

## Problem A. Ban or Pick, What's the Trick

Input file:           standard input  
Output file:         standard output  
Time limit:          4 seconds  
Memory limit:       1024 megabytes

Bobo has recently learned how to play Dota2. In Dota2 competitions, the mechanism of banning/picking heroes is introduced, modified and simplified as follows for the sake of the problem:

Suppose a game is played between two teams: Team A and Team B. Each team has a hero pool of  $n$  heroes with **positive** utility scores  $a_1, \dots, a_n$  and  $b_1, \dots, b_n$ , respectively. **Here we assume all heroes in two teams' hero pool are distinct.**

The two teams then perform ban/pick operations alternately, with Team A going first. In one team's turn, it can either pick a hero for itself, or ban an **unselected** hero from the opponent's hero pool.

After  $2n$  turns, all heroes are either picked or banned. Each team then needs to choose **at most**  $k$  heroes from **all heroes it picked** to form a *warband* and the score for the warband is calculated as the sum of utility scores over all heroes in it.

Let  $s_A, s_B$  be the score of the warband formed by Team A and Team B, respectively. Team A wants to maximize the value of  $s_A - s_B$  while Team B wants to minimize it.

Bobo wants to know, what should be the final value of  $s_A - s_B$ , if both teams act optimally? He's not really good at calculating this, so he turned to you for help.



An example of banning/picking heroes in Dota2. Source: TI10 True Sight

### Input

The first line contains two integers  $n, k$  ( $1 \leq n \leq 10^5, 1 \leq k \leq 10$ ).

The second line contains  $n$  integers  $a_1, a_2, \dots, a_n$  ( $1 \leq a_i \leq 10^8$ ), denoting the utility score of heroes for Team A.

The third line contains  $n$  integers  $b_1, b_2, \dots, b_n$  ( $1 \leq b_i \leq 10^8$ ), denoting the utility score of heroes for Team B.

### Output

Output an integer in one line, denoting the answer.

## Examples

standard input	standard output
2 1 3 6 2 4	2
4 1 1 3 5 7 2 4 6 8	0
4 2 4 6 7 9 2 5 8 10	3

## Problem B. Call Me Call Me

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

Bobo is going to organize a conference. He has a list of  $n$  people, and decides to call them one by one to invite them to the conference. However, people are only likely to attend a conference with enough acquaintances, that's why the  $i$ -th people say,

“Call me and I'll attend the conference when there are **at least**  $k_i$  people in the interval  $[l_i, r_i]$  who already decided to attend!”

Bobo wonders, what's the largest number of people he can invite to the conference?

### Input

The first line contains an integer  $n$  ( $1 \leq n \leq 4 \times 10^5$ ), denoting the number of people on the list.

Then  $n$  lines follow. The  $i$ -th ( $1 \leq i \leq n$ ) line contains three integers  $l_i, r_i, k_i$  ( $1 \leq l_i \leq r_i \leq n, 0 \leq k_i \leq n$ ).

### Output

Output an integer in one line, denoting the largest number of people Bob can invite to the conference.

### Examples

standard input	standard output
3 2 3 2 2 3 1 2 2 0	3
3 2 3 1 2 3 2 2 2 0	2

### Note

For the first sample test, Bob can invite people in the order of (3, 2, 1), so that all three people will attend the conference.

For the second sample test, after inviting the third person and the first person in order, the second person will reject Bobo's invitation as his/her requirement is not met.

## Problem C. Catch You Catch Me

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

Bobo accidentally dropped into the secret garden when he was listening to *Song from a secret garden*. Guess what did he see? Butterflies!



Yes, it is.

The structure of the secret garden can be modeled as an undirected tree of  $n$  nodes labeled from 1 to  $n$ , where the node labeled 1 is the exit of the secret garden. There is initially a butterfly at each node **labeled from 2 to  $n$** . Bobo noticed that, at each minute, every butterfly would move through an edge towards the direction of the node labeled 1. When a butterfly flies to the node labeled 1, it leaves the secret garden and disappears forever.

Bobo wants to catch all the butterflies before they fly to the exit and disappear. Bobo is initially not at any node of the tree. What he can do is to choose **any** node of the tree at some minute (including minute 0, where all butterflies are at their respective nodes) and catch all butterflies in that node (Bobo cannot catch butterflies at node 1 as butterflies disappear instantly when they get there). This counts as an operation. Bobo acts really fast, and the time it takes for him to operate can be neglected. Bobo wonders, what is the minimum number of times he has to operate to catch all butterflies?



### Input

The first line contains an integer  $n$  ( $1 \leq n \leq 10^5$ ), denoting the number of nodes in the tree.

Then  $n - 1$  lines follow. Each line contains two integers  $u, v$  ( $1 \leq u, v \leq n, u \neq v$ ), denoting the number of edges in the tree.

It is guaranteed that the input forms a valid tree.

Output

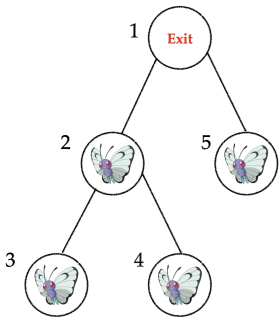
Output an integer in a line, denoting the minimum number of times Bobo has to operate to catch all butterflies.

