If vertex (i, j) in original cyclic grid graph is broken, then $a_{i,j} = -1$.

Note that there are multiple answers since flipping, shifting and sometimes transposing the matrix do not affect the correctness. You are required to print the following one:

Let

$$b_{i,j} = \begin{cases} a_{i,j} & \text{if, } a_{i,j} > 0 \\ r \cdot c + 1 & \text{if, } a_{i,j} = -1 \end{cases}$$

You should print the answer such that the sequence $(b_{1,1}, b_{1,2}, \dots, b_{1,c}, b_{2,1}, \dots, b_{r,c})$ is lexicographically smallest among all answers.

Please note that if your programming language is C++ and you use `std::cout`, please use “\n” instead of `std::endl` to insert a new line, because `std::endl` will flush the output buffer so it may be too slow.

Example

standard input	standard output
24 46 5 5	1 16 9 10 -1
24 23	19 21 2 5 20
24 4	15 17 24 23 13
4 7	12 8 4 7 3
4 14	22 11 14 18 6
14 18	
14 9	
9 10	
9 2	
2 5	
2 24	
23 13	
23 7	
7 3	
7 18	
18 6	
18 10	
10 5	
5 20	
5 23	
13 15	
13 3	
3 12	
3 6	
6 22	
20 19	
20 13	
15 17	
15 12	
12 8	
12 22	
22 11	
22 1	
1 16	
1 19	
19 21	
19 15	
17 24	
17 8	
8 4	
8 11	
11 14	
11 16	
16 9	
16 21	
21 2	
21 17	

Problem C. Multiplication

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 1024 megabytes

Xiao fan is a third grader in Springfield Flower Kindergarten, and he learned multiplication of multi-digit numbers by one-digit numbers last week. Clever as he was, he soon found some unusual numbers – multiplying these numbers by a specific one-digit integer k ($2 \leq k \leq 9$) results in movement of some trailing digits to the front. For example, the following picture shows that 142857 is a 3-shifting number.

$$\underline{142857} \times 3 = \underline{428571}$$

Xiao fan wants to find the largest k -shifting number less than or equal to n . However, this task is too difficult for him, so he turns to you for help!

Input

The first and second line contains two integers n and k ($0 \leq n < 10^{100}$, $2 \leq k \leq 9$) respectively.

Output

Print a single integer – the largest k -shifting number less than or equal to n . The answer always exists since 0 is always an k -shifting number for any k .

Examples

standard input	standard output
142857 3	142857
19260817 3	307692
1145141919810 8	1139240506329
1145141919810 7	0

Problem D. Agenj

Input file: `standard input`
 Output file: `standard output`
 Time limit: 1 second
 Memory limit: 1024 megabytes

Jenga is a game of physical skill created by British board game designer and author Leslie Scott. Jenga is played with 54 wooden blocks. Players take turns removing one block from any layer below the highest completed one and placing it horizontally atop the tower, perpendicular to any blocks on which it is to rest. The game ends when any portion of the tower collapses, caused by either the removal of a block or its new placement.



Alice and Bob are playing Agnej, a Jenga-like building blocks game. At the beginning, Alice build a block tower of $n + 1$ layers. The blocks within each layer are oriented in the same direction, with their long sides touching, and are perpendicular to the ones in the layer immediately below. Each block is m times as long as it is wide, so there are exactly m positions to place blocks in each layer, numbered from 1 to m . Alice and Bob take turns removing one block from any layer **except the top layer**, and the removed block **won't** be placed back atop the tower. The player who causes the tower to collapse loses the game, while the other player wins.

For simplicity, we can assume that the blocks from layer 1 (from top to bottom) to layer $i - 1$ are light enough for the blocks of layer i , and both Alice and Bob are careful enough when removing blocks. Therefore, the tower will collapse **if and only if** after a player removes a block of some layer l , there are no blocks on position 1 through position $\lceil \frac{m}{2} \rceil$ in layer l , or there are no blocks on position $\lfloor \frac{m}{2} \rfloor + 1$ through position m in layer l .

You are given the initial shape of the tower. Assume both Alice and Bob plays optimally and Alice plays first, please determine who will win the game.

