Problem A. Access Levels

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

BerSoft is the biggest IT corporation in Berland, and Monocarp is the head of its security department. This time, he faced the most difficult task ever.

Basically, there are n developers working at BerSoft, numbered from 1 to n. There are m documents shared on the internal network, numbered from 1 to m. There is a table of access requirements a such that $a_{i,j}$ (the j-th element of the i-th row) is 1 if the i-th developer should have access to the j-th document, and 0 if they should have no access to it.

In order to restrict the access, Monocarp is going to perform the following actions:

- choose the number of access groups $k \geq 1$;
- assign each document an access group (an integer from 1 to k) and the required access level (an integer from 1 to 10^9);
- assign each developer k integer values (from 1 to 10^9) — their access levels for each of the access groups.

The developer i has access to the document j if their access level for the access group of the document is greater than or equal to the required access level of the document.

What's the smallest number of access groups Monocarp can choose so that it's possible to assign access groups and access levels in order to satisfy the table of access requirements?

Input

The first line contains two integers n and m ($1 \leq n, m \leq 500$) — the number of developers and the number of documents.

Each of the next n lines contains a binary string of length m — the table of access requirements. The j-th element of the i-th row is 1 if the i-th developer should have access to the j-th document, and 0 if they should have no access to it.

Output

The first line should contain a single integer k — the smallest number of access groups Monocarp can choose so that it's possible to assign access groups and access levels in order to satisfy the table of access requirements.

The second line should contain m integers from 1 to k — the access groups of the documents.

The third line should contain m integers from 1 to 10^9 — the required access levels of the documents.

The i-th of the next n lines should contain k integers from 1 to 10^9 — the access level of the i-th developer on each of the access groups.

If there are multiple solutions, print any of them.

Examples

standard input	standard output
3 2	2
01	1 2
11	2 2
10	1 2
	2 2
	2 1
2 3	1
101	1 1 1
100	1 10 5
	8
	3

Note

In the first example, we assign the documents to different access groups. Both documents have level 2 in their access group. This way, we can assign the developers, who need the access, level 2, and the developers, who have to have no access, level 1.

If they had the same access group, it would be impossible to assign access levels to developers 1 and 3. Developer 1 should've had a lower level than developer 3 in this group to not be able to access document 1. At the same time, developer 3 should've had a lower level than developer 1 in this group to not be able to access document 2. Since they can't both have lower level than each other, it's impossible to have only one access group.

In the second example, it's possible to assign all documents to the same access group.

Problem B. Broken Keyboard

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Recently, Mishka started noticing that his keyboard malfunctions — maybe it's because he was playing rhythm games too much. Empirically, Mishka has found out that every other time he presses a key, it is registered as if the key was pressed twice. For example, if Mishka types text, the first time he presses a key, exactly one letter is printed; the second time he presses a key, two same letters are printed; the third time he presses a key, one letter is printed; the fourth time he presses a key, two same letters are printed, and so on. Note that the number of times a key was pressed is counted for the whole keyboard, not for each key separately. For example, if Mishka tries to type the word `osu`, it will be printed on the screen as `ossu`.

You are given a word consisting of n lowercase Latin letters. You have to determine if it can be printed on Mishka's keyboard or not. You may assume that Mishka cannot delete letters from the word, and every time he presses a key, the new letter (or letters) is appended to the end of the word.

Input

The first line of the input contains one integer t ($1 \le t \le 100$) — the number of test cases.

The first line of the test case contains one integer n ($1 \le n \le 100$) — the length of the word.

The second line of the test case contains a string s consisting of n lowercase Latin letters — the word that should be checked.

Output

For each test case, print `YES` if the word s can be printed on Mishka's keyboard, and `NO` otherwise.

Example

standard input	standard output
4 4 ossu 2 aa 6 addonn 3 qwe	YES NO YES NO

Note

In the first test case, Mishka can type the word as follows: press `o` (one letter `o` appears at the end of the word), then presses `s` (two letters `s` appear at the end of the word), and, finally, press `u` (one letter appears at the end of the word, making the resulting word `ossu`).

In the second test case, Mishka can try typing the word as follows: press `a` (one letter `a` appears at the end of the word). But if he tries to press `a` one more time, two letters `a` will appear at the end of the word, so it is impossible to print the word using his keyboard.

