Problem A. Goldberg Machine 2

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	512 mebibytes

Rube is back in business with a new mind-blowing contraption! The contraption consists of a grid with n rows and m columns, with some cells of the grid containing an arrow pointing right or down, and other cells being empty. One can place a token in the top left corner cell of the grid, and then the magic happens: the token starts moving through the cells according to the arrow directions. Further, each arrow the token passes through switches direction when the token leaves its cell. Formally, if a token is in a cell with an arrow pointing right (resp. down), it then moves to the cell immediately to the right (resp. down), and the arrow in the original cell switches to pointing down (resp. right). The movement stops once the token arrives at an empty cell, or leaves the grid altogether, at which point the token is discarded.

As an example, consider a 3×3 grid shown below on the left ("`>`" and "`v`" describe arrows pointing right and down respectively, and "`.`" describes an empty cell). Two more grids show grid states after placing a token in the top left corner of the grid, and after placing another token after that.

```
>v>    v>>    >>>
vv> -> v>> -> >>>
v.>    v.>    >.v
```

Rube has manufactured many copies of his machine and sells them for a good price. However, he has been confronted by two very sophisticated customers who argue that their machines look the same, which offends their exquisite taste for truly unique things. Indeed, their machines have the same *shape*, that is, the sets of cells containing the arrows are equal for both machines, although the arrow configurations themselves may be different.

Rube agreed to modify the machines so that, despite looking similar, machines will never end up with the same arrow configurations after feeding any number of tokens in any, or both, of them. He is now bombarded with requests from the customers on how exactly machines should be modified: each request is to change a single arrow state in one of the machines. After each request Rube needs to determine if machines can eventually arrive to the same arrow configuration, and if so, what is the smallest total number of times you need to place a token in either of the machines before that happens, distributing the tokens arbitrarily between the two machines. Note that the changes **persist**, that is, no modification is undone after giving an answer to the request.

Once again, Rube is lamenting the complexity of his inventions since the task looks incredibly difficult. Help him answer all of customers' requests and avoid being sued out of existence.

Input

The first line contains three integers n, m, q — dimensions of the grid, and the number of requests ($1 \le n, m \le 100$, $1 \le q \le 10^5$).

The following n lines describe the state of the first machine. Each of these lines contains m characters. Each of the characters is one of "`>`", "`v`", or "`.`", which describe an arrow pointing right or down, or an empty cell respectively.

The following n lines after that describe the state of the second machine in the same format. It is guaranteed that the two grids have the same shape, that is, they contains arrows (regardless of directions) in the same set of cells. Further, it is guaranteed that the top left corner of each grid contains an arrow, and that a token can potentially reach each arrow cell after placing some number of tokens in each of the machines.

The following q lines describe requests in order they are received. Each request is described by three integers t, r, c — the number of the machine to be operated, and the row and the column indices of the cell

to be switched ($t \in \{1, 2\}$, $1 \le r \le n$, $1 \le c \le m$). Rows are numbered top to bottom, and columns are numbered left to right, both starting from 1. It is guaranteed that the cell to be switched in each request contains an arrow.

Extra empty lines may be present between sections for visual clarity.

Output

Print $q + 1$ lines, each containing answers after processing the first $0, 1, 2, \ldots, q$ requests respectively, in order. If the machines in their current states can not arrive to the same arrow configuration after any number of tokens placed in either of them, print -1 as the answer. Otherwise, print the smallest total number of tokens that need to be placed in the machines before they arrive to the same configuration.

Do not print leading zeros.

Example

standard input	standard output
3 3 9 >v> vv> v.> v>> v>> v.> 2 3 1 2 2 1 2 1 1 1 3 3 1 2 2 2 3 3 2 3 1 1 1 2 1 2 1	1 -1 -1 2 14 -1 -1 -1 -1 0

Problem B. Nein

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Given k and n, find the n-th positive integer x such that the decimal representation of $x \cdot \underbrace{999\ldots9}_{k}$ doesn't contain any 9.