Input

The first line contains two integer n and m ($1 \leq n, m \leq 10^5$, $1 \leq n \times m \leq 10^5$).

Then follows n lines. The i -th line contains a 0-1 string of length m , denoting the shape of layer $i + 1$ (from top to bottom). The j -th bit of the string denotes whether there is a block at j -th position (0 denotes no while 1 denotes yes).

Please note that the top layer does not affect the answer, so it does not appear in input. The input just shows the shape from layer 2 to layer $n + 1$. You can assume the top layer is not empty.

It is guaranteed that the tower won't collapse before the game starts.

Output

Print “Alice” if Alice will win the game, “Bob” otherwise (both without quotes).

Examples

standard input	standard output
4 3 111 101 110 010	Alice
3 4 1101 1011 1111	Bob
3 5 00101 00111 10100	Bob

Problem E. Geometry Problem

Input file: **standard input**
 Output file: **standard output**
 Time limit: 6 seconds
 Memory limit: 1024 megabytes

You are given n **distinct** circles in the 2-dimensional plane. Please find a minimum square to enclose all the circles.

Input

The first line contains a single integer n ($1 \leq n \leq 5 \times 10^4$), denoting the number of circles.

Then follows n lines, each line contains three integers x, y, r ($-10^6 \leq x, y \leq 10^6, 1 \leq r \leq 10^6$), denoting a circle with its center at (x, y) and radius r .

Output

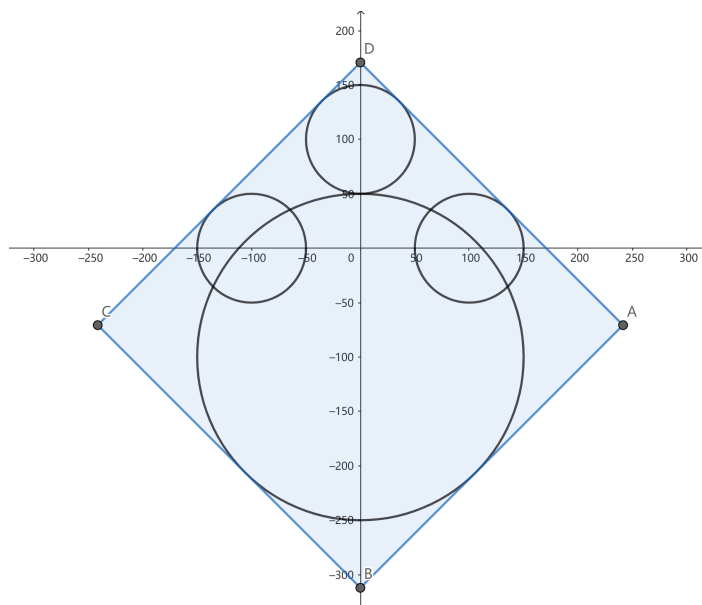
Print a single real number - the minimum side length of the square. Your answer will be accepted if the absolute or relative error to the jury's answer is less than or equal to 10^{-6} .

Examples

standard input	standard output
4 0 100 50 100 0 50 -100 0 50 0 -100 150	341.4213562373
4 0 0 1 0 5 1 2 0 1 2 5 1	6.9497474683

Note

The first example is shown as follows:



Problem F. IUPC

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	1024 megabytes

The International Unfair Programming Contest (IUPC) is a competition for cheating skills. In this competition, participants are required to solve as many problems as possible by cheating without drawing *suspicion*. As is known to all, Nan Xiang Institute of Science and Technology (NXIST) is one of the best cheaters in IUPC. They shot to fame by winning the gold medal in the IUPC Ningxia Railway Station, Yinchuang Site held in May 2020, where they successfully solved 6 problems.



You are the leader of IUPC team in NXIST, and your team is going to participate in IUPC Silk-Road Contest next week. There are n problems in this contest, and participants have t minutes to solve them. As a master cheater, you already know how to steal solutions submitted by others, so all your team need is to submit the stolen solutions without drawing *suspicion* during the contest. Precisely, if problem i is first solved by other team at the x_i -th minute of the contest, your team will steal the solution **immediately** and can submit it **exactly once** at any time before the contest ends. However, to avoid *suspicion*, you set the following rules for your team:

- Your team can submit at most one solution in each minute.
- For every k -minute time interval $[x, x + k - 1]$ ($x \geq 1$, $x + k - 1 \leq t$) in the contest, your team can submit at most m solutions. Here k and m are given numbers.