In the fourth test case, Mishka has to start by pressing `q`. Then, if he presses `w`, two copies of `w` will appear at the end of the word, but the third letter should be `e` instead of `w`, so the answer is `NO`.

Problem C. Card Guessing

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

Consider a deck of cards. Each card has one of 4 suits, and there are exactly n cards for each suit — so, the total number of cards in the deck is $4n$. The deck is shuffled randomly so that each of $(4n)!$ possible orders of cards in the deck has the same probability of being the result of shuffling. Let c_i be the i-th card of the deck (from top to bottom).

Monocarp starts drawing the cards from the deck one by one. Before drawing a card, he tries to guess its suit. Monocarp remembers the suits of the k last cards, and his guess is the suit that appeared the least often among the last k cards he has drawn. So, while drawing the i-th card, Monocarp guesses that its suit is the suit that appears the minimum number of times among the cards $c_{i-k}, c_{i-k+1}, \dots, c_{i-1}$ (if $i \le k$, Monocarp considers all previously drawn cards, that is, the cards $c_1, c_2, \dots, c_{i-1}$). If there are multiple suits that appeared the minimum number of times among the previous cards Monocarp remembers, he chooses a random suit out of those for his guess (all suits that appeared the minimum number of times have the same probability of being chosen).

After making a guess, Monocarp draws a card and compares its suit to his guess. If they match, then his guess was correct; otherwise it was incorrect.

Your task is to calculate the expected number of correct guesses Monocarp makes after drawing all $4n$ cards from the deck.

Input

The first (and only) line contains two integers n ($1 \le n \le 500$) and k ($1 \le k \le 4 \cdot n$).

Output

Let the expected value you have to calculate be an irreducible fraction $\frac{x}{y}$. Print one integer — the value of $x \cdot y^{-1} \bmod 998244353$, where y^{-1} is the inverse to y (i. e. an integer such that $y \cdot y^{-1} \bmod 998244353 = 1$).

Examples

standard input	standard output
`1 1`	`748683266`
`3 2`	`567184295`
`2 7`	`373153250`
`2 8`	`373153250`

Problem D. Watch the Videos

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Monocarp wants to watch n videos. Each video is only one minute long, but its size may be arbitrary. The i-th video has the size a_i megabytes. All videos are published on the Internet. A video should be downloaded before it can be watched. Monocarp has poor Internet connection — it takes exactly 1 minute to download 1 megabyte of data, so it will require a_i minutes to download the i-th video.

Monocarp's computer has a hard disk of m megabytes. The disk is used to store the downloaded videos. Once Monocarp starts the download of a video of size s, the s megabytes are immediately reserved on a hard disk. If there are less than s megabytes left, the download cannot be started until the required space is freed. Each single video can be stored on the hard disk, since $a_i \le m$ for all i. Once the download is started, it cannot be interrupted. It is not allowed to run two or more downloads in parallel.

Once a video is fully downloaded to the hard disk, Monocarp can watch it. Watching each video takes exactly 1 minute and does not occupy the Internet connection, so Monocarp can start downloading another video while watching the current one.

When Monocarp finishes watching a video, he doesn't need it on the hard disk anymore, so he can delete the video, instantly freeing the space it occupied on a hard disk. Deleting a video takes negligible time.

Monocarp wants to watch all n videos as quickly as possible. The order of watching does not matter, since Monocarp needs to watch all of them anyway. Please calculate the minimum possible time required for that.

Input

The first line contains two integers n and m ($1 \le n \le 2 \cdot 10^5$; $1 \le m \le 10^9$) — the number of videos Monocarp wants to watch and the size of the hard disk, respectively.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le m$) — the sizes of the videos.

Output

Print one integer — the minimum time required to watch all n videos.

Examples

standard input	standard output
5 6 1 2 3 4 5	16
5 5 1 2 3 4 5	17
4 3 1 3 2 3	12

Problem E. Exchange

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Monocarp is playing a MMORPG. There are two commonly used types of currency in this MMORPG — gold coins and silver coins. Monocarp wants to buy a new weapon for his character, and that weapon costs n silver coins. Unfortunately, right now, Monocarp has no coins at all.

