Problem A. A Very Hard Question

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

Husam was preparing himself for the Graduate Record Examinations (GRE). Yesterday, he read a very hard question, but he could not find a solution for it, so he did not sleep all the night.

Husam decided to tell you about the question, so you can help him to find the solution. The question is: `If the price of the orange was increased by` x`%. How many oranges can be bought for the amount that used to buy` y `oranges?`

Can you help Husam to solve this question?

Input

The first line contains an integer T ($1 \leq T \leq 10^4$), where T is the number of test cases.

Then T lines follow, each line contains two integers y and x ($1 \leq y \leq 10^6$) ($0 \leq x \leq 100$), where y is the number of oranges, and x is the percentage increase in price.

Output

For each test case, print a single line containing the number of oranges that can be bought for the same amount of money that used to buy y oranges before the price increased.

It is guaranteed that all answers are integer numbers. Do not print any floating-point values.

Example

standard input	standard output

3
10 25
300 20
550 100
	8
	250
	275

Problem B. Linear Algebra Test

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Dr. Wail is preparing for today's test in linear algebra course. The test's subject is `Matrices Multiplication`.

Dr. Wail has n matrices, such that the size of the i^{th} matrix is $(a_i \times b_i)$, where a_i is the number of rows in the i^{th} matrix, and b_i is the number of columns in the i^{th} matrix.

Dr. Wail wants to count how many pairs of indices i and j exist, such that he can multiply the i^{th} matrix with the j^{th} matrix.

Dr. Wail can multiply the i^{th} matrix with the j^{th} matrix, if the number of columns in the i^{th} matrix is equal to the number of rows in the j^{th} matrix.

Input

The first line contains an integer T $(1 \leq T \leq 100)$, where T is the number of test cases.

The first line of each test case contains an integer n $(1 \leq n \leq 10^5)$, where n is the number of matrices Dr. Wail has.

Then n lines follow, each line contains two integers a_i and b_i $(1 \leq a_i, b_i \leq 10^9)$ $(a_i \neq b_i)$, where a_i is the number of rows in the i^{th} matrix, and b_i is the number of columns in the i^{th} matrix.

Output

For each test case, print a single line containing how many pairs of indices i and j exist, such that Dr. Wail can multiply the i^{th} matrix with the j^{th} matrix.

Example

standard input	standard output

1
5
2 3
2 3
4 2
3 5
9 4
	5

Note

As input/output can reach huge size it is recommended to use fast input/output methods: for example, prefer to use `scanf/printf` instead of `cin/cout` in C++, prefer to use `BufferedReader/PrintWriter` instead of `Scanner/System.out` in Java.

In the first test case, Dr. Wail can multiply the 1^{st} matrix (2×3) with the 4^{th} matrix (3×5), the 2^{nd} matrix (2×3) with the 4^{th} matrix (3×5), the 3^{rd} matrix (4×2) with the 1^{st} and second matrices (2×3), and the 5^{th} matrix (9×4) with the 3^{rd} matrix (4×2). So, the answer is 5.

Problem C. Ahmad and Spells

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

Ahmad has recently started playing a new game called DotA (Defense of the Agents), which is a very interesting computer game, and currently he is learning a new strategy for winning with his special hero "Chimpanzee King". In order to perform the strategy, the hero must execute n spells.

Initially, each spell needs x seconds of channeling to be executed, but Ahmad knows two ways that can reduce the channeling time, which are:

1. There are m talents (the hero can learn no more than one talent), the i^{th} of them costs b_i mana-points, and changes the channeling time of all spells to a_i instead of x.
2. He can ask for help from other heroes (no more than one) to immediately execute spells. There are k such heroes, and the i^{th} of them costs d_i mana-points, and instantly executes c_i spells.

The total number of mana-points that Ahmad spends should not exceed s.

Ahmad wants to use this strategy at its best, so he is interested in the minimum time he needs to spend in order to execute n spells.