Example

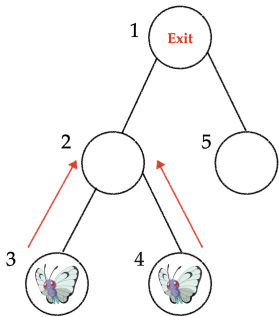
standard input	standard output
5 1 2 2 3 2 4 1 5	3

Note

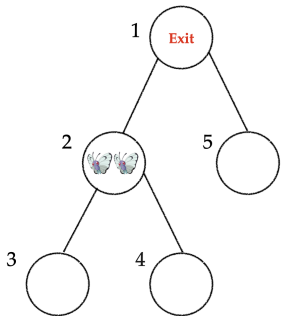
For the first sample test case, the initial state at minute 0 is shown as follows:



At minute 0, Bobo **must** perform an operation on node 2 and node 5, otherwise the two butterflies at the two nodes will reach the exit in the next minute.



The two butterflies initially at nodes 3 and 4 both arrive at node 2 at minute 1. Then Bobo can perform one operation at node 2 to catch them both. This takes three operations in all.



## Problem D. Gambler's Ruin

Input file:           standard input  
Output file:         standard output  
Time limit:          5 seconds  
Memory limit:       1024 megabytes

The football match between Bobo United (BU) and Bobo City (BC) is about to start. As an odds compiler working for a gambling company, Bobo needs to set odds for each team.

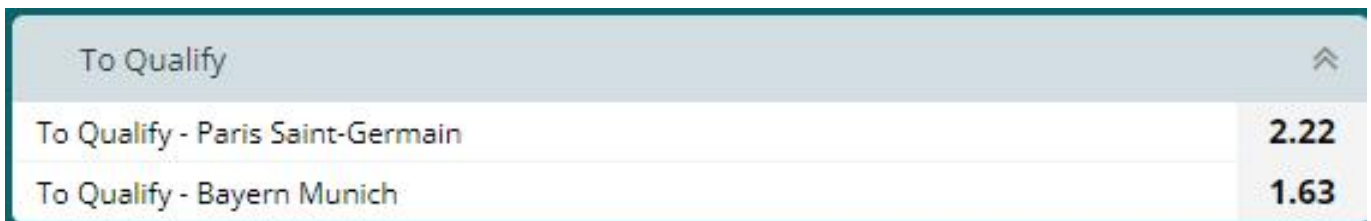
There are  $n$  gamblers ready to gamble on this game, and each has an estimated  $p_i$  of BU's probability of winning. Here, we consider the setting that the gambling company previously collects all gamblers' information, so each  $p_i$  is known.

If you set odd  $x$  for BU and odd  $y$  for BC, then for each gambler  $i$ :

- if  $p_i \cdot x \geq 1$ , he/she will bet  $c_i$  dollars on BU.
- **otherwise**, if  $(1 - p_i) \cdot y \geq 1$ , he/she will bet  $c_i$  dollars on BC.

Suppose the total amount of money bet on BU is  $s_x$  dollars and the total amount of money bet on BC is  $s_y$  dollars. If BU eventually wins the match, the company needs to pay out  $s_x \cdot x$  dollars; if BC wins, the company needs to pay out  $s_y \cdot y$  dollars. **In the worst case**, the profit of the gambling company is  $s_x + s_y - \max(s_x \cdot x, s_y \cdot y)$  dollars (the profit might be negative, meaning the company actually loses money).

Bobo needs to set the value of  $x$  and  $y$  to maximize the profit **in the worst case**, or otherwise, he might be fired by the company. Can you help him?



To Qualify	
To Qualify - Paris Saint-Germain	2.22
To Qualify - Bayern Munich	1.63

An example of pot odds offered by the online gambling company. Source: some mysterious website

### Input

The first line contains an integer  $n$  ( $1 \leq n \leq 10^6$ ), denoting the number of gamblers.

The  $n$  lines follow. The  $i$ -th ( $1 \leq i \leq n$ ) line contains a real number  $p_i$  and an integer  $c_i$  ( $0 \leq p_i \leq 1, 1 \leq c_i \leq 10^8$ ). with meaning already given in the statement. It is guaranteed  $p_i$  contains at most 6 digits after the decimal point.

### Output

Output a number in one line, denoting the maximum profit the gambling company can get in the worst case by optimally setting the value of  $x$  and  $y$ . 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 considered correct if  $\frac{|a-b|}{\max(b,1)} \leq 10^{-6}$ .

## Examples

standard input	standard output
2 1 15 0 10	10.0000000000
3 0.4 100 0.5 100 0.6 100	33.3333333333

## Problem E. Hammer to Fall

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

Boboland is a beautiful country with  $n$  cities and  $m$  two-way roads, ruled by King Bobo. A two-way road  $(u, v, w)$  connects city  $u$  and city  $v$  with distance  $w$ . It is guaranteed that for each city in Boboland, at least one road connects to it.

There were  $a_i$  residents in the  $i$ -th city, living a happy and peaceful life until the day came when it began to fall hammers! Specifically, every day hammers will start to fall on exactly one city of Boboland, causing all residents in that city to die.

As the king of Boboland, Bobo needs to deal with this situation. But unfortunately, he doesn't know why this disaster would happen, and he can't figure out a way so that the hammer would eventually stop falling. Despite this, he figured out the following "dynamic zero-casualty policy": it is OK as long as when the hammer falls on some city someday, all residents in that city have already been transferred to some other city so that no death is incurred.

After talking with the prophet in Boboland, Bobo has gained the following information: in the upcoming  $q$  days, hammers will fall on cities  $b_1, b_2, \dots, b_q$  in order. At any time, Bobo can arrange to transfer any single resident at some city  $u$  to some adjacent city  $v$  with cost  $w$  if a road  $(u, v, w)$  exists. Multiple residents can be transferred simultaneously. Also, any resident can be transferred multiple times.

Bobo wants to ensure no resident dies after the  $q$  days, using as little cost as possible. But, as always, a king never does the counting by himself, so you have to calculate the minimum cost to achieve the goal.