Monocarp can earn gold coins by completing quests in the game. Each quest yields exactly one gold coin. Monocarp can also exchange coins via the in-game trading system. Monocarp has spent days analyzing the in-game economy; he came to the following conclusion: it is possible to sell one gold coin for a silver coins (i. e. Monocarp can lose one gold coin to gain a silver coins), or buy one gold coin for b silver coins (i. e. Monocarp can lose b silver coins to gain one gold coin).

Now Monocarp wants to calculate the minimum number of quests that he has to complete in order to have at least n silver coins after some abuse of the in-game economy. Note that Monocarp can perform exchanges of both types (selling and buying gold coins for silver coins) any number of times.

Input

The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases.

Each test case consists of one line containing three integers n, a and b ($1 \le n \le 10^7$; $1 \le a, b \le 50$).

Output

For each test case, print one integer — the minimum possible number of quests Monocarp has to complete.

Example

standard input	standard output

4
100 25 30
9999997 25 50
52 50 48
49 50 1
	4
	400000
	1
	1

Note

In the first test case of the example, Monocarp should complete 4 quests, and then sell 4 gold coins for 100 silver coins.

In the second test case, Monocarp should complete 400000 quests, and then sell 400000 gold coins for 10 million silver coins.

In the third test case, Monocarp should complete 1 quest, sell the gold coin for 50 silver coins, buy a gold coin for 48 silver coins, and then sell it again for 50 coins. So, he will have 52 silver coins.

In the fourth test case, Monocarp should complete 1 quest and then sell the gold coin he has obtained for 50 silver coins.

Problem F. Chemistry Lab

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Monocarp is planning on opening a chemistry lab. During the first month, he's going to distribute solutions of a certain acid.

First, he will sign some contracts with a local chemistry factory. Each contract provides Monocarp with an unlimited supply of some solution of the same acid. The factory provides n contract options, numbered from 1 to n. The i-th solution has a concentration of $x_i\%$, the contract costs w_i burles, and Monocarp will be able to sell it for c_i burles per liter.

Monocarp is expecting k customers during the first month. Each customer will buy a liter of a $y\%$-solution, where y is a **real** number chosen uniformly at random from 0 to 100 independently for each customer. More formally, the probability of number y being less than or equal to some t is $P(y \le t) = \frac{t}{100}$.

Monocarp can mix the solution that he signed the contracts with the factory for, at any ratio. More formally, if he has contracts for m solutions with concentrations $x_1, x_2, \dots, x_m$, then, for these solutions, he picks their volumes $a_1, a_2, \dots, a_m$ so that $\sum\limits_{i=1}^{m} a_i = 1$ (exactly 1 since each customer wants exactly one liter of a certain solution).

The concentration of the resulting solution is $\sum\limits_{i=1}^{m} x_i \cdot a_i$. The price of the resulting solution is $\sum\limits_{i=1}^{m} c_i \cdot a_i$.

If Monocarp can obtain a solution of concentration $y\%$, then he will do it while maximizing its price (the cost for the customer). Otherwise, the customer leaves without buying anything, and the price is considered equal to 0.

Monocarp wants to sign some contracts with a factory (possibly, none or all of them) so that the expected profit is maximized — the expected total price of the sold solutions for all k customers minus the total cost of signing the contracts from the factory.

Print the maximum expected profit Monocarp can achieve.

Input

The first line contains two integers n and k ($1 \le n \le 5000$; $1 \le k \le 10^5$) — the number of contracts the factory provides and the number of customers.

The i-th of the next n lines contains three integers x_i, w_i and c_i ($0 \le x_i \le 100$; $1 \le w_i \le 10^9$; $1 \le c_i \le 10^5$) — the concentration of the solution, the cost of the contract and the cost per liter for the customer, for the i-th contract.

Output

Print a single real number — the maximum expected profit Monocarp can achieve.

