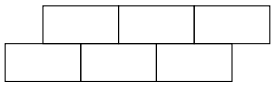
Problem A. Another Brick in the Wall

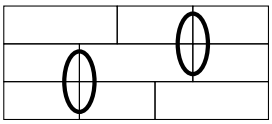
Time limit: 2 seconds
Memory limit: 1024 megabytes

Alice likes building toy walls. She has a lot of 1×2 bricks and a limited supply of 1×3 bricks. Both types of bricks have a height of 1 and can not be rotated.

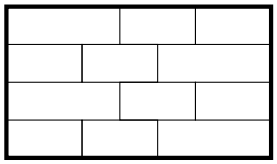
Alice is going to build a one unit thick wall of length l and height h out of these bricks. A wall is *solid* if there are no seams directly above another seam.



Good seam placement



Bad seam placement



Solid 7×4 wall

Help Alice determine the minimum number of 1×3 bricks required to build a solid wall of length l and height h.

Input

The only line contains two integers l and h, denoting the length and the height of the wall ($5 \le l \le 1000$; $2 \le h \le 1000$).

Output

Print the minimum number of 1×3 bricks required to build a solid $l \times h$ wall.

It can be shown that it is always possible to build a solid wall of length l and height h.

Example

standard input	standard output
7 4	4

Problem B. Brick in the Wall, Part 2

Time limit:	5 seconds
Memory limit:	1024 megabytes

Barrett has discovered an ancient maze under his house. It has the shape of an $n \times m$ grid, where some cells are empty, while others are blocked. It is possible to walk from one empty cell to another if they share a side. Two of the empty cells are an entrance and an exit, and it is possible to reach one from the other by walking through empty cells.

Barrett wants to isolate his house by building a wall inside the maze, blocking some cells to make the exit unreachable from the entrance. The wall should be straight and oriented either vertically or horizontally. Specifically, a wall of length k will block a consecutive row or column of exactly k cells. The wall may not contain the entrance, the exit, or any already blocked cells.

Help Barrett determine the minimum possible length of the wall.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^5$). The description of the test cases follows.

The first line of each test case contains two integers n and m, denoting the height and the width of the maze ($2 \le n, m \le 1000$).

The i-th of the following n lines contains m characters and describes the i-th row of the maze, where:

- '`.`' denotes an empty cell;
- '`#`' denotes a blocked cell;
- '`s`' denotes an entrance cell;
- '`f`' denotes an exit cell.

The maze contains exactly one entrance cell and exactly one exit cell, and it is possible to reach one from the other by walking through empty cells.

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed 10^6.

Output

For each test case, print the minimum length of the wall required to make the exit unreachable from the entrance.

If it is impossible to build such a wall, print -1 instead.

Example

standard input	standard output
3 3 3 s.# ... #.f 6 7 ..#.#.. s..#..# ....#f. #..#... #..... #.....# 2 2 s. .f	1 2 -1

Problem C. Capybara Cozy Carnival

Time limit: 4 seconds
Memory limit: 1024 megabytes

Chilling capybaras celebrate Capybara Cozy Carnival. Chairman capybara cuts convex cake. Cake contains n colorful corners. Countless colors comprise k choices. Creating m continuous crossing-free corner-to-corner cuts, chairman cuts cake chunks, catering $m + 1$ comrades. Curiously, consecutive cake chunks corners contain contrasting colors.

Calculate cake corners color combinations, considering cuts conditions.

In other words, you are given a cake in the shape of a regular n-sided polygon and m non-intersecting diagonal cuts, which divide it into $m + 1$ slices.

Calculate the number of ways to color each corner of the original cake with one of the k colors, such that no two neighboring corners of the resulting slices have the same color. Two corners are considered neighboring if they are either consecutive in the original cake, or they are the endpoints of the same cut. It is not necessary to use all the colors. As the number of ways might be large, find it modulo 998 244 353.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains three integers n, m, and k, denoting the number of cake corners, the number of cuts, and the number of available colors ($3 \le n \le 10^9$; $0 \le m \le 2 \cdot 10^5$; $2 \le k \le 10^6$).