Since the answer might be large, you should output the answer modulo 998244353. Note that your minimization target is still the cost before modulo, not after.

### Input

The first line contains three integers  $n, m, q$  ( $2 \leq n \leq 10^5, 1 \leq m, q \leq 10^5$ ), denoting the number of cities in Boboland, the number of roads in Boboland, and the length of information from the prophet, respectively.

The second line contains  $n$  integers  $a_1, a_2, \dots, a_n$  ( $1 \leq a_i \leq 10^9$ ), denoting the number of residents in each city.

The  $m$  lines follow, each line containing 3 integers  $u, v, w$  ( $1 \leq u, v \leq n, u \neq v, 1 \leq w \leq 10^9$ ), denoting a two-way road  $(u, v, w)$  in Boboland. It is guaranteed that for each city in Boboland, at least one road connects to it.

The next line contains  $q$  integers  $b_1, b_2, \dots, b_q$  ( $1 \leq b_i \leq n$ ), denoting the city on which the hammer will fall in the upcoming  $q$  days, in order.

### Output

Output an integer in one line, denoting the minimum cost to achieve Bobo's goal, modulo 998244353.



## Examples

standard input	standard output
3 2 2 1 1 1 2 3 10 1 2 1 3 2	12
2 1 10 5000 5000 1 2 10000 1 2 2 1 2 2 1 1 1 2	550000000

## Note

For the first sample test, an optimal arrangement is to transfer the only resident at city 3 to city 2 at the beginning of the second day and then transfer the two residents at city 2 to city 1 at the beginning of the third day. The total cost is  $10 \cdot 1 + 1 \cdot 2 = 12$ .

## Problem F. Infinite Strife

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

The dispute between Bobotown and Boboland has not been resolved over decades. As the prime minister of Bobotown, Bobo has sent down an instruction to deploy  $n$  newly developed weapons located at  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ . The coverage of the  $i$ -th weapon is determined by a tunable integer parameter  $k_i$  ( $0 \leq k_i < 2m$ ). Specifically, the coverage of the  $i$ -th weapon with parameter  $k_i$  is all points  $(x, y)$  satisfying

$$x \cos \frac{k_i}{m} \pi + y \sin \frac{k_i}{m} \pi \geq x_i \cos \frac{k_i}{m} \pi + y_i \sin \frac{k_i}{m} \pi.$$

The territory of Boboland is an axis-aligned square of side  $2R$  centered at the origin. Bobo now wonders about the number of choices of integer parameters  $k_1, k_2, \dots, k_n$  satisfying that the territory of Boboland is fully covered by these weapons; more precisely, every point of the Boboland's territory is within the coverage of at least one deployed weapon. Sadly, as the vice prime minister of Bobotown, you have to do all the calculations.

The answer might be enormous, and you should output the answer modulo 998 244 353.

### Input

The first line contains three integers  $n, m, R$  ( $1 \leq n \leq 100, 1 \leq m, R \leq 10$ ), as stated above.

The  $i$ -th of the next  $n$  lines contains two integers  $x_i, y_i$  ( $-10 \leq x_i, y_i \leq 10$ ), denoting the location of the  $i$ -th weapon.

### Output

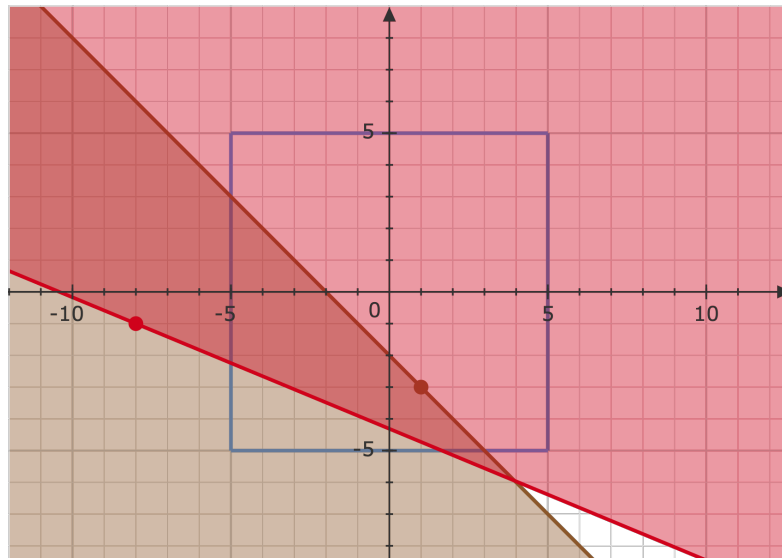
Output an integer in a line, denoting the answer modulo 998 244 353.

### Examples

standard input	standard output
2 8 5 1 -3 -8 -1	71
1 8 8 1 2	0

### Note

For the first sample test case, one of the possible choices of parameters is 10 and 3, illustrated below.



For the second sample test case, there exists no parameter that satisfies the requirement.

## Problem G. Let Them Eat Cake

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

*Let them eat cake.* King Bobo said while looking down at the  $n$  poor people begging on their knees inside the majestic palace. What makes King Bobo different from Marie Antoinette is that King Bobo truly *did* offer them cakes.

King Bobo lets the  $n$  people stand in a line, with the  $i$ -th ( $1 \leq i \leq n$ ) people from the left having a label  $a_i$ . **All labels are integers from 1 to  $n$  and pairwise distinct.** King Bobo then offers the cakes in rounds, by the following rule:

- When only one person is in the line, King Bobo gives him/her a cake, and the process ends.
- Otherwise, King Bobo offers a cake to all people whose label is smaller than some of his/her adjacent people (Each person may have 1 or 2 adjacent people, for those with 2 adjacent people, having a label smaller than either of the two can grant him/her a cake). Those who received the cake leave the line, and the remaining ones close the gaps and maintain the same order in the line. This counts as a round.