Input

The only line contains two integers k and n ($1 \le k \le 18$, $1 \le n \le 10^{18}$).

Output

Print the answer.

Examples

standard input	standard output
1 1	2
1 8	9
1 9	12
1 10	13
5 1	11112
5 84	11235
5 668	12345
5 733942	2281488

Note

For $k = 1$, the sequence of all valid numbers starts with $2, 3, 4, 5, 6, 7, 8, 9, 12, 13, \ldots$

Problem C. MIPT: Connecting People

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Moscow IPT campus is under renovation. New student accomodations are located in the freshly built skyscraper housing estate. There are n skyscrapers in a row, the i-th from the start of the row having h_i floors. Skyscraper foundations are perfectly level with each other, and all floors have exact same height, thus same floors (counting from the bottom) in any pair of skyscrapers are on the same vertical level.

Each floor of each skyscraper is occupied by exactly one resident (how fancy is that!). Residents can move up and down inside each skyscraper via elevators. Moving one floor in any direction within the skyscraper i takes tv_i seconds.

Really, the only downside of the project is the lack of budget for guarding the entrances to the skyscrapers, thus, for the safety purposes, at the moment entering or leaving the complex is not possible at all. To make up for this, it was decided to build extra corridors connecting skyscrapers in the complex. Each corridor must be perfectly horizontal, thus it must connect floors with same numbers in respective skyscrapers. Further, the corridor can not overlap with any skyscraper standing in between its endpoints. Formally, if floors x are connected with a corridor in skyscrapers i and j, then $h_k < x$ must be satisfied for all k such that $i < k < j$ (and also, naturally, $h_i, h_j \geq x$ must hold).

Corridors are state-of-the-art, thus travelling through any corridor takes th seconds, regardless of the distance travelled. However, they are also expensive, thus only $n - 1$ of them can be built.

To make residents happier (and also complain less about being imprisoned), the following conditions must be satisfied:

- It has to be possible to get from any floor of any skyscraper to any other floor of any other skyscraper via elevators and corridors.
- If we arbtrarily number all residents from 1 to $R = \sum_{i=1}^{n} h_i$, and define $d(x, y)$ as the smallest time (in seconds) the resident x needs to get to the accomodation of the resident y via elevators and corridors, then $\sum_{1 \leq x < y \leq R} d(x, y)$ must the as small as possible.

Help the MIPT planning board to complete this astounding project.

Input

The first line contains two integers n and th — the number of skyscrapers and the time (in seconds) needed to travel any horizontal corridor respectively ($1 \leq n \leq 60$, $1 \leq th \leq 10^6$).

The following n lines describe the skyscrapers. The i-th of these lines contains two integers h_i, tv_i — the number of floors (as well as residents), and the time (in seconds) needed to travel one floor vertically within the skyscraper i respectively ($1 \leq h_i \leq 3000$, $1 \leq tv_i \leq 10^6$).

It is guaranteed that $R = \sum_{i=1}^{n} h_i \leq 3000$.

Output

Print a single integer — the smallest value of $\sum_{1 \leq x < y \leq R} d(x, y)$ over all valid ways to construct $n - 1$ corridors, as defined above.

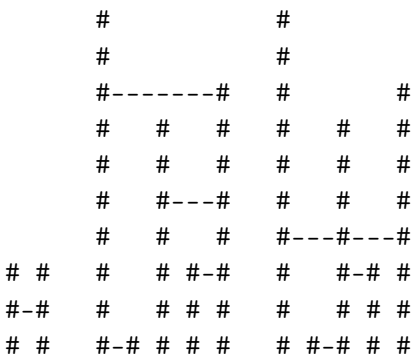
Examples

standard input	standard output
1 1 5 1	20
2 1 3 3 3 2	59
5 1000 10 1 1 1 7 1 3 1 8 1	460314
5 1 10 1000 1 1000 7 1000 3 1000 8 1000	1626464

Note

In the first sample, there are no corridors, thus the answer is simply the sum of vertical distances.