The i-th of the following m lines contains two integers u_i and v_i, denoting the corners connected by the i-th cut ($1 \le u_i < v_i \le n$). No two cuts may coincide or intersect except at the ends of the cuts. All cuts are straight, going strictly inside the cake.

It is guaranteed that the sum of m over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print the number of ways to color the cake corners such that no two neighboring corners have the same color, modulo 998 244 353. Remember that you don't have to use all the colors.

Example

standard input	standard output
4 4 1 3 1 3 5 0 2 9 4 3 1 3 1 6 4 6 6 8 3 0 1001	6 0 54 1754647

Note

In the first test case, corner 1 has one of 3 colors. Corner 2 has one of the remaining 2 colors. Corner 3 has the remaining color, and corner 4 has the same color as corner 2. Thus, there are 6 ways in total.

In the second test case, we have an odd number of corners and two colors, and every pair of consecutive corners must have different colors; that is impossible.

Problem D. Defective Script

Time limit: 2 seconds
Memory limit: 1024 megabytes

Devin is a system administrator at a tech company that manages a network of n servers arranged in a ring topology. Each server is handling a certain amount of computational load, represented by a non-negative integer a_i, where i ranges from 1 to n.

To optimize the network performance and ensure fairness, Devin wants to equalize the load across all servers, making each server handle the same amount of load. He aims to maximize this equal load as much as possible.

Devin has developed a script to reduce the load on any server. When he runs the script on server i, it is supposed to decrease the load on that server by 2 units (down to a minimum of zero). However, due to a known bug in the script, every time it's executed on server i, it inadvertently removes an additional 1 unit of load from the previous server in the network (server $i - 1$). If $i = 1$, the previous server is server n (since the servers form a ring).

Devin can run this buggy script any number of times (including zero), each time choosing any server to run it on. He can run the script on a server even if its current load is less than 2 units, or if the load of the previous server is zero (in both cases the load goes to zero).

Help Devin determine the maximum possible equal load that can be achieved on all servers using his script.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n, denoting the number of servers ($2 \le n \le 2 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$, denoting the amounts of load the servers are handling ($0 \le a_i \le 10^9$).

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print the maximum possible equal load that can be achieved on all servers.

Example

standard input	standard output
5 4 9 9 6 8 2 3 5 9 9 9 8 2 4 4 3 5 3 3 777 777 777 6 0 1 0 1 0 1	5 1 0 777 0

Note

In the first test case, Devin can run the script once on server 1, twice on server 2, and once on server 4. As a result, each server will be handling 5 units of load.

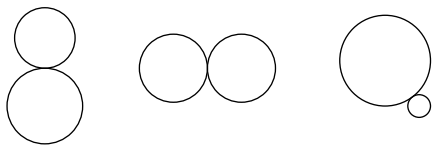
Problem E. Eight-Shaped Figures

Time limit: 5 seconds
Memory limit: 1024 megabytes

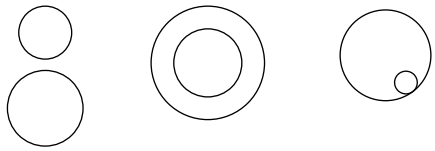
Looking at problems "K-Shaped Figures" and "H-Shaped Figures" from the past two years, you took the warning seriously. You came prepared. For each of the remaining 24 letters of the alphabet, you theorized what the problem could be. You even implemented all 24 solutions and used up all of your Digital Team Reference Document space just to bring these codes to the contest. If the judges are so unoriginal that they set another problem about letter shapes, you'll just get it accepted on minute 1 and leave everyone puzzled.

What, another shapes problem? Really?! Ha-ha! Oh... wait a second...

Let's say that two circles on a plane form an *8-shaped figure* if they touch each other, but neither of them lies inside the other one.



Valid 8-shaped figures



Invalid 8-shaped figures

You are given a collection of n circles on the plane. No two circles have more than one common point. In other words, no two circles intersect twice or coincide, but they can touch or lie one within another.