Input

The first line of the input contains an integer T, where T is the number of test cases.

The first line of each test case contains three integers n, m, and k ($1 \leq n \leq 2 \times 10^9, 1 \leq m, k \leq 10^5$), where n is the number of spells Ahmad has to execute, m is the number of talents, and k is the number of heroes.

The second line of each test case contains two integers x and s ($2 \leq x \leq 2 \times 10^9, 1 \leq s \leq 2 \times 10^9$), where x is the initial number of seconds required to channel one spell, and s is the number of mana-points Ahmad can use.

The third line of each test case contains m integers $a_1, a_2, \ldots, a_m$ ($1 \leq a_i \leq x$), where a_i is the number of seconds it will take to prepare one potion of the i^{th} spell from the first type.

The fourth line of each test case contains m integers $b_1, b_2, \ldots, b_m$ ($1 \leq b_i \leq 2 \times 10^9$), where b_i is the number of required mana-points to use the i^{th} spell from the first type.

The fifth line of each test case contains k integers $c_1, c_2, \ldots, c_k$ ($1 \leq c_i \leq n$), where c_i is the number of potions that will be immediately created if the i^{th} spell from the second type was used.

The sixth line of each test case contains k integers $d_1, d_2, \ldots, d_k$ ($1 \leq d_i \leq 2 \times 10^9$), where d_i is the number of mana-points required to use the i^{th} spell from the second type.

Output

For each test case, print a single line containing the minimum time Ahmad has to spend in order to prepare n potions.

Example

standard input	standard output

2
20 3 2
10 99
2 4 3
20 10 40
4 15
10 80
20 3 2
10 99
2 4 3
200 100 400
4 15
100 800
	20
	200

Note

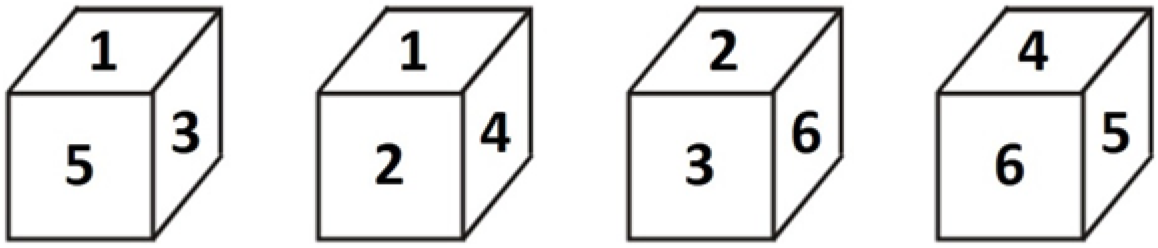
As input/output can reach huge size it is recommended to use fast input/output methods: for example, prefer to use `scanf/printf` instead of `cin/cout` in C++, prefer to use `BufferedReader/PrintWriter` instead of `Scanner/System.out` in Java.

Problem D. Dice Game

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

A dice is a small cube, with each side having a different number of spots on it, ranging from 1 to 6.

Each side in the dice has 4 adjacent sides that can be reached by rotating the dice (i.e. the current side) 90 degrees. The following picture can help you to conclude the adjacent sides for each side in the dice.



In this problem, you are given a dice with the side containing 1 spot facing upwards, and a sum n, your task is to find the minimum number of required moves to reach the given sum.

On each move, you can rotate the dice 90 degrees to get one of the adjacent sides to the side that currently facing upwards, and add the value of the **new side** to your current sum. According to the previous picture, if the side that currently facing upwards contains 1 spot, then in one move you can move to one of sides that contain 2, 3, 4, or 5 spots.

Initially, your current sum is 0. Even though at the beginning the side that containing 1 spot is facing upwards, but its value will not be added to your sum from the beginning, which means that you must make at least one move to start adding values to your current sum.

Input

The first line contains an integer T $(1 \leq T \leq 200)$, where T is the number of test cases.