Your answer is 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 is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Examples

standard input	standard output

2 10
0 10 20
100 15 20
	175.000000000000000

2 10
0 100 20
100 150 20
	0.000000000000000

6 15
79 5 35
30 13 132
37 3 52
24 2 60
76 18 14
71 17 7
	680.125000000000000

10 15
46 11 11
4 12 170
69 2 130
2 8 72
82 7 117
100 5 154
38 9 146
97 1 132
0 12 82
53 1 144
	2379.400000000000000

Problem G. Guess the String

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 megabytes

This is an interactive problem. You have to use `flush` operation right after printing each line. For example, in C++ you should use the function `fflush(stdout)`, in Java or Kotlin — `System.out.flush()`, and in Python — `sys.stdout.flush()`.

The jury has a string s consisting of characters `0` and/or `1`. The first character of this string is `0`. The length of this string is n. You have to guess this string. Let's denote $s[l..r]$ as the substring of s from l to r (i. e. $s[l..r]$ is the string $s_l s_{l+1} \dots s_r$).

Let the *prefix function* of the string s be an array $[p_1, p_2, \dots, p_n]$, where p_i is the greatest integer $j \in [0, i-1]$ such that $s[1..j] = s[i-j+1..i]$. Also, let the *antiprefix function* of the string s be an array $[q_1, q_2, \dots, q_n]$, where q_i is the greatest integer $j \in [0, i-1]$ such that $s[1..j]$ differs from $s[i-j+1..i]$ in **every** position.

For example, for the string `011001`, its *prefix function* is $[0, 0, 0, 1, 1, 2]$, and its *antiprefix function* is $[0, 1, 1, 2, 3, 4]$.

You can ask queries of two types to guess the string s:

- 1 i — "what is the value of p_i?";
- 2 i — "what is the value of q_i?".

You have to guess the string by asking no more than 789 queries. Note that giving the answer does not count as a query.

In every test and in every test case, the string s is fixed beforehand.

Interaction Protocol

Initially, the jury program sends one integer t ($1 \le t \le 100$) — the number of test cases.

At the start of each test case, the jury program sends one integer n ($2 \le n \le 1000$) — the length of the string.

After that, your program can submit queries to the jury program by printing one of the following lines **(do not forget to flush the output after printing a line!)**:

- 1 i — the query "what is the value of p_i?";
- 2 i — the query "what is the value of q_i?".

For every query, the jury prints one integer on a separate line. It is either:

- the answer for your query, if the query is correct and you haven't exceeded the query limit;
- or the integer -1, if your query is incorrect (for example, the constraint $1 \le i \le n$ is not met) or if you have asked too many queries while processing the current test case.

To submit the answer, your program should send a line in the following format **(do not forget to flush the output after printing a line!)**:

- 0 s, where s is a sequence of n characters `0` and/or `1`.

If your guess is correct, the jury program will print one integer 1 on a separate line, indicating that you may proceed to the next test case (or terminate the program, if it was the last test case) and that the number of queries you have asked is reset. If it is not correct, the jury program will print one integer -1 on a separate line.

After your program receives -1 as the answer, it should immediately terminate. This will lead to your submission receiving the verdict "Wrong Answer". If your program does not terminate, the verdict of your submission is undefined.

Example

standard input	standard output
`2 // 2 test cases`	
`6 // n = 6`	
	`1 3      // what is p[3]?`
`0 // p[3] = 0`	
	`2 2      // what is q[2]?`
`1 // q[2] = 1`	
	`2 6      // what is q[6]?`
`4 // q[6] = 4`	
	`1 4      // what is p[4]?`
`1 // p[4] = 1`	
	`0 011001 // the guess is 011001`
`1 // answer is correct`	
`5 // n = 5`	
	`1 2      // what is p[2]?`
`1 // p[2] = 1`	
	`2 4      // what is q[4]?`
`2 // q[4] = 2`	
	`2 5      // what is q[5]?`
`2 // q[5] = 2`	
	`0 00111  // the guess is 00111`
`1 // answer is correct`	

Note

The example contains one possible way of interaction in a test where $t = 2$, and the strings guessed by the jury are `011001` and `00111`. Note that everything after the `//` sign is a comment that explains which line means what in the interaction. **The jury program won't print these comments in the actual problem, and you shouldn't print them**. The empty lines are also added for your convenience, **the jury program won't print them, and your solution should not print any empty lines**.

Problem H. Hospital Queue

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

There are n people (numbered from 1 to n) signed up for a doctor's appointment. The doctor has to choose in which order he will appoint these people. The i-th patient should be appointed among the first p_i people. There are also m restrictions of the following format: the i-th restriction is denoted by two integers (a_i, b_i) and means that the patient with the index a_i should be appointed earlier than the patient with the index b_i.