Find the number of pairs of circles from this collection that form an 8-shaped figure.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n, denoting the number of circles ($2 \le n \le 2 \cdot 10^5$).

The i-th of the following n lines contains three integers x_i, y_i, and r_i, denoting the coordinates of the center of the i-th circle and its radius ($-10^9 \le x_i, y_i \le 10^9$; $1 \le r_i \le 10^9$). No two circles intersect twice or coincide, but they can touch or lie one within another.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print the number of pairs of circles that form an 8-shaped figure.

Example

standard input	standard output
2 5 1 1 1 1 3 1 3 1 1 3 3 1 6 7 4 6 -3 0 3 -2 0 2 -1 0 1 1 0 1 2 0 2 3 0 3	5 9

Problem F. False Alarm

Time limit: 2 seconds
Memory limit: 1024 megabytes

Faina is going to sleep, but she needs to wake up early tomorrow for a very important contest. She has already set n alarms at different times between 7:00 and 9:00 in the morning.

However, Faina is a deep sleeper. She knows that in order to wake up, she will need to hear at least three alarms within a 10-minute timespan. In other words, for some three alarms, the difference between the first and the last alarm must be 10 minutes or less.

Faina is not sure if the current set of alarms she has satisfies this condition, and she is worried she might oversleep the contest (and make her teammates angry!). Thus, she wants to set some additional alarms. All new alarms should also be set between 7:00 and 9:00, and all alarms, including the old ones, must be set at different times.

Find the smallest number of additional alarms Faina has to set to be confident that she will wake up. In particular, if she can already be sure she'll wake up, the number of additional alarms is 0.

Input

The first line contains a single integer n, denoting the number of alarms Faina has set ($1 \le n \le 20$).

The i-th of the following n lines contains the time of the i-th alarm in the `h:mm` format ($7 \le$ `h` ≤ 9; $00 \le$ `mm` ≤ 59; if `h` $= 9$, then `mm` $= 00$). The alarms are given in strictly increasing order of time.

Output

Print the smallest number of additional alarms Faina has to set in order to guarantee waking up.

Examples

standard input	standard output
5 7:47 7:56 7:59 8:05 8:13	0
7 8:00 8:10 8:20 8:30 8:40 8:50 9:00	1
3 7:13 7:41 8:36	2

Note

In the first test, three alarms at 7:56, 7:59, and 8:05 guarantee that Faina will wake up.

In the second test, any time between 8:00 and 9:00 that does not coincide with existing alarms works.

In the third test, one possible solution is to set two more alarms at 7:45 and 7:46.

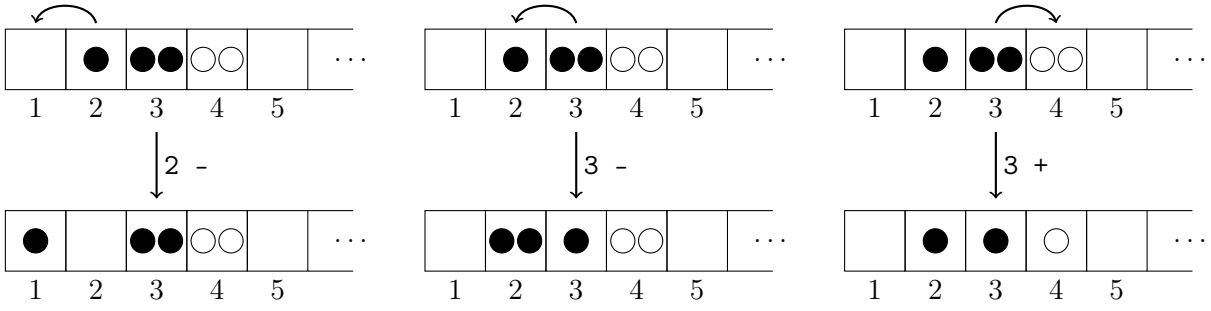
Problem G. Game of Annihilation

Time limit: 2 seconds
Memory limit: 1024 megabytes

Two players are playing a game on a tape divided into cells that is infinite to the right. The cells are numbered with $1, 2, 3, \ldots$ from left to right. Each cell x is adjacent to cells $x-1$ and $x+1$, except for cell 1, which is only adjacent to cell 2.