Now you can predict the exact moment when each problem will be solved by other teams, that is, you know when your team can get the solution to each problem. Please calculate the number of different plans for your team to solve **all** problems without drawing *suspicion*. Formally, a plan can be denoted as a sequence a_1, a_2, \dots, a_t , where $a_i = p$ if your team submits the solution to problem p in the i -th minute, or $a_i = 0$ if your team does nothing in the i -th minute. Two plans are different if the sequences are different.

Input

The first line contains four integers n , t , k and m ($1 \leq n \leq 13$, $1 \leq t \leq 300$, $1 \leq k \leq 10$, $1 \leq m \leq k$).

The second line contains n integers x_1, x_2, \dots, x_n ($1 \leq x_i \leq t$), where x_i denotes the moment when the problem i is first solved by others.

Output







Print an integer - the number of different plans for your team to solve all problems. Since the answer may be very large, you are required to print the value modulo $10^9 + 7$.

Examples

standard input	standard output
4 10 3 2 4 4 5 8	156
6 300 10 1 12 263 44 108 118 291	7759211

Note

For the second example, one of the valid plans can be denoted as follows:

 00:12	 04:23	 00:44	 01:48	 01:58	 04:51
---	---	---	---	---	---

Please note that all people, locations, and events mentioned above are entirely fictional. Any resemblance to actual people, places, or events is purely coincidental.

Problem G. Palindrome Subsequence

Input file: standard input
 Output file: standard output
 Time limit: 7 seconds
 Memory limit: 1024 megabytes

You are given a string s of length n , and there are some fixed positions in the string. You need to find a palindrome subsequence of s which contains all fixed positions and has the largest lexicographical order.

Formally, let the string $s = c_1c_2\dots c_n$, and the fixed positions $F = \{a_1, a_2, \dots, a_m\}$ ($1 \leq a_i \leq n$). You should find a sequence $P = (p_1, p_2, \dots, p_t)$ ($1 \leq p_i \leq n$, $p_i < p_{i+1}$) satisfying:

- $\forall 1 \leq i \leq m$, a_i appears in P .
- $c_{p_1}c_{p_2}\dots c_{p_t}$ is a palindrome and its lexicographical order is the largest.

A string is a palindrome if it reads the same backwards as forwards.

Input

The first line contains a single integer T ($1 \leq T \leq 1000$) - the number of test cases.

For each test case, the first line contains an integer n ($1 \leq n \leq 1000$), denoting the length of the string.

The second line contains a string s of length n , and s consists of first 10 lowercase letters ('a' - 'j').

The third line contains n integers a_1, a_2, \dots, a_n ($a_i \in \{0, 1\}$). $a_i = 1$ denotes i is a fixed position.

It is guaranteed that the sum of n over all test cases does not exceed 1000.

Output

For each test case, print a string - the palindrome subsequence that has the largest lexicographical order. If the answer doesn't exist, print "-1" (without quotes) instead.

Example

standard input	standard output
4	dddd
10	bcb
abdcadcaddb	-1
0 0 0 0 0 0 0 0 0 0	hehheh
5	
abcbb	
0 0 0 1 0	
5	
abcbb	
1 1 1 1 1	
11	
hehhehahhhh	
0 1 0 0 0 0 0 0 1 0 0	

Problem H. Differential Equation

Input file: **standard input**
 Output file: **standard output**
 Time limit: 3 seconds
 Memory limit: 512 megabytes

AntiLeaf is a master on solving differential equations and polynomial techniques, thus he has came out the following problem for you.

There is a series of functions $F_k(x)$, where k is a natural number, and x is the independent variable. The value of $F_k(x)$ is defined as follows:

- $F_0(x) = x$;
- For any non-zero k , $F_k(x) = (x^2 - 1) F'_{k-1}(x)$, where $F'_{k-1}(x)$ denotes the derivative of $F_{k-1}(x)$.

Given a non-negative integer n and a non-negative integer x_0 , please calculate the value of $F_n(x_0)$.