King Bobo wants to know how many rounds will be counted until the process ends. A king will never do the counting by himself, so it becomes your work.

### Input

The first line contains an integer  $n$  ( $1 \leq n \leq 10^5$ ), denoting the number of poor people.

The next line contains  $n$  integers  $a_1, a_2, \dots, a_n$  ( $1 \leq a_i \leq n$ ,  $a_i$  is pairwise distinct), denoting the label of each person in the line from left to right.

### Output

Output an integer in a line, denoting the answer.

### Examples

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

### Note

For the first sample test, in the first round, people with labels 1, 2, 3, 4 receive a cake and leaves the line, leaving the line with only one person. So the process ends in one round.

For the second sample test, in the first round, people with labels 1, 2, 3 receive a cake and leaves the line; in the second round, the person with label 4 receives a cake and leaves the line. So the process ends in two rounds.

## Problem H. Life is Hard and Undecidable, but...

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

*To John Horton Conway, his Game of Life, and also the life we are living...*

**Disclaimer:** For those who have no patience for the whole context or are already familiar with Game of Life, the problem is in the last paragraph on the third page.

Life is hard. There are beautiful dreams, yet you can never realize them. There are regrets and lost loved ones, yet you can never turn back time. There is death lying ahead with the anxiety it persistently gives you, yet you have no way to escape from it. There are problems right here for you to solve, yet you simply don't know how.

99539	PureVessel	2020-05-12 17:10:44	<a href="#">Problem D. Slime and Biscuits</a> ***** Why is this problem so hard Life is hard.	Life is hard Answered by 300iq
-------	------------	------------------------	--	-----------------------------------

Life is undecidable. You just don't know what you are going to get from this box of chocolates. Experienced through all the Black Swan and Grey Rhino events happening those years, we all wonder, what's going to happen next? What will the future be like? *"C'est la vie", say the old folks. It goes to show you never can tell.*

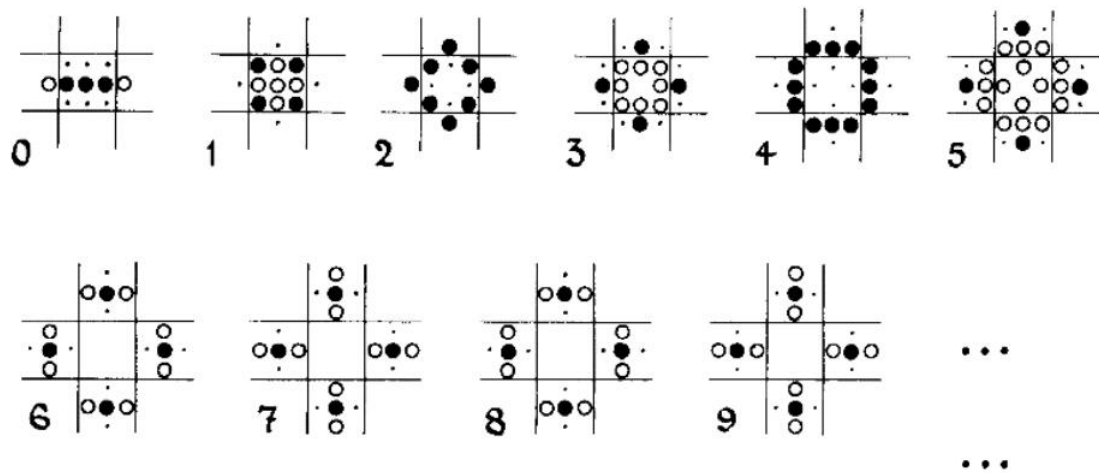
But, life just goes on. We have wines to drink, songs to sing, problems to solve, and games to play. The game we will play now is, **Game of Life**, or simply, **Life**, devised by the famous British mathematician John Horton Conway in 1970.

Life is a zero-player "game" played on an **infinite** squared board. At any time, some of the cells will be **live** and others **dead**. Which cells are live at generation 0 is up to you! But then you've got nothing else to do because the state at any later time is determined by the previous one by the following rules:

- **BIRTH.** A cell that's *dead* at generation  $t$  becomes *live* at generation  $t + 1$  if and only if *exactly three* of its eight neighbors were live at generation  $t$ .
- **DEATH by overcrowding.** A cell that's *live* at generation  $t$  becomes *dead* at generation  $t + 1$  if *at least four* of its eight neighbors were live at generation  $t$ .
- **DEATH by exposure.** A cell that's *live* at generation  $t$  becomes *dead* at generation  $t + 1$  if *at most one* of its eight neighbors were live at generation  $t$ .
- **Survival.** A cell that's *live* at generation  $t$  is still *live* at generation  $t + 1$  if *two or three* of its eight neighbors were live at generation  $t$ .

The rule above can be summarized and memorized as "Just 3 for birth, 2 or 3 for survival."

The following picture is an example of starting from a configuration where only five cells in a line are live. This will end up forming a "traffic light" pattern that repeats **infinitely** with a period of 2. In the picture, filled circles represent *live* cells that will still be *live* at the next generation. Unfilled circles represent *live* cells that will be *dead* at the next generation. Dots represent *dead* cells that will be *live* at the next generation. (Credit: All pictures below are from *Winning Ways For Your Mathematical Plays, Volume 4* by Elwyn R. Berlekamp and John H. Conway).