There is a finite number of red chips (the first player's chips) and blue chips (the second player's chips) on the tape. Each cell contains either several red chips, or several blue chips, or no chips at all.

The players take turns. On their turn, a player can either skip the turn or take one of their chips and move it to an adjacent cell. If there are no opponent's chips in the adjacent cell, the turn ends; if there is at least one opponent's chip there, one chip from each player is removed from that cell — thus, at the end of the turn, there will still be no two chips of different colors in the same cell.



Moving to an empty cell

Moving to a cell with your own chips

Moving to a cell with your opponent's chips

If both players run out of chips, the game ends in a draw. If only one player runs out of chips, they are declared the loser, and their opponent is declared the winner. Finally, if after 10^{100} turns the game has not ended, it is forcibly concluded and declared a draw.

You are given the initial setup of the tape. Determine who will win with perfect play from both players, and find any optimal first move for the first player.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n, denoting the number of cells that initially contain at least one chip ($2 \le n \le 2 \cdot 10^5$).

The i-th of the following n lines contains two integers x_i, m_i, and a character c_i, denoting the coordinate of the i-th non-empty cell from the left, the number of chips in it, and the color of these chips ($1 \le x_1 < x_2 < \cdots < x_n \le 10^6$; $1 \le m_i \le 10^6$; $c_i \in \{\texttt{R}, \texttt{B}\}$). There is at least one chip of each color on the tape.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print:

- "`First` x d", if the first player (moving red chips) will win;
- "`Second`", if the second player (moving blue chips) will win;
- "`Draw` x d", if the outcome of the game will be a draw.

In the first and third cases, x d specifies any winning or drawing move, respectively — that is, a move after which, with perfect play from the second player, there remains a possibility to win or draw the game.

Here, x is the coordinate of the red chip that the first player should move, and $d \in \{\texttt{-}, \texttt{+}\}$ is the direction of the move ('-' means the chip should be moved to cell $x-1$, and '+' means the chip should be moved to cell $x+1$). If d is '-', then x must be greater than 1. If you suggest that the first player skips their turn, print "`0 0`" instead of "x d".

You can print each letter in upper- or lowercase: for instance, the strings "`First`", "`FIRST`", "`fiRST`" will be considered equivalent by the checker.

Example

standard input	standard output
10 2 1 1 R 2 1 B 2 1 1 B 2 1 R 2 1 2 B 4 1 R 4 1 1 B 2 1 R 4 3 B 6 1 R 2 1 2 B 3 1 R 2 1 2 B 2 1 R 2 1 1 R 2 2 B 2 1 2 R 3 1 B 3 1 1 R 2 1 R 4 1 B 2 1 2 R 2 1 B	Draw 0 0 Draw 2 - Draw 4 - Draw 2 - Draw 0 0 Draw 2 + Second Draw 0 0 Draw 2 - First 1 +

Note

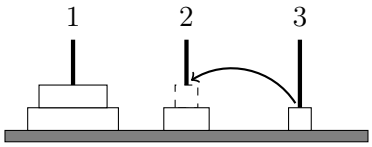
In the last test case, there is only one possible move besides "`1 +`", namely, "`0 0`" (skip the turn). It is a drawing move, though; hence, it will not be accepted.

Problem H. Hanoi Towers Reloaded

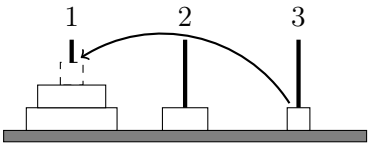
Time limit: 2 seconds
Memory limit: 1024 megabytes

The *Towers of Hanoi* is a famous mathematical puzzle consisting of three rods and n disks with diameters $1, 2, \ldots, n$. Each of the three rods contains some disks, stacked in order of decreasing diameter from bottom to top, so that the smallest disk is always at the top. A valid move consists of taking the smallest disk from a rod and putting it on top of another rod. This move must preserve the sorted order: you can't put a larger disk onto a smaller one. The original puzzle's goal is to transfer all disks from one rod to another.