Optimal configurations for the other sample tests are pictured below:



Problem D. Matryoshka Dolls

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

Denisson is an average enjoyer of Matryoshka dolls. A set of matryoshkas consists of a wooden figure, which separates at the middle, top from bottom, to reveal a smaller figure of the same sort inside, which has, in turn, another figure inside of it, and so on.

Recently he bought a set of these toys and placed them on the table in some order. There are exactly n dolls of different sizes in his set, so the current order of matryoshkas can be represented as a permutation p of length n, where p_i is equal to the size of i-th doll.

Denisson usually has a very tight schedule, but today he wants to relax with his toys, and will play the following game:

- Firstly, he will choose some segment $[l, r]$ of his dolls permutation;
- Then he will take the smallest matryoshka on this segment and place it into the matryoshka of the next size. The time needed for him to perform this procedure equals $|i - j|$ where p_i and p_j are sizes of the two smallest dolls on the chosen segment;
- He will repeat this action until there is only one matryoshka left on the segment. The total time needed for his game is the sum of times needed for all performed procedures.

Suddenly, he realized that his interesting game could last for a very long time, but he really cares about his schedule. He came to you with q different segments $[l_i, r_i]$ and wonders what time he needs to play the game on each of these segments. He hopes that you will not spend too much time finding it out.

Input

The first line contains two integers n and q — the number of matryoshkas and the number of requests ($1 \le n \le 10^5$, $1 \le q \le 5 \cdot 10^5$).

The second line describes the order of the matryoshkas represented as a permutation p ($1 \le p_i \le n$).

The following q lines describe requests in the order they are received. Each request is described by two integers l_i and r_i ($1 \le l_i \le r_i \le n$) — endpoints of the i-th segment.

Output

For i-th request, print the total time needed to play the game on segment $[l_i, r_i]$.

Example

standard input	standard output
5 5 1 5 2 4 3 1 5 1 4 1 3 1 2 1 1	7 5 3 1 0

Problem E. No Rest for the Wicked

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

The 41st Petrozavodsk Programming Camp comes to its end. We hope that you enjoyed the last week. We will probably meet some of you at the ICPC 2020 World Finals, which is planned to be held in about a month, and there are different strategies to spend this time. One option is to train hard and solve more problems. Another possible option is, on the opposite, to clear your mind and take a good rest after all this work.

You may have probably noticed that this camp was held online. COVID-19 has changed the way we live now. It has also constrained travel possibilities a lot.

Let's say that there are n countries (enumerate them from 1 to n for convenience), and there are also m bidirectional flights between them. Each country i has three properties: a spectacularness value s_i, a COVID level c_i and a security threshold $t_i \geq c_i$. Their meaning is the following: if one wants to fly to the j-th country, and they have ever been to the i-th country, then $c_i \leq t_j$ must hold.

Assume that a person from the i-th country wants to visit other countries, and their goal is to go the most spectacular they can — that is, to eventually visit a country j with the maximal possible s_j. Find this spectacularness value for each starting i.

Note that there always is an option to stay home, so the answer always exists. For the sake of simplicity we **do not** require the possibility to return home after visiting the most spectacular possible country (let's say that you can always go home somehow even if there are no flights to it).

Input

The first line contains two integers n and m separated by space ($1 \leq n, m \leq 2 \cdot 10^5$). Then n lines follow, i-th of them contains three space-separated integers c_i, t_i, and s_i ($1 \leq c_i, t_i, s_i \leq 10^9$, $c_i \leq t_i$).

The following m lines describe edges, each of them containing two integers u and v ($1 \leq u, v \leq n$) and denoting a flight between u and v. The described graph is guaranteed to have no self-loops and no multiple edges.

Output

Print n integers, i-th of them being the maximal spectacularness one could see starting from the i-th country.

Example

standard input	standard output
4 3 2 3 1 1 1 4 1 2 2 1 1 3 1 2 1 3 1 4	2 4 2 3

Problem F. The Last Samurai

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

This is an output-only problem.