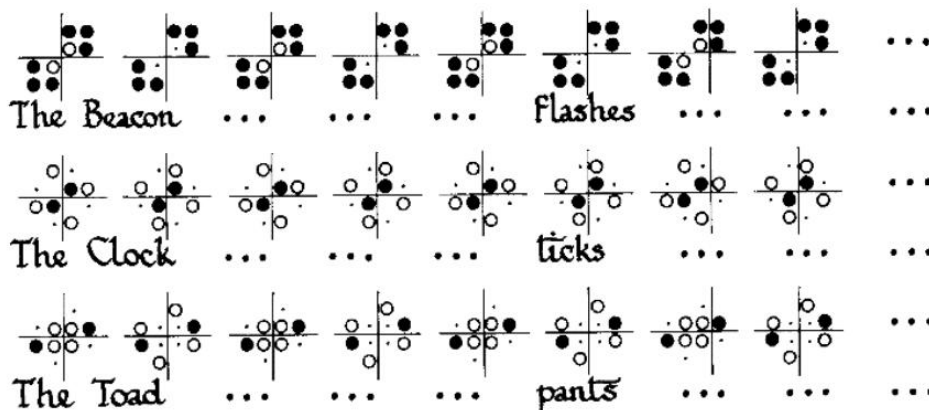
A Line of five becomes traffic lights.

There are some other examples of starting configurations with different "destinies":

Time	0	1	2	3	...	
(a)					...	A Blinker
(b)					...	A Blinker
(c)					...	A Block

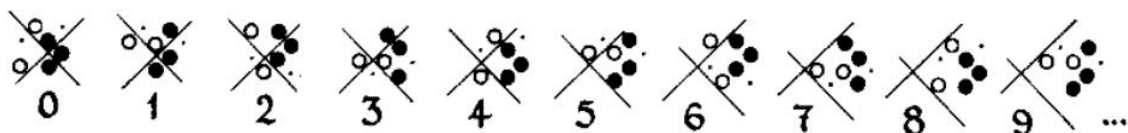
Different start, different destiny.

There are examples where one ends up in a life cycle,



Three life cycles with period 2.

and an example of a moving "glider".



A glider moving diagonally.

Mathematicians and theoretical computer scientists are fascinated by the mystery of Life. They ask the following question: “Given a starting configuration of Life, can we decide if it will eventually die out? i.e., is there some finite number  $t$  such that there is no live cells at generation  $t$ ?” However, it turns out this problem is not only **NP-hard**, but even **undecidable**. It is possible to build a pattern in Life that acts like a finite-state machine connected to two counters. This has the same computational power as a universal Turing machine, so Life is theoretically as powerful as any computer with unlimited memory and no time constraints; it is Turing complete. In fact, several different programmable computer architectures have been implemented in Life, including a pattern that simulates Tetris.

Although as the above suggests, we can never unravel the mystery of Life, there are still certain things we can do. Construction is just one of those things, and is also exactly what this problem asks for:

Given a positive integer  $k$  ( $1 \leq k \leq 100$ ), construct a starting configuration of Game of Life, with all live cells having **positive** coordinates not exceeding 300, so that the configuration lasts **exactly**  $k$  generations, that is, there exist live cells at generation  $k - 1$ , but no live cells at generation  $k$ . It is guaranteed that at least one such starting configuration exists.

## Input

The input consists of only one integer  $k$  ( $1 \leq k \leq 100$ ), denoting the number of generations the configuration need to last.

## Output

The first line contains an integer  $n$ , ( $1 \leq n \leq 90000$ ), denoting the number of live cells in your starting configuration.

Then  $n$  lines follow, each line containing two integers  $x, y$  ( $1 \leq x, y \leq 300$ ), denoting the coordinates of a live cell in your starting configuration. The coordinates of each live cell should be output exactly once.

If there are many possible starting configurations, outputting any of them will be considered correct.

## Examples

standard input	standard output
1	1 1 1
2	3 100 100 101 100 102 99

## Problem I. Mental Abuse To Humans

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

Bobo is asked to solve the following math problem:

Given  $m$  pairs of integers  $(x_1, t_1), (x_2, t_2), \dots, (x_m, t_m)$  where  $0 \leq x_i < n$  and  $t_i \in \{0, 1\}$ , count the number of subsets  $A \subseteq S_n = \{0, 1, \dots, n-1\}$  satisfying

- $\forall 1 \leq i \leq m, x_i \in A$  if and only if  $t_i = 1$ .
- $A +_n (S_n \setminus A) = S_n$ .

For any two sets  $A, B$  consisting of nonnegative integers,  $A +_n B$  is defined by

$$A +_n B = \{(a + b) \bmod n \mid a \in A, b \in B\}.$$

The answer might be enormous, and you should output the answer modulo 998 244 353.

### Input

Each test contains multiple test cases. The first line contains an integer  $T$  ( $1 \leq T \leq 5$ ) — the number of test cases. The following lines contain the description of each test case.

The first line of each test case contains two integers  $n$  and  $m$  ( $1 \leq n \leq 10^{18}$ ,  $0 \leq m \leq 5$ ), denoting the size of the universe and the number of integer pairs, respectively.

Then  $m$  lines follow. The  $i$ -th line contains two integers  $x_i, t_i$  ( $0 \leq x_i < n, t_i \in \{0, 1\}$ ), describing the  $i$ -th integer pair.

It is guaranteed that  $x_1, x_2, \dots, x_m$  are pairwise distinct.

### Output

For each test case, output a single integer, denoting the desired answer modulo 998 244 353.