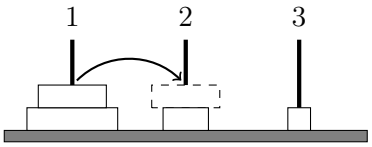
In this variation of the puzzle, you can only move the disks **between adjacent rods**: you can move a disk between rods 1 and 2, and between rods 2 and 3, but not between rods 1 and 3.



Valid move



Invalid move: non-adjacent rods



Invalid move: moving a larger disk onto a smaller one

Given two configurations of this puzzle, find the minimum number of moves required to reach the second configuration starting from the first one. As this number might be large, print it modulo 998 244 353.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^3$). Descriptions of the test cases follow.

The first line of each test case contains an integer n, denoting the number of disks involved ($1 \le n \le 10^5$).

The second line contains n integers $x_1, x_2, \ldots, x_n$, describing the initial configuration of the puzzle, where x_i is the rod that contains the i-th disk ($x_i \in \{1, 2, 3\}$).

The third line describes the final configuration of the puzzle in the same format.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, print the minimum number of moves required to reach the second configuration from the first one, modulo 998 244 353.

It can be shown that any two configurations are reachable from each other in this variation of the puzzle.

Example

standard input	standard output
4 1 1 3 2 3 3 2 1 3 3 2 1 1 2 3 4 2 1 3 2 2 1 3 2	2 7 20 0

Problem I. If I Could Turn Back Time

Time limit: 2 seconds
Memory limit: 1024 megabytes

Inna is an avid hiker. She's visiting a range of n mountains with heights $h_1, h_2, \ldots, h_n$.

At a nearby shop, Inna has found a book that mentions that at some point in the past, the heights of the mountains were $p_1, p_2, \ldots, p_n$ in the same order. However, there is no evidence of how old this book is.

The book also describes a model of erosion that makes the mountains shorter year after year. Every year, based on the weather, a certain height threshold x can be determined. Then, every mountain with the current height of at least x decreases in height by exactly 1. Different years can have different values of x.

Inna is curious how old the book actually is, and whether the described model is sound. Help her figure out the smallest number of years in which erosion could take the mountains from heights $p_1, p_2, \ldots, p_n$ to heights $h_1, h_2, \ldots, h_n$ in the same order, or determine that it is impossible under the given model.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n, denoting the number of mountains ($1 \le n \le 10^5$).

The second line contains n integers $h_1, h_2, \ldots, h_n$, denoting the current heights of the mountains ($1 \le h_i \le 10^6$).

The third line contains n integers $p_1, p_2, \ldots, p_n$, denoting the heights of the mountains in the same order at some point in the past ($1 \le p_i \le 10^6$).

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, print the smallest number of years in which erosion could take the mountains from heights $p_1, p_2, \ldots, p_n$ to heights $h_1, h_2, \ldots, h_n$, or a single integer -1 if the described model is unsound.

Example

standard input	standard output
4 4 3 2 4 2 5 3 6 2 1 10 100000 5 1 2 3 4 5 1 2 3 4 5 3 1 4 6 4 1 8	2 99990 0 -1

Note

In the first test case, the heights of the mountains could go from $(5, 3, 6, 2)$ to $(3, 2, 4, 2)$ in just two years:

- Suppose that in the first year, $x = 4$. After this year, the heights of the mountains are $(4, 3, 5, 2)$.
- Suppose that in the second year, $x = 3$. After this year, the heights of the mountains are $(3, 2, 4, 2)$.

Problem J. Just Half is Enough

Time limit: 2 seconds
Memory limit: 1024 megabytes

Jacob is studying graph theory. Today he learned that a *topological ordering* of a directed graph is a linear ordering of its vertices such that for every directed edge (u, v) from vertex u to vertex v, u comes before v in the ordering.

It is well-known that topological orderings exist only for graphs without cycles. But how do we generalize this concept for arbitrary graphs?