Since the answer may be very large, you are required to output the answer modulo 998,244,353. It can be proved that the answer is always an integer.

Input

The only line of input consists of two integers n ($0 \leq n \leq 2 \times 10^5$) and x_0 ($0 \leq x_0 \leq 998,244,352$).

Output

Print one non-negative integer in one line, representing the value of $F_n(x_0)$ modulo 998,244,353.

Examples

standard input	standard output
3 2	66
100 6	397318361

Note

$$F_0(x) = x$$

$$F_1(x) = x^2 - 1$$

$$F_2(x) = 2x^3 - 2x$$

$$F_3(x) = 6x^4 - 8x^2 + 2$$

Therefore $F_3(2) = 6 \times 16 - 8 \times 4 + 2 = 66$.

Problem I. Hello

Input file: **standard input**
 Output file: **standard output**
 Time limit: 15 seconds
 Memory limit: 1024 megabytes

In a distant land, there is a small peaceful village, which can be represented as a 2-dimensional plane.

There are n houses in the village, the i -th of which locates at the coordinate (x_i, y_i) . There are also $n - 1$ **straight** bidirectional roads, each connecting two houses. In other words, each house can be treated as a point on the 2-d plane, and each road can be treated as a line segment with two houses as its endpoints. In this village, every two houses can reach to each other through the roads, and no two roads intersect except at their endpoints.

Today m villagers plan to go for a walk. The i -th villager will start walking from the coordinate (sx_i, sy_i) , and end at the coordinate (ex_i, ey_i) . It is guaranteed that both the starting point and ending point are either at a certain house or on a certain road in the village.

During their walk, when two villagers meet, they will say hello to each other. However, since their starting time may differ, it is difficult to determine who will greet each other. So let us focus on a simpler task: for each villager, figure out how many other villagers' routes intersect with his/her own.

The route of a villager is defined as the unique simple path connecting his/her starting and ending points via the roads. Two routes are defined to intersect, if and only if they share at least one common point.

Input

The first line of input consists of two positive integers n, m ($1 \leq n \leq 10^5$, $1 \leq m \leq 10^5$), representing the number of houses and the number of villagers respectively.

Then follows n lines. The i -th line consist of two integers x_i, y_i ($|x_i|, |y_i| \leq 10^6$), denoting the coordinate of the i -th house. It is guaranteed that all the coordinates of houses are distinct.

Then follows $n - 1$ lines. The i -th line consist of two positive integers u_i and v_i ($1 \leq u_i, v_i \leq n$, $u_i \neq v_i$), indicating there is a road connecting the u_i -th house and the v_i -th house. It is guaranteed that the roads meet with the restrictions mentioned above.

Then follows m lines. The i -th line consist of eight integers $p_i^{sx}, q_i^{sx}, p_i^{sy}, q_i^{sy}, p_i^{ex}, q_i^{ex}, p_i^{ey}, q_i^{ey}$ ($|p_i^*| \leq 10^{12}$, $0 < q_i^* \leq 10^6$), denoting the route of the i -th villager – the starting point is $(\frac{p_i^{sx}}{q_i^{sx}}, \frac{p_i^{sy}}{q_i^{sy}})$, and the ending point is $(\frac{p_i^{ex}}{q_i^{ex}}, \frac{p_i^{ey}}{q_i^{ey}})$. It is guaranteed that each pair of numerator and denominator is co-prime, and the starting point and ending point won't coincide.