For example, if $n = 4$, $p = [2, 3, 2, 4]$, $m = 1$, $a = [3]$ and $b = [1]$, then the only order of appointment of patients that does not violate the restrictions is $[3, 1, 2, 4]$. For $n = 3$, $p = [3, 3, 3]$, $m = 0$, $a = []$ and $b = []$, any order of appointment is valid.

For each patient, calculate the minimum position in the order that they can have among all possible orderings that don't violate the restrictions.

Input

The first line contains two integers n and m ($1 \le n \le 2000$; $0 \le m \le 2000$).

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$).

Then m lines follow. The i-th of them contains two integers a_i and b_i ($1 \le a_i, b_i \le n$; $a_i \ne b_i$). All pairs of (a_i, b_i) are distinct (i. e. if $i \ne j$, then either $a_i \ne a_j$, $b_i \ne b_j$, or both).

Additional constraint on the input: there is at least one valid order of patients.

Output

Print n integers, where i-th integer is equal to the minimum position of i-th patient in the order, among all valid orders. Positions in the order are numbered from 1 to n.

Examples

standard input	standard output

4 1
2 3 2 4
3 1
	2 3 1 4

3 0 3 3 3	1 1 1

5 3
4 3 3 2 5
3 1
1 5
4 2
	4 2 1 1 5

Note

In the first example, $[3, 1, 2, 4]$ the only one valid order, so the minimum position of each patient is equal to their position in this order.

In the second example, any order is valid, so any patient can be appointed first.

In the third example, there are three valid orders: $[4, 2, 3, 1, 5]$, $[3, 4, 2, 1, 5]$ and $[4, 3, 2, 1, 5]$.

Problem I. Infinite Chess

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

The black king lives on a chess board with an infinite number of columns (files) and 8 rows (ranks). The columns are numbered with all integer numbers (including negative). The rows are numbered from 1 to 8.

Initially, the black king is located on the starting square (x_s, y_s), and he needs to reach some target square (x_t, y_t). Unfortunately, there are also white pieces on the board, and they threaten the black king. After negotiations, the white pieces agreed to let the black king pass to the target square on the following conditions:

- each turn, the black king makes a move according to the movement rules;
- the black king cannot move to a square occupied by a white piece;
- the black king cannot move to a square which is under attack by any white piece. A square is under attack if a white piece can reach it in one move according to the movement rules;
- the white pieces never move.

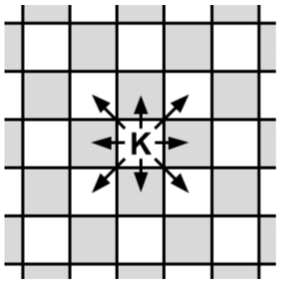
Help the black king find the minimum number of moves needed to reach the target square while not violating the conditions. The black king cannot leave the board at any time.

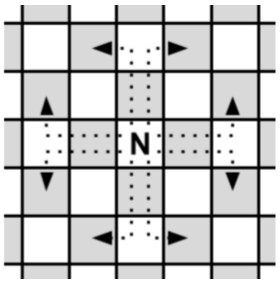
The black king moves according to the movement rules below. Even though the white pieces never move, squares which they can reach in one move are considered to be under attack, so the black king cannot move into those squares.

Below are the movement rules. Note that the pieces (except for the knight) cannot jump over other pieces.