Jacob came up with the concept of a *half-topological ordering*: a linear ordering of the graph's vertices such that **for at least half** of all directed edges (u, v) in the graph, u comes before v in the ordering.

In other words, if the graph has m edges, and for a particular ordering, k of them satisfy the condition above, then the ordering is called *half-topological* if $k \geq \lceil \frac{m}{2} \rceil$.

Help Jacob find any half-topological ordering of the given graph, or report that none exist.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^4$). The description of the test cases follows.

The first line of each test case contains two integers n and m, denoting the number of vertices and the number of edges in the graph ($2 \leq n \leq 10^5$; $1 \leq m \leq 2 \cdot 10^5$).

The i-th of the following m lines contains two integers u_i and v_i, describing an edge from vertex u_i to vertex v_i ($1 \leq u_i, v_i \leq n$; $u_i \neq v_i$). The graph does not contain multiple edges: each directed edge (u, v) appears at most once. However, having both edges (u, v) and (v, u) is allowed.

It is guaranteed that the sum of n over all test cases does not exceed 10^5, and the sum of m over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print a single integer -1 if the required half-topological ordering does not exist.

Otherwise, print n distinct integers $p_1, p_2, \ldots, p_n$, describing the ordering of the given graph ($1 \leq p_i \leq n$). For at least $\lceil \frac{m}{2} \rceil$ of the edges (u_i, v_i), integer u_i must come before integer v_i in this list. If there are multiple answers, print any of them.

Example

standard input	standard output
2 3 3 1 2 2 3 3 1 5 6 4 2 2 1 4 3 1 4 3 2 3 5	1 2 3 3 1 5 4 2

Problem K. Keyboard Chaos

Time limit: 2 seconds
Memory limit: 1024 megabytes

Haven't you ever thought that an ordinary flat keyboard is boring, and you can come up with something more interesting?

A little boy named Kevin came up with a keyboard with n unusual keys. Each key i initially contains a sequence of letters: $L_{i,1}, L_{i,2}, \ldots, L_{i,|L_i|}$. Some letters in this sequence can be equal. Each letter is one of the first e lowercase English letters.

Every time key i is pressed, the first letter of its sequence is typed and immediately moved to the end of the sequence. Thus, the first time key i is pressed, letter $L_{i,1}$ is typed, and the sequence becomes $L_{i,2}, \ldots, L_{i,|L_i|}, L_{i,1}$. The second time key i is pressed, letter $L_{i,2}$ is typed, and the sequence becomes $L_{i,3}, \ldots, L_{i,|L_i|}, L_{i,1}, L_{i,2}$, and so on.

For example, suppose that key 1 contains the sequence '`a`', '`b`', '`a`', and key 2 contains the sequence '`c`', '`d`'. Then, if you press keys $2, 1, 2, 2, 1, 1, 1, 2$ in this order, the string "`cadcbaad`" will be typed.

Help Kevin understand how useful his keyboard is, and find the shortest possible string consisting of the first e lowercase English letters that cannot be typed with such a keyboard from the given initial state.

Input

The first line contains two integers n and e, denoting the number of keys and the size of the alphabet ($1 \le n \le 100$; $2 \le e \le 26$).

The i-th of the following n lines consists of characters $L_{i,1}, L_{i,2}, \ldots, L_{i,|L_i|}$, denoting the sequence of letters key i initially contains ($1 \le |L_i| \le 10$). Every character is one of the first e lowercase English letters.

Output

Print the shortest possible string, consisting of the first e lowercase English letters, that can not be typed using Kevin's keyboard from the initial state. If there are multiple shortest strings, print any of them.

If any string can be typed, print a single string "`NO`" instead.

Examples

standard input	standard output
1 26 win	f
3 3 abc bca cab	aa
4 2 aab bb a bab	NO

Note

In the first test, the only strings that can be typed with Kevin's keyboard are prefixes of "`winwinwinwin...`". Since you can not start the string with any letter other than '`w`', any lowercase English letter except '`w`' is a correct answer.

In the second test, "`bb`" and "`cc`" are other possible answers.