Output

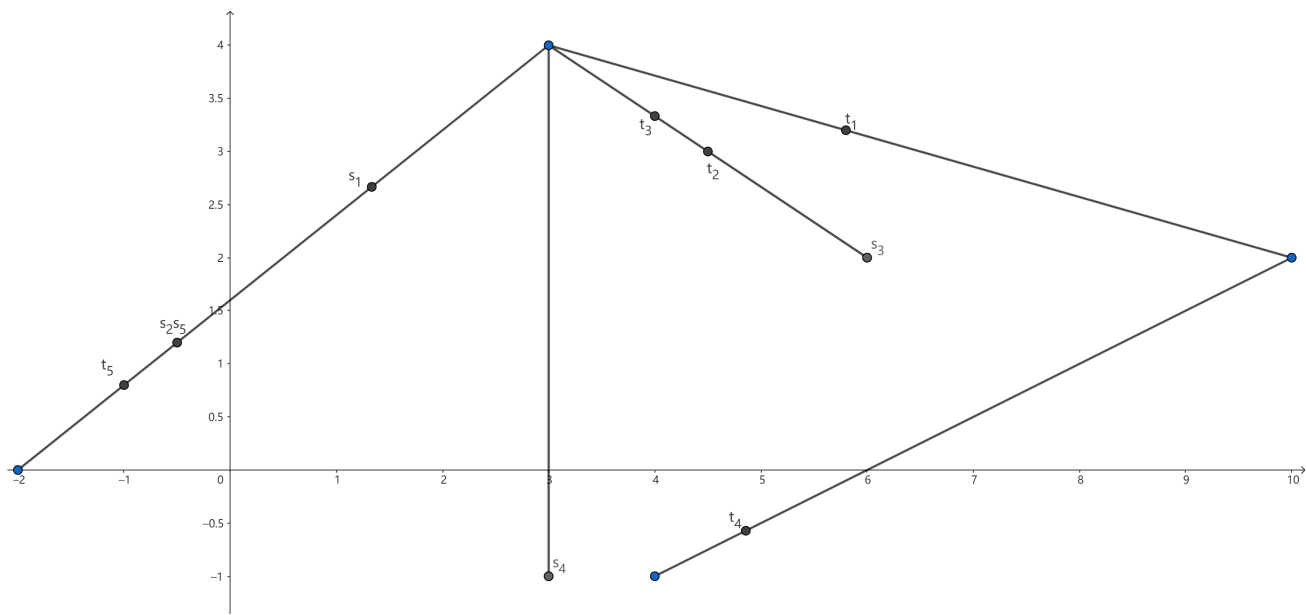
Print m non-negative integers in a single line, each separated by a single space. The i -th integer represents the number of other villagers whose route intersects with that of the i -th villager.

Example

standard input	standard output
6 5	2 4 1 2 1
-2 0	
3 4	
10 2	
4 -1	
6 2	
3 -1	
1 2	
2 3	
3 4	
2 5	
2 6	
4 3 8 3 29 5 16 5	
-1 2 6 5 9 2 3 1	
6 1 2 1 4 1 10 3	
3 1 -1 1 34 7 -4 7	
-1 2 6 5 -1 1 4 5	

Note

The example can be shown as follows:



Problem J. Pumping

Input file: **standard input**
Output file: **standard output**
Time limit: 3 seconds
Memory limit: 1024 megabytes

Prerequisite: Deterministic Finite Automaton, DFA

A deterministic finite automaton (DFA) M is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, consisting of:

- a finite set of states Q
- a finite set of input symbols called the alphabet Σ
- a transition function $\delta : Q \times \Sigma \rightarrow Q$
- a start state $q_0 \in Q$
- a set of accept states $F \subseteq Q$ (F is not empty)

Let $s = c_1 c_2 \cdots c_n$ be a string over the alphabet Σ . The automaton M **accepts** the string s if a sequence of states, r_0, r_1, \cdots, r_n , exists in Q with the following conditions:

1. $r_0 = q_0$
2. $r_{i+1} = \delta(r_i, c_{i+1})$, for $i = 0, \cdots, n-1$
3. $r_n \in F$.

In words, the first condition says that the machine starts in the start state q_0 . The second condition says that given each character of string s , the machine will transition from state to state according to the transition function δ . The last condition says that the machine **accepts** s if the last input of s causes the machine to halt in one of the accepting states. Otherwise, it is said that the automaton **rejects** the string - there is two cases where M rejects the string:

- At some intermediate state r_i , there doesn't exist a transition of state r_i and character c_{i+1} , that is, $(r_i, c_{i+1}) \notin \text{dom}(\delta)$.
- The final state is not an accept state, that is, $r_n \notin F$.

The set of strings that M accepts is called the language recognized by M and this language is denoted by $L(M)$.

Problem

You are given a deterministic finite automaton M . We say a string $s \in L(M)$ can be *pumped* if the string can be divided into three parts $s = xyz$ satisfying:

- $|y| > 0$
- for each $i \geq 0$, $xy^iz \in L(M)$. Here y^i means repeating string y for i times.