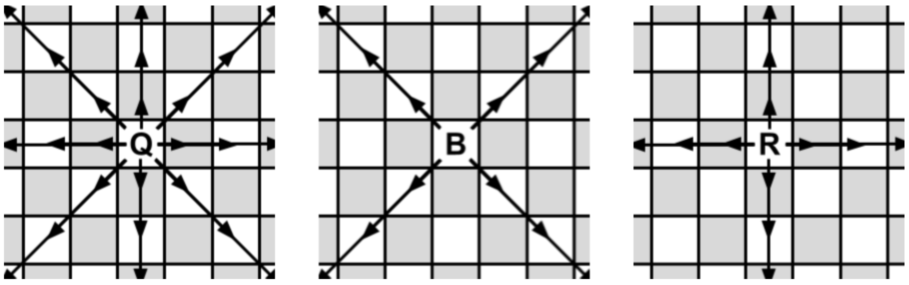
- a king moves exactly one square horizontally, vertically, or diagonally.
- a rook moves any number of vacant squares horizontally or vertically.
- a bishop moves any number of vacant squares diagonally.
- a queen moves any number of vacant squares horizontally, vertically, or diagonally.
- a knight moves to one of the nearest squares not on the same rank, file, or diagonal (this can be thought of as moving two squares horizontally then one square vertically, or moving one square horizontally then two squares vertically — i. e. in an "L" pattern). Knights are not blocked by other pieces, they can simply jump over them.

There are no pawns on the board.





King and knight possible moves, respectively. Dotted line shows that knight can jump over other pieces.



Queen, bishop, and rook possible moves, respectively.

Input

The first line contains two integers x_s and y_s ($1 \le x_s \le 10^8$; $1 \le y_s \le 8$) — the starting coordinates of the black king.

The second line contains two integers x_t and y_t ($1 \le x_t \le 10^8$; $1 \le y_t \le 8$) — the coordinates of the target square for the black king.

The third line contains one integer n ($0 \le n \le 2000$) — the number of white pieces on the board.

Then n lines follow, the i-th line contains one character t_i and two integers x_i and y_i ($1 \le x_i \le 10^8$; $1 \le y_i \le 8$) — the type and the coordinates of the i-th white piece. The types of pieces are represented by the following uppercase Latin letters:

- `K` — king
- `Q` — queen
- `R` — rook
- `B` — bishop
- `N` — knight

There can be any number of white pieces of any type listed above on the board, for example, 3 white kings or 4 white queens. There are no pawns on the board.

Additional constrains on the input:

- no square is occupied by more than one white piece;
- the starting square for the black king is different from the square he wants to reach, and neither of these two squares is occupied or is under attack by any white piece.

Output

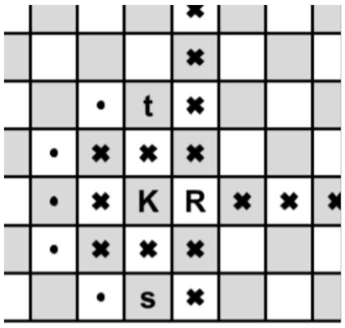
Print one integer — the minimum number of moves needed for the black king to reach the target square while not violating the conditions, or -1 if it is impossible.

Examples

standard input	standard output
1 8 7 8 2 N 4 8 B 4 6	10
1 1 1 5 2 K 1 3 R 2 3	6
2 2 6 4 1 Q 4 3	-1

Note

The image below demonstrates the solution for the second example. Here, the letters `K`, `R`, `s`, and `t` represent the white king, the white rook, the starting square, and the target square, respectively. Bold crosses mark the squares which are under attack by the white pieces. Bold dots show the shortest path for the black king.



Problem J. Hero to Zero

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 megabytes

There are no heroes in this problem. I guess we should have named it "To Zero".

You are given two arrays a and b, each of these arrays contains n non-negative integers.

Let c be a matrix of size $n \times n$ such that $c_{i,j} = |a_i - b_j|$ for every $i \in [1, n]$ and every $j \in [1, n]$.

Your goal is to transform the matrix c so that it becomes the zero matrix, i. e. a matrix where **every** element is **exactly** 0. In order to do so, you may perform the following operations any number of times, in any order:

- choose an integer i, then decrease $c_{i,j}$ by 1 for every $j \in [1, n]$ (i. e. decrease all elements in the i-th row by 1). In order to perform this operation, you **pay** 1 coin;
- choose an integer j, then decrease $c_{i,j}$ by 1 for every $i \in [1, n]$ (i. e. decrease all elements in the j-th column by 1). In order to perform this operation, you **pay** 1 coin;
- choose two integers i and j, then decrease $c_{i,j}$ by 1. In order to perform this operation, you **pay** 1 coin;
- choose an integer i, then increase $c_{i,j}$ by 1 for every $j \in [1, n]$ (i. e. increase all elements in the i-th row by 1). When you perform this operation, you **receive** 1 coin;
- choose an integer j, then increase $c_{i,j}$ by 1 for every $i \in [1, n]$ (i. e. increase all elements in the j-th column by 1). When you perform this operation, you **receive** 1 coin.