Problem L. Longest Common Substring

Time limit: 5 seconds
Memory limit: 1024 megabytes

Lisa wrote a program to solve the Longest Common Substring problem. She then used the program to find, for some two strings s and t consisting of characters '0' and '1', the longest string w that is a substring of both s and t. If there were multiple such longest strings, she found an arbitrary one.

Notably, the length of w Lisa found was very small — at most 3.

Lisa remembers n (the length of s), m (the length of t), and w, but she doesn't remember strings s and t themselves. Now she wonders how many pairs of strings s and t exist such that they have lengths n and m, respectively, consist of characters '0' and '1', and have w as one of their longest common substrings.

Help Lisa and find this number of pairs modulo 998 244 353. Note that if $n = m$ and $s \neq t$, pairs (s, t) and (t, s) are considered distinct.

Input

The first line contains three integers n, m, and k, denoting the lengths of the strings s, t, and w ($1 \le n, m \le 100$; $1 \le k \le \min(3, n, m)$).

The second line contains the string w of length k consisting of characters '0' and '1'.

Output

Print the number of pairs of strings (s, t) that have w as one of their longest common substrings, modulo 998 244 353.

Examples

standard input	standard output
2 2 1 1	6
3 4 2 01	28
7 5 3 110	399
23 42 3 000	174497840

Note

Note that a string a is a substring of a string b if a can be obtained from b by deleting zero or more characters from the beginning and zero or more characters from the end.

In the first test, all pairs of strings satisfying the conditions are ("01", "10"), ("01", "11"), ("10", "01"), ("10", "11"), ("11", "01"), and ("11", "10").

Problem M. Misère

Time limit: 2 seconds
Memory limit: 1024 megabytes

Préférence is a card game which is very popular in Eastern Europe. It is usually played with a 32-card deck, which consists of pip cards from 7 to 10, Jack, Queen, King, and Ace in each of the four suits: Spades, Clubs, Diamonds, and Hearts. In each round of the game, three players receive ten cards each, and two cards are left on the table as a talon. Then, a phase of auction happens, where players make their bids, which are obligations to take at least a certain number of tricks. A special case of a bid is a so-called *misère*, which is an obligation to take no tricks regardless of other players' moves.

In this task, we will consider a special modification of préférence which is played with a modified deck containing $A \cdot B$ cards, where A is a number of suits, and B is the number of ranks in each suit. For example, the standard 32-card deck for the préférence game has $A = 4$ suits and $B = 8$ ranks. For convenience, we'll number the suits from 1 to A, and the ranks from 1 to B.

You need to solve a puzzle about this modification of préférence. In this modification, we'll say that a misère is *guaranteed* if for every suit, after we order the cards belonging to this suit in your hand by their rank as $b_1 < b_2 < \cdots < b_k$ (where k is the number of cards of the suit in your hand), the following condition is satisfied: $b_i \le 2i - 1$ for all i from 1 to k. If you don't have any cards of the suit ($k = 0$), the condition is trivially satisfied.

You have n cards in your hand, and you will be allowed to choose any x cards you don't have and add them to your hand. Then, you must select any x of your $n + x$ cards and drop them, leaving some n cards in your hand. Your task is to find the smallest possible x such that you can transform your hand to a guaranteed misère.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). The description of the test cases follows.

The first line of each test case contains three integers n, A, and B, denoting the number of cards in your hand, the number of suits in the deck, and the number of ranks in the deck ($1 \le n \le 5000$; $1 \le A, B \le 10^9$).

The i-th of the following n lines contains two integers a_i and b_i and describes one card, where a_i is the suit of the i-th card, and b_i is its rank ($1 \le a_i \le A$; $1 \le b_i \le B$). All the cards in your hand are distinct.

It is guaranteed that the sum of n over all test cases does not exceed 5000.

Output

For each test case, print the smallest non-negative integer value of x such that you can transform your hand to a guaranteed misère by first adding x cards that you don't have to your hand, and then dropping any x cards from your hand.

It can be shown that such a value of x always exists.

Example

standard input	standard output
2 4 2 6 1 1 1 2 1 6 2 3 2 4 5 3 4 2 4	1 2