### Example

standard input	standard output
2	0
3 0	4
6 2	
0 0	
1 1	



## Problem J. Middle Race

Input file:            `standard input`  
Output file:          `standard output`  
Time limit:           2 seconds  
Memory limit:        1024 megabytes

This is an interactive problem.

Middle Race is a board game featured in the game show “The Genius”. In Middle Race, several players will run a race on the board, avoiding finishing the first or the last. In this problem, you are required to play a simplified version of this game versus BoBo and oBoB.

This game consists of  $n$  rounds. Before the game starts, three integer parameters  $A$ ,  $B$ , and  $C$  will be presented. Three items with values  $A$ ,  $B$ , and  $C$ , respectively, will be given in each round. You, BoBo, and oBoB will choose to take one item among the remaining ones **in order**.

Let  $X, Y, Z$  be the total values of the items you, BoBo, and oBoB take in all rounds, respectively. You win the game if and only if  $\min(Y, Z) \leq X \leq \max(Y, Z)$ .

To show that you are smarter than BoBo, please win the game or determine that there exists no winning strategy, i.e., you have no chance of winning if BoBo and oBoB are taking optimal moves.

### Interaction Protocol

Each test run contains multiple test cases. You should first read a line with an integer  $T$  ( $1 \leq T \leq 10^5$ ), representing the number of test cases.

For each test case, you begin the interaction by reading integers  $n, A, B, C$  ( $1 \leq n, A, B, C \leq 10^5$ ) in a line, denoting the number of rounds and parameters of the game.

If no winning strategy exists under the given input, output  $-1$  in a separate line, then continue processing the next test case, if any.

Otherwise, you should complete  $n$  rounds of the game. In each round, you should first output an integer  $x \in \{A, B, C\}$  in one line, indicating that you are going to take an item with value  $x$  in this round. If you perform an invalid move, you will read  $-1 -1$  in the next separate line, and you must terminate your program immediately to receive the correct verdict. Otherwise, two integers  $y, z$  will be given in the next line, indicating that BoBo takes an item with value  $y$  and oBoB takes the one with value  $z$  in this round. After completing all  $n$  rounds, please continue processing the next test case, if any.

It is guaranteed that the sum of  $n$  over all test cases in a single run will not exceed  $10^5$ .

After printing an integer, do not forget to output end of line and flush the output. Otherwise, you may not get the correct verdict. To flush the output, use:

- `fflush(stdout)` or `cout.flush()` in C/C++;
- `System.out.flush()` in Java;
- `stdout.flush()` in Python.

## Example

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

## Note

Empty lines are for better understanding, and you must not output empty lines in your program.

## Problem K. Pattern Matching in A Minor “Low Space”

Input file:            standard input  
Output file:          standard output  
Time limit:           5 seconds  
Memory limit:        1024 megabytes

Bobo is asked to solve the following classical pattern matching problem:

Given two strings  $s$  and  $t$ , calculate how many times  $s$  appears in  $t$  as a substring?

### Input

The first line contains two integers  $n, m$  ( $1 \leq n, m \leq 10^7$ ), denoting the length of strings  $s$  and  $t$ , respectively.

The next line contains a string  $s$  of length  $n$ , consisting of only lowercase English letters.

The next line contains a string  $t$  of length  $m$ , consisting of only lowercase English letters.

**Please refer to the “Note” section for more specific restrictions.**

### Output

Output an integer in a line, denoting the number of times  $s$  appears in  $t$  as a substring.

### Examples

standard input	standard output
3 7 aba abababc	2
1 11 a abracadabra	5
8 10 possible impossible	1
10 21 pleasenote theunusualmemorylimit	0

### Note

Due to technical issues, this problem is implemented as an interactive problem. You can safely treat it as a normal traditional problem, except that your program is only allowed to read the input sequentially at most once. For example, it is explicitly prohibited to read the entire input, then rewind the input stream to the beginning and re-read the input. **Also, you should read all input before outputting the answer, otherwise you may not get the correct verdict.**

## Problem L. Por Una Cabeza

Input file:           standard input  
Output file:         standard output  
Time limit:          2 seconds  
Memory limit:       1024 megabytes

Bobo is the director of the famous TV Show “Sunday Night Live”. The show recently introduced a new stage named “World Debate”, which has been widely acclaimed since its first appearance. Despite its possibly daunting name, it is a stage where the audience votes on some particular topic. Specifically, the show staff will choose a controversial topic, and the audience will vote 0 (meaning “not agree”) or 1 (meaning “agree”) towards it. These votes will then be collected to produce a final opinion on this topic.

What makes this stage particularly stand out from the rest is the use of *voting machines*. Voting machines are automatic systems that receive an odd number of inputs in  $\{0, 1\}$  and take the **majority** of input received as the output. **Note that it is not necessary for the voting machine to receive all inputs to produce an output.** Once a voting machine receives an amount of a same value that is more than half of its input slots, it will immediately produce the value as an output.

The show staff will then use these voting machines to produce a result. Suppose there are  $n$  audience numbered from 1 to  $n$ , and  $m$  voting machines numbered from 1 to  $m$ . The final opinion will be produced in the following manner:

- All audience votes in sequential order from 1 to  $n$ .
- Each voting machine takes its input from either some audience’s vote or the output of a voting machine **with a number smaller than it**.
- The vote of each audience and the output of each voting machine except the one numbered  $m$  is taken as the input of **exactly one** voting machine.
- The output of the voting machine numbered  $m$  will be taken as the final opinion.