Please find a string in $L(M)$ that can be pumped or report it is impossible.

Input

The first line contains three integers n, m, k ($1 \leq n \leq 5 \times 10^5$, $0 \leq m \leq 10^6$, $1 \leq k \leq n$), denoting the number of states, the number of transitions and the number of accept states in the DFA. We number the states in the DFA from 1 to n , and let the start state be state 1.

Then follows a line with k distinct integers a_1, a_2, \dots, a_k ($1 \leq a_i \leq n$), denoting the accept states.

Then follows m lines, and each line contains three integers u, v, c ($1 \leq u, v \leq n$, and c is a lowercase character), denoting a transition $\delta(u, c) = v$ in the DFA. It is guaranteed that there is at most one transition for each state u and each character c .

Note that the input data size can be large, so if you use `std::cin` in C++, you may need to call `std::ios::sync_with_stdio(false); cin.tie(nullptr);` to disable stream synchronization.

Output

Print a string of length up to $3n$ that can be pumped. If there are multiple answers, print any. If there is no answer, print -1 instead.

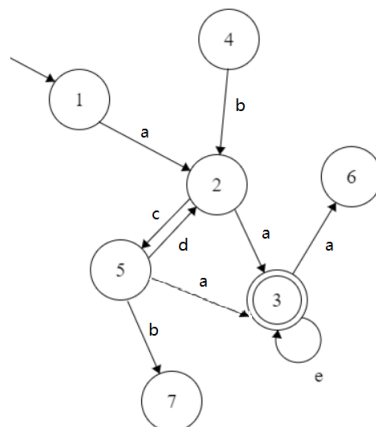
It can be proved that if answer exists, then at least one answer has length $3n$ or less.

Examples

standard input	standard output
<pre> 7 9 1 3 1 2 a 2 5 c 5 2 d 5 7 b 5 3 a 4 2 b 2 3 a 3 3 e 3 6 a </pre>	acdca
<pre> 2 3 1 1 2 2 a 1 2 b 1 2 a </pre>	-1

Note

The DFA in the first example is shown in the figure below.



The sample output “acdca” can be divided into three parts $x = ac$, $y = dc$ and $z = a$ to satisfy above requirements.

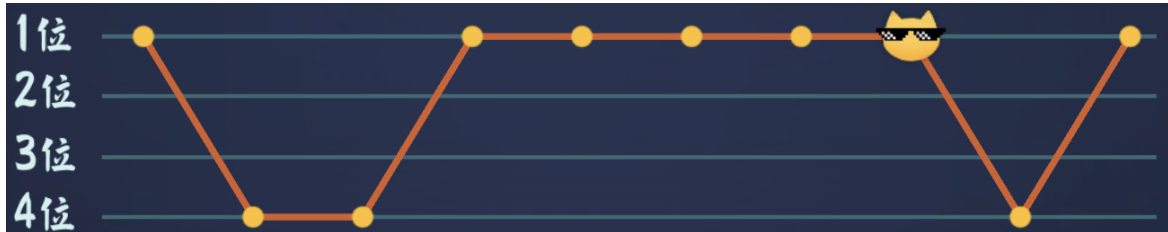
There are also some other answers, for example:

- $s = acda$, where $x = a$, $y = cd$ and $z = a$.
- $s = acaee$, where $x = aca$, $y = ee$, and z is empty.

Problem K. First Last

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

You are playing a game with n players, where you always find yourself ranking either first or last.



A weird game log.

This is quite strange, right? One possible explanation is that the game system is rigged; another possible explanation is that this player is risk-taking and would seize every opportunity to reach the top spot at the expense of lower expected scores.

Now, suppose you are an average player who ranks uniformly at random each time independently. How likely is it that you are always the “outstanding” one, either first or last, in m consecutive games?

To be more precise, in an n player game, an average player takes rank i ($1 \leq i \leq n$) with probability $1/n$. The first means the rank 1, and the last means the rank n .

Input

Input contains two integers, n, m ($1 \leq n, m \leq 20$), the number of players in each game and the number of games to be considered.