You have to calculate the minimum number of coins required to transform the matrix c into the zero matrix. Note that all elements of c should be equal to 0 **simultaneously** after the operations.

Input

The first line contains one integer n ($2 \le n \le 2 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^8$).

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($0 \le b_j \le 10^8$).

Output

Print one integer — the minimum number of coins required to transform the matrix c into the zero matrix.

Examples

standard input	standard output

3
1 2 3
2 2 2
	2

3
3 1 3
1 1 2
	5

2
1 0
2 1
	2

2
1 4
2 3
	4

4
1 3 3 7
6 9 4 2
	29

Note

In the first example, the matrix looks as follows:

$$\begin{matrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{matrix}$$

You can turn it into a zero matrix using 2 coins as follows:

- subtract 1 from the first row, paying 1 coin;
- subtract 1 from the third row, paying 1 coin.

In the second example, the matrix looks as follows:

$$\begin{matrix} 2 & 2 & 1 \\ 0 & 0 & 1 \\ 2 & 2 & 1 \end{matrix}$$

You can turn it into a zero matrix using 5 coins as follows:

- subtract 1 from the first row, paying 1 coin;
- subtract 1 from the third row, paying 1 coin;
- subtract 1 from the third row, paying 1 coin;
- subtract 1 from $a_{2,3}$, paying 1 coin;
- add 1 to the third column, receiving 1 coin;
- subtract 1 from the first row, paying 1 coin;
- subtract 1 from $a_{2,3}$, paying 1 coin.

Problem K. Torus Path

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

You are given a square grid with n rows and n columns, where each cell has a non-negative integer written in it. There is a chip initially placed at the top left cell (the cell with coordinates $(1, 1)$). You need to move the chip to the bottom right cell (the cell with coordinates (n, n)).

In one step, you can move the chip to the neighboring cell, but:

1. you can move only right or down. In other words, if the current cell is (x, y), you can move either to $(x, y + 1)$ or to $(x + 1, y)$. There are two special cases:
   - if the chip is in the last column (cell (x, n)) and you're moving right, you'll teleport to the first column (to the cell $(x, 1)$);
   - if the chip is in the last row (cell (n, y)) and you're moving down, you'll teleport to the first row (to the cell $(1, y)$).
2. you **cannot** visit the same cell twice. The starting cell is counted visited from the beginning (so you cannot enter it again), and you can't leave the finishing cell once you visit it.

Your total score is counted as the sum of numbers in all cells you have visited. What is the maximum possible score you can achieve?

Input

The first line contains the single integer n ($2 \le n \le 200$) — the number of rows and columns in the grid.

Next n lines contains the description of each row of the grid. The i-th line contains n integers $a_{i,1}, a_{i,2}, \ldots, a_{i,n}$ ($0 \le a_{i,j} \le 10^9$) where $a_{i,j}$ is the number written in the cell (i, j).

Output

Print one integer — the maximum possible score you can achieve.

Examples

standard input	standard output

2
1 2
3 4
	8

3
10 10 10
10 0 10
10 10 10
	80

Problem L. Project Manager

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 megabytes

There are n employees at Bersoft company, numbered from 1 to n. Each employee works on some days of the week and rests on the other days. You are given the lists of working days of the week for each employee.

There are regular days and holidays. On regular days, only those employees work that have the current day of the week on their list. On holidays, no one works. You are provided with a list of days that are holidays. The days are numbered from 1 onwards, day 1 is Monday.

The company receives k project offers they have to complete. The projects are numbered from 1 to k in the order of decreasing priority.

Each project consists of multiple parts, where the i-th part must be completed by the a_i-th employee. The parts must be completed in order (i. e. the $(i + 1)$-st part can only be started when the i-th part is completed). Each part takes the corresponding employee a day to complete.

The projects can be worked on simultaneously. However, one employee can complete a part of only one project during a single day. If they have a choice of what project to complete a part on, they always go for the project with the highest priority (the lowest index).

For each project, output the day that project will be completed on.