The chess developers released a new custom mode called "The Last Samurai". On the initial configuration of the board there is one black king and multiple white pieces. The only moving piece is the black king, and the goal is to capture all white pieces without ever putting itself under attack.

Consider a strategy that repeats the following steps:

1. If there are no white pieces left, black have won.
2. If there is no way to capture any of the remaining white pieces, black have lost.
3. Otherwise, consider the shortest sequence of moves that the black king can take to capture a white piece while not placing itself under attack at any point (including immediately after capturing). If there are several different pieces that can be captured in the smallest number of moves, pick one in the topmost possible row; if there is still a tie, pick the leftmost possible piece.
4. Perform the chosen sequence of moves to capture a piece; repeat from step 1.

You need to design a level with the following properties:

- The board size is at most 200×200 (that is, no side exceeds 200).
- Each white piece is either a rook, or a bishop, or a knight.
- There is exactly one black king on the board, which is initially not under attack from any white piece.
- The greedy strategy above successfully completes the level, but makes more than 10^6 moves with the black king.

Please provide any level satisfying these requirements.

Input

The problem has no input.

Output

In the first line print two space-separated integers n and m ($1 \le n, m \le 200$) standing for the size of the board. In the next n lines print the level description. More specifically, the i-th of them must be a string consisting of characters from "`.rbnkRBNK`", where the j-th of them is

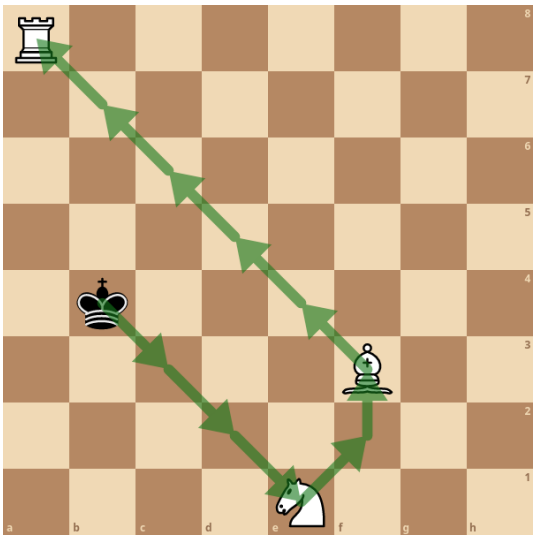
- "`.`" if the corresponding cell is free,
- "`r`" or "`R`" if the corresponding cell is occupied by a white rook,
- "`b`" or "`B`" if the corresponding cell is occupied by a white bishop,
- "`n`" or "`N`" if the corresponding cell is occupied by a white knight,
- "`k`" or "`K`" if the corresponding cell is occupied by the black king.

Example

standard input	standard output
<empty>	8 8 r....... .... .K...... .....b.. ....n...

Note

The sample output is only given to clarify the output format. The greedy strategy captures all pieces in 10 moves, with the king's route shown below:



Problem G. Nikanor Loves Games

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Nikanor spends all his free time on games. Because of this, he gets bad marks at the university, but that's another story. He also likes gambling. In this problem, we consider the modification of the game called "Orlyanka". There are two players, and each of them has his own coin. Each of the two sides of a coin contains an integer. Players toss their coins, and the winner is the one with the highest number. We can assume that for each coin the probabilities of coming up both sides are equal.

Tonight Nikanor is playing this game with his friends. Nikanor has n friends, and he will play with each of them for a bet of x_i rubles. Fortunately, Nikanor knows that his i-th friend has a coin with the numbers a_i and b_i. If Nikanor wins against his friend, he will receive x_i rubles. Otherwise, he will pay x_i rubles to his friend. If Nikanor and his friend dropped the same value, Nikanor is declared the winner.