As the director, Bobo does not really care about the final result. All he cares about is the rating of the show. He wants to create an intense state called “Por Una Cabeza” so that **for every voting machine, it doesn’t produce an output until it receives all its inputs**. This may not be possible if all audience votes by their own will, but Bobo can make a difference. Bobo knows the vote  $a_i$  of the audience numbered  $i$  ( $1 \leq i \leq n$ ), and the cost  $b_i$  Bobo needs to pay to make the audience change his/her vote to the other side. Under a quite tight budget, Bobo wants to know the minimum cost to achieve “Por Una Cabeza”.

But wait! The TV show is live, so situations might change. The  $a_i$  and  $b_i$  may undergo a total of  $q$  **permanent** changes, and Bobo needs to know the minimum cost to achieve “Por Una Cabeza” after each change. Bobo has put too much effort into this TV show and is too tired. Can you help him?

### Input

The first line contains two integers  $n, m, q$  ( $1 \leq n, m, q \leq 10^5$ ), denoting the number of audiences, voting machines, and changes, respectively.

The  $n$  lines follow. The  $i$ -th ( $1 \leq i \leq n$ ) line contains two integers  $a_i, b_i$  ( $a_i \in \{0, 1\}, 1 \leq b_i \leq 10^9$ ), denoting the votes and the cost needed to change the vote of the  $i$ -th audience.

Then the description of the input of  $m$  voting machines follows, with order sequentially from 1 to  $m$ . The input of each voting machine is described by a line of the form  $\ell, c_1, c_2, \dots, c_\ell$  ( $1 \leq \ell \leq n + m - 1, -m \leq c_i \leq n, c_i \neq 0, c_i$  is pairwise distinct,  $\ell$  is odd), where  $c_i > 0$  means the vote of audience numbered  $c_i$  is taken as an input of the current voting machine, and  $c_i < 0$  means the output of voting machine numbered  $-c_i$  is taken as an input of the current voting machine.

The  $q$  lines describing the changes follow. Each line is of the form  $x, y, z$  ( $1 \leq x \leq n, y \in \{0, 1\}, 1 \leq z \leq 10^9$ ), meaning  $a_x$  will be changed to  $y$  and  $b_x$  will be changed  $z$ . The change is **permanent**, meaning it will not be immediately rolled back after the query.

It is guaranteed each voting machine takes its input from either some audience's vote or the output of a voting machine **with a number smaller than it**.

It is guaranteed that the vote of each audience and the output of each voting machine except the one numbered  $m$  is taken as the input of **exactly one** voting machine.

## Output

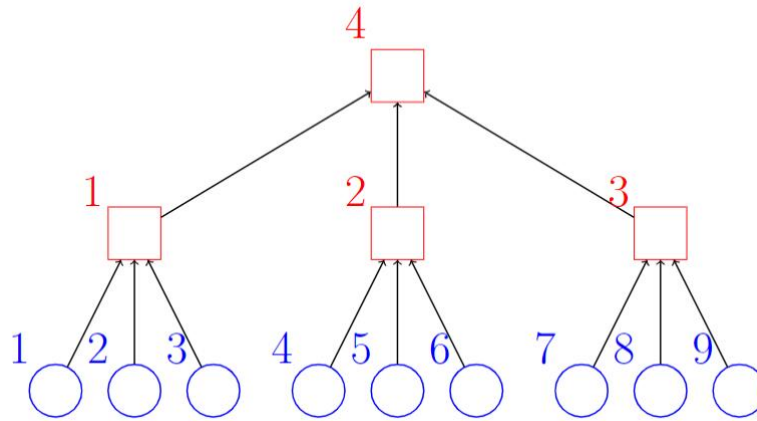
After each of the  $q$  changes, output a number in a line, denoting the minimum cost for Bobo to achieve "Por Una Cabeza". It can be proven it is always possible for Bobo to achieve "Por Una Cabeza" under the given input constraints.

## Examples

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

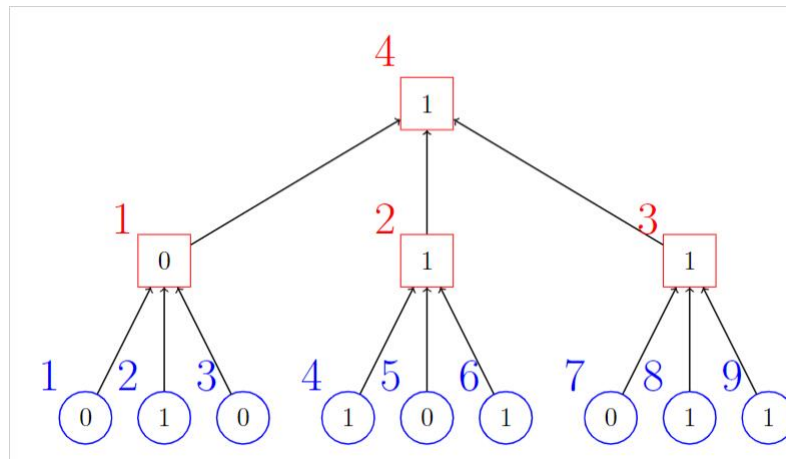
## Note

Here is an example of the structure formed by the audience and voting machines in the second sample test, where circles represent the audience and the squares represent the voting machines, both labeled with corresponding numbers, and the output from the start of an arrow is given as the input of the end of the arrow. Note that the output of machine number 4 will be produced as the final result.