Input

The first line contains three integers n, m and k ($1 \le n, m, k \le 2 \cdot 10^5$) — the number of employees, the number of holidays and the number of projects.

The i-th of the next n lines contains the list of working days of the i-th employee. First, a single integer t ($1 \le t \le 7$) — the number of working days. Then t days of the week in the increasing order. The possible days are: "`Monday`", "`Tuesday`", "`Wednesday`", "`Thursday`", "`Friday`", "`Saturday`", "`Sunday`".

The next line contains m integers $h_1, h_2, \ldots, h_m$ ($1 \le h_1 < h_2 < \cdots < h_m \le 10^9$) — the list of holidays.

The j-th of the next k lines contains a description of the j-th project. It starts with an integer p ($1 \le p \le 2 \cdot 10^5$) — the number of parts in the project. Then p integers $a_1, a_2, \ldots, a_p$ ($1 \le a_x \le n$) follow, where p_i is the index of the employee that must complete the i-th part.

The total number of parts in all projects doesn't exceed $2 \cdot 10^5$.

Output

Print k integers — the j-th value should be equal to the day the j-th project is completed on.

Example

standard input	standard output

`3 5 4`
`2 Saturday Sunday`
`2 Tuesday Thursday`
`4 Monday Wednesday Friday Saturday`
`4 7 13 14 15`
`5 1 1 3 3 2`
`3 2 3 2`
`5 3 3 3 1 1`
`8 3 3 3 3 3 3 3 3`
	`25 9 27 27`

Problem M. Minimum LCM

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

You are given an integer n.

Your task is to find two positive (greater than 0) integers a and b such that $a + b = n$ and the least common multiple (LCM) of a and b is the minimum among all possible values of a and b. If there are multiple answers, you can print any of them.

Input

The first line contains a single integer t ($1 \le t \le 100$) — the number of test cases.

The first line of each test case contains a single integer n ($2 \le n \le 10^9$).

Output

For each test case, print two positive integers a and b — the answer to the problem. If there are multiple answers, you can print any of them.

Example

standard input	standard output

4
2
9
5
10
	1 1
	3 6
	1 4
	5 5

Note

In the second example, there are 8 possible pairs of a and b:

- $a = 1$, $b = 8$, $LCM(1, 8) = 8$;
- $a = 2$, $b = 7$, $LCM(2, 7) = 14$;
- $a = 3$, $b = 6$, $LCM(3, 6) = 6$;
- $a = 4$, $b = 5$, $LCM(4, 5) = 20$;
- $a = 5$, $b = 4$, $LCM(5, 4) = 20$;
- $a = 6$, $b = 3$, $LCM(6, 3) = 6$;
- $a = 7$, $b = 2$, $LCM(7, 2) = 14$;
- $a = 8$, $b = 1$, $LCM(8, 1) = 8$.

In the third example, there are 5 possible pairs of a and b:

- $a = 1$, $b = 4$, $LCM(1, 4) = 4$;
- $a = 2$, $b = 3$, $LCM(2, 3) = 6$;
- $a = 3$, $b = 2$, $LCM(3, 2) = 6$;
- $a = 4$, $b = 1$, $LCM(4, 1) = 4$.

Problem N. Number Reduction

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

You are given a positive integer x.

You can apply the following operation to the number: remove one occurrence of any digit in such a way that the resulting number **does not contain any leading zeroes** and **is still a positive integer**. For example, 10142 can be converted to 1142, 1042, 1012 or 1014 (note that 0142 is not a valid outcome); 10 can be converted to 1 (but not to 0 since it is not positive).

Your task is to find the minimum positive integer that you can obtain from x if you can apply the aforementioned operation exactly k times.

Input

The first line contains a single integer t ($1 \le t \le 10^5$) — the number of test cases.

The first line of each test case contains a single integer x ($1 \le x < 10^{500000}$).

The second line contains a single integer k ($0 \le k < |x|$), where $|x|$ is the length of the number x.

The sum of $|x|$ over all test cases does not exceed $5 \cdot 10^5$.

Output

For each test case, print one integer — the minimum positive number that you can obtain from x if you can apply the operation exactly k times.

Example

standard input	standard output
5 10000 4 1337 0 987654321 6 66837494128 5 7808652 3	1 1337 321 344128 7052