Output

Print a real number presented by decimal format, denoting the possibility.

Your answer will be accepted if the absolute or relative error is at most 10^{-9} .

Examples

standard input	standard output
4 10	0.000976562500000
2 20	1.000000000000000
3 3	0.296296296296296

Problem L. Grayscale Confusion

Input file: **standard input**
Output file: **standard output**
Time limit: 1 second
Memory limit: 1024 megabytes

Sometimes, you may want to print a color image in grayscale mode, either because you want to save ink or your printer can only do so. However, when you do that, you may encounter a problem: some colors that are distinct in the color image may become indistinguishable in the grayscale image. This is the information loss caused by the grayscale conversion.



Two colors that look the same when printed in grayscale mode: (165, 160, 210) and (220, 160, 60), with the same RGB Luminance 167 calculated by $0.3R + 0.59G + 0.11B$.

Of course, a good conversion from RGB (Red, Green, Blue) to Grayscale is not unique. Here, we define a property that a good conversion must satisfy: if a color (r, g, b) is strictly less than (r', g', b') : $r < r', g < g', b < b'$, then the conversion f must hold that $f(r, g, b) < f(r', g', b')$. It means that the strict partial order on color space should be preserved.

You are given an RGB image and you want to convert it to a grayscale image. However, you also want to confuse the viewers by intentionally mapping two specified colors to the same grayscale value, while keeping the order of the colors. Your task is to find a good conversion that confuses two given colors from n given colors.

Input

The first line of input contains an integer n ($2 \leq n \leq 1\,000$), representing the number of distinct colors in the RGB image.

The next n lines of input each contain three integers r_i, g_i, b_i ($0 \leq r_i, g_i, b_i \leq 255$), representing the red, green and blue components of a color c_i .

The two colors that should be confused is the first two colors, c_1 and c_2 .

Output

If it's impossible to do so, print -1 in one line.

Otherwise, output n lines, each containing an **integer** b_i ($0 \leq b_i \leq 255$), representing the grayscale value of the corresponding color.

Examples

standard input	standard output
3 0 0 0 2 2 2 1 1 1	-1
3 1 0 0 0 1 0 0 0 1	0 0 1
3 0 0 0 0 1 2 255 255 255	0 0 1


Problem M. Fair Equation


Input file: standard input
 Output file: standard output
 Time limit: 1 second
 Memory limit: 1024 megabytes


Equations are not solvable in a general fashion. Here's an example.

- What is the solution of    if they represent some positive integers?

$$\frac{\text{apple}}{\text{orange} + \text{banana}} + \frac{\text{banana}}{\text{apple} + \text{orange}} + \frac{\text{orange}}{\text{banana} + \text{apple}} = 4$$

 = 154476802108746166441951315019919837485664325669565431700026634898253202035277999

Answer:  = 36875131794129999827197811565225474825492979968971970996283137471637224634055579

 = 4373612677928697257861252602371390152816537558161613618621437993378423467772036

95% of people can't solve this! True dude.

It's not fair to solve unsolvable problems in the contest, so we would like you to solve a easier one. It's decidable in polynomial time, so it's fair.

You are given an equation of the form $A + B = C$, where A , B and C are positive integers. However, the equation may not be true at the moment. You can insert a single digit (0-9) anywhere in the equation, before or after any existing digit, to make it true.

For example, if the equation is $12 + 34 = 146$, you can insert 1 to 12 (either $\bar{1}12$ or $1\bar{1}2$) to make it $112 + 34 = 146$.

Your task is to determine if there exists a way to insert a digit in the equation to make it true, or it is already true.

Input

The first and only line of input contains an equation of the form $A + B = C$, where A , B and C are positive integers with no leading zeros. The input satisfies $1 \leq A, B, C \leq 10^6$.

It's guaranteed that there are exactly one space before and after "+" and "=".

Output

If there exists a way to insert a digit in the equation to make it true, or it is already true, output **Yes** and the modified equation on two separate lines. Please note that the equation should be presented like input format, i.e., no leading zeros and exactly one space before and after "+" and "=".

Otherwise, output **No** on a single line.

Examples

standard input	standard output
12 + 34 = 146	Yes 112 + 34 = 146
12 + 34 = 56	No