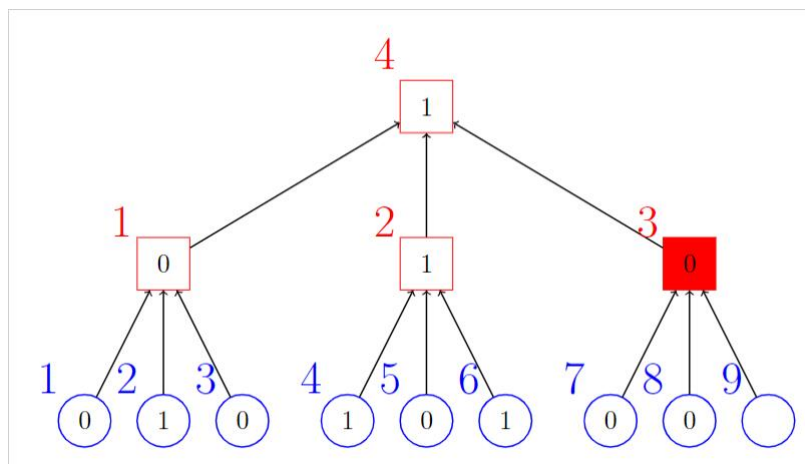
the structure in the second sample test

After the first change, the result is given in the following picture. Here for every voting machine, it doesn't produce an output until it receives all its inputs. So "Por Una Cabeza" is already achieved and there's no need to change the vote of any person and minimum cost is therefore, 0.



the second sample test, after the first change

After the second change, the result is given in the following picture. Here after the 8-th person voted, the outcome of the third voting machine is already decided, without receiving all its inputs, so "Por Una Cabeza" is not achieved and it is optimal for Bobo to change the vote of the 8-th person, which costs 6 exactly after this change.

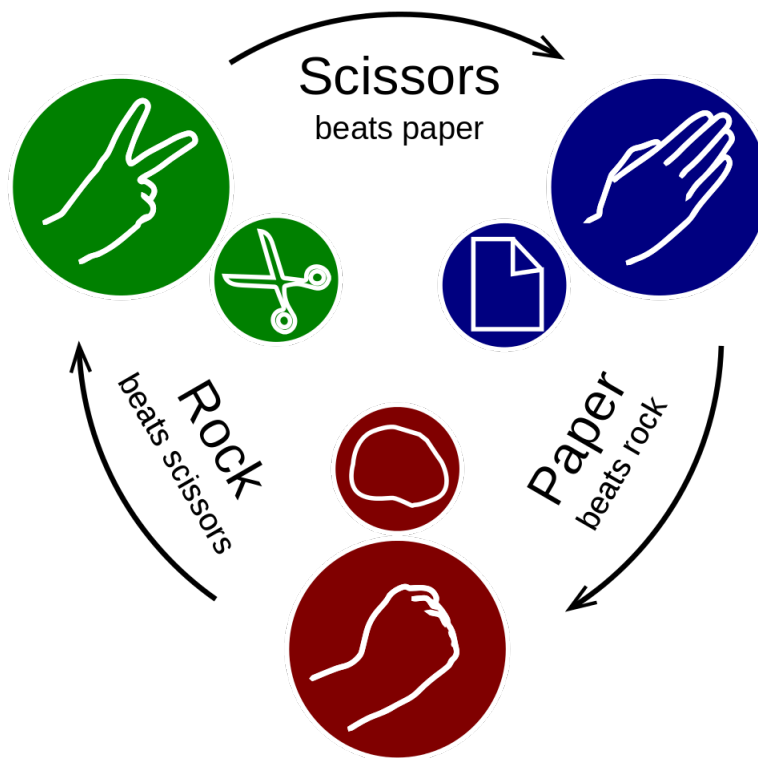


the second sample test, after the second change

## Problem M. Rock-Paper-Scissors Pyramid

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

The rock-paper-scissors is a well-known hand game that originated in China, usually played between two people, in which each player simultaneously forms one of three shapes with an outstretched hand. These shapes are “rock” (a closed fist), “paper” (a flat hand), and “scissors” (a fist with the index finger and middle finger extended, forming a V). “Scissors” is identical to the two-fingered V sign (also indicating “victory” or “peace”) except that it is pointed horizontally instead of being held upright in the air. The rule is simple. Paper beats rock. Rock beats scissors. Scissors beats paper.



Example of Rock Paper Scissors. Source: Wikipedia

We use the uppercase letters R, P and S to represent rock, paper and scissors, respectively. Given an initial string  $s$  containing R, P and S of length  $n$ . Bobo designs a “rock-paper-scissors” pyramid of height and base length  $n$  as follows: Place the initial string  $s$  placed at the bottom  $n$  blocks in order, then the pyramid evolves by the following rule:

- If the two blocks immediately below a block have the same shape (namely, both are R, P, or S), we will place **the same shape** on the block.
- If the two blocks immediately below a block have different shapes, we will place **the winning shape** on the block.

Refer to the pictures in the notes section for a clearer illustration.

Bobo wants to know, following this rule, what is the shape at the top of the pyramid?

### Input

The first line contains an integer  $T$ , denoting the number of test cases.

For each test case, a string  $s(1 \leq |s| \leq 10^6)$  containing only symbols 'R', 'P' and 'S' is given as the configuration on the bottom of the pyramid.

It is the guaranteed that the sum of lengths over all strings doesn't exceed  $10^6$ .

Output

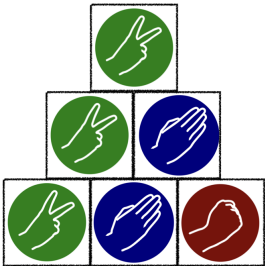
For each test case, output a character  $c \in \{ 'R', 'P', 'S' \}$  in a line, denoting the shape at the top of the pyramid.

Example

standard input	standard output
2 SPR SPSRRP	S P

Note

The following picture gives an example with initial string SPR, as in the first test case of the sample test.



The following picture gives an example with initial string SPSRRP, as in the second test case of the sample test.