Now Nikanor is going to go to the store and buy one coin for all games to maximize his expected profit, taking the coin cost into account. In this shop, a coin with the numbers a and b costs $a \cdot b$ rubles. Nikanor can buy any coin with positive integers.

It's so hard for Nikanor to make the right decision... Nikanor asks you to help him choose a coin so that the expected profit is as high as possible.

Input

The first line contains one integer n ($1 \le n \le 2 \cdot 10^5$) denoting the number of friends.

Each of the following n lines contains three integers a_i, b_i, and x_i ($1 \le a_i, b_i, x_i \le 10^9$) representing the numbers on i-th friend's coin and i-th bet in rubles.

Output

Print a single integer — the maximum expected profit.

Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6}.

Formally, let your answer be a, and the jury's answer be b. Your answer will be accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Examples

standard input	standard output
2 1 4 15 3 5 10	2.5000000000
1 2 2 8	4.0000000000

Problem H. Roads of the Empire

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

King is dead. Long live the king!

The young emperor has just inherited the vast empire of his father. After the successful reign, it consists of such many cities that it is plausible to consider that there are infinitely many of them.

However, the transportation system of the empire is pretty poor. As all new rulers do, the emperor wants to change his predecessor's policies. Thus, instead of war, he decides to build some new roads in the empire. However, the country's religious beliefs require the emperor to follow a specific ritual of building new roads.

First, he has to choose a positive number n. Then, n cities are taken, numbered from 1 to n. After that, for all pairs of cities with numbers x and y such that $1 \le x < y \le n$, the road between them is built if and only if $x + n$ is evenly divisible by y.

After all those manipulations, the emperor has to choose two cities u and v, satisfying $1 \le u, v \le n$, and find the number of roads in the shortest path between them. Moreover, he must choose them randomly and equiprobably to be blessed by God, the holy protector of the empire. Knowing this length, the Religious Council will decide whether the plan is acceptable.

The emperor is worried before the meeting with the Council, so he prepared several such plans. For every plan, answer the question of the length of the shortest path!

Input

The first line contains one integer T ($1 \le T \le 2 \cdot 10^5$) — the number of plans proposed by the emperor.

Each of the following T lines contains three integers n, u, v ($1 \le n \le 10^{18}, 1 \le u, v \le n$) — the number of cities chosen for the plan, and the two cities for which you are asked to find the length of the shortest path in the described graph.

It is guaranteed that u and v were chosen randomly and equiprobably after n had been selected for each test.

Output

For every query, print a number on a separate line — the length of the shortest path between the corresponding cities. If the path does not exist, print -1.

Examples

standard input	standard output
4 5 1 2 8 2 5 7 7 2 6 2 5	1 1 -1 2
1 88 14 2	2

Problem I. Drunkards

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

He is a positive man in every way, except that he goes to the bar every evening...

One of your friends is a bartender in the city's most famous (and only) bar. There are $2 \cdot n + 1$ houses in the city, located along one long road and numbered from 0 to $2 \cdot n$. The bar is located in the house numbered n.

The interesting fact is that all drunkards in the city have the same habit. Of course, they leave the bar having a condition not allowing them to go along a way home, so they start to walk without a goal. Namely, every drunkard has an array a in mind, which has length n. In the i-th second after leaving the bar, the drunkard wants to change his position in the road by a_i ($|a_i| = 1$). If the drunkard was in front of the house j, he would be in front of the house $j + a_i$ after this change.

However, they are so drunk that each second with probability $\frac{p}{100}$ they are not capable of moving and stay in their current position.

If a drunkard arrives in front of his house, his family members see him and take him home. Possibly, if he lives in the n-th house itself, his family members will take him immediately. However, after n seconds, if not taken, a drunkard becomes disappointed and sleeps in the street.

Another drunkard came to the bar. The bartender does not know where he lives, so he just assumes for every house the probability is $\frac{1}{2 \cdot n + 1}$ that the drunkard lives there. Calculate the probability that his family members will take him home modulo 998 244 353.

Input

The first line contains two integers n and p ($1 \le n \le 5000$, $0 \le p \le 100$), which are described in the statement.

The second line contains n integers $a_1, \ldots, a_n$ ($|a_i| = 1$) — the drunkard's intentions in the i-th second.

Output

Output one integer — the answer modulo 998 244 353.

Formally, let $M = 998\,244\,353$. It can be shown that the answer can be expressed as an irreducible fraction $\frac{p}{q}$, where p and q are integers and it is guaranteed that q is not divisible by M. Output the integer equal to $p \cdot q^{-1} \bmod M$. In other words, output such an integer x that $0 \le x < M$ and $x \cdot q = p \pmod{M}$.

Example

standard input	standard output
2 28 1 1	702764025

Problem J. Rational Dimasik

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Little Dimasik is a rational numbers fan. He has n rational numbers $\frac{x_i}{y_i}$. Recently Dimasik learned how to subtract rational numbers.

Recall that every rational number may be expressed in a unique way as an irreducible fraction $\frac{a}{b}$, where a and b are coprime integers and $b > 0$.

Let us define the function $d\left(\frac{x_i}{y_i}\right)$ as the denominator of the rational number $\frac{x_i}{y_i}$ in irreducible notation. For example, $d(\frac{14}{6}) = d(\frac{7}{3}) = 3$.

Now Dimasik wants to calculate the value

$$\prod_{1 \le i < j \le n} d\left(\left|\frac{x_i}{y_i} - \frac{x_j}{y_j}\right|\right).$$

But soon he realized that this problem is too hard for him. Dimasik asks you to help him. As the value may be very large, find it modulo 998 244 353.

Input

The first line contains one integer n ($1 \le n \le 2 \cdot 10^5$) denoting the number of rational numbers Dimasik has.

Each of the following n lines contains two integers x_i and y_i ($0 \le x_i \le 10^9$, $1 \le y_i \le 10^6$) representing the numerator and denominator of the i-th rational number.

Output

Print a single integer — the answer to the problem modulo 998 244 353.

Examples

standard input	standard output
2 1 3 3 7	21
3 3 2 7 15 5 12	7200

Problem K. Mission Impossible: Grand Theft Auto

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Tom Cruise had his car stolen during the filming in Birmingham. You, as the chief police officer, are asked to catch the thief.

Your people are patrolling the countryside, so the criminal can only be in one of n towns, and the highways between them represent a tree — that is, there are $n-1$ bidirectional roads each connecting a pair of towns, and it is possible to traverse from each of them to any other. On each day you can pick any two towns A and B (not necessarily distinct) and request a troop which will check every town on the simple path between A and B (including both of them). If the thief is in one of these towns then it's game over for him. Otherwise, later that night he can move into any other **adjacent** town by a single road or stay still in the town where he was.

Since this is a very important case, you are very short of time. More specifically, let m be the number of leaves in the tree (that is, towns with only one outgoing road). Then you have to come up with a plan of $\lfloor m/2 \rfloor + 1$ days which catches the thief: for each day you tell the corresponding A and B for this day, and your goal is to catch the guy independently on his actions between the days.

Find a plan satisfying the requirements. You will have to answer several test cases.

Input

The first line of input contains the only integer T ($1 \le T \le 100$) — the number of test cases. T tests follow.

The first line of each test case contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — the number of towns. Each of the next $n-1$ lines consists of two integers u and v separated by space ($1 \le u, v \le n$), denoting a road between towns u and v.

It is guaranteed that each test case represents a tree, and that the sum of all n does not exceed $2 \cdot 10^5$.

Output

For each test case, print exactly $\lfloor m/2 \rfloor + 1$ lines, each consisting of two integers A and B ($1 \le A, B \le n$) denoting the endpoints of the corresponding path. If you are sure that you can catch the thief using less operations, just add arbitrary paths at the bottom. One can prove that an answer always exists under these constraints.

Example

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

Note

Extra lines in the sample output are to visually separate answers for different cases. You do not have to print them.