Then T lines follow, each line contains an integer n $(1 \leq n \leq 10^4)$, where n is the required sum you need to reach.

Output

For each test case, print a single line containing the minimum number of required moves to reach the given sum. If there is no answer, print -1.

Example

standard input	standard output
2 5 10	1 2

Note

In the first test case, you can rotate the dice 90 degrees one time, and make the side that contains 5 spots facing upwards, which make the current sum equal to 5. So, you need one move to reach sum equal to 5.

In the second test case, you can rotate the dice 90 degrees one time, and make the side that contains 4 spots facing upwards, which make the current sum equal to 4. Then rotate the dice another 90 degrees, and make the side that contains 6 spots facing upwards, which make the current sum equal to 10. So, you need two moves to reach sum equal to 10.

Problem E. The Architect Omar

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

Architect Omar is responsible for furnishing the new apartments after completion of its construction. Omar has a set of living room furniture, a set of kitchen furniture, and a set of bedroom furniture, from different manufacturers.

In order to furnish an apartment, Omar needs a living room furniture, a kitchen furniture, and two bedroom furniture, regardless the manufacturer company.

You are given a list of furniture Omar owns, your task is to find the maximum number of apartments that can be furnished by Omar.

Input

The first line contains an integer T ($1 \leq T \leq 100$), where T is the number of test cases.

The first line of each test case contains an integer n ($1 \leq n \leq 1000$), where n is the number of available furniture from all types. Then n lines follow, each line contains a string s representing the name of a furniture.

Each string s begins with the furniture's type, then followed by the manufacturer's name. The furniture's type can be:

- `bed`, which means that the furniture's type is bedroom.
- `kitchen`, which means that the furniture's type is kitchen.
- `living`, which means that the furniture's type is living room.

All strings are non-empty consisting of lowercase and uppercase English letters, and digits. The length of each of these strings does not exceed 50 characters.

Output

For each test case, print a single integer that represents the maximum number of apartments that can be furnished by Omar

Example

standard input	standard output
1 6 bedXs kitchenSS1 kitchen2 bedXs living12 livingh	1

Problem F. Building Numbers

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

In this problem, you can build a new number starting from 1, by performing the following operations as much as you need:

- Add 1 to the current number.
- Add the current number to itself (i.e. multiply it by 2).

For example, you can build number 8 starting from 1 with three operations ($1 \to 2 \to 4 \to 8$). Also, you can build number 10 starting from 1 with five operations ($1 \to 2 \to 4 \to 8 \to 9 \to 10$).

You are given an array a consisting of n integers, and q queries. Each query consisting of two integers l and r, such that the answer of each query is the total number of operations you need to preform to build all the numbers in the range from l to r (inclusive) from array a, **such that each number** a_i ($l \le i \le r$) **will be built with the minimum number of operations**.

Input

The first line contains an integer T ($1 \le T \le 50$), where T is the number of test cases.

The first line of each test case contains two integers n and q ($1 \le n, q \le 10^5$), where n is the size of the given array, and q is the number of queries.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^{18}$), giving the array a.

Then q lines follow, each line contains two integers l and r ($1 \le l \le r \le n$), giving the queries.

Output

For each query, print a single line containing its answer.

Example

standard input	standard output

1
5 3
4 7 11 8 10
4 5
1 5
3 3
	7
	18
	5

Note

As input/output can reach huge size it is recommended to use fast input/output methods: for example, prefer to use `scanf/printf` instead of `cin/cout` in C++, prefer to use `BufferedReader/PrintWriter` instead of `Scanner/System.out` in Java.

In the first query, you need 3 operations to build number 8, and 4 operations to build number 10. So, the total number of operations is 7.

Problem G. Most Common Suffix

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

You are given n strings, and q queries. For each query i, your task is to find the most common suffix of length x_i, you need to print how common it is.

The suffix i (or the i^{th} suffix) of a string is a special case of substring that goes from the i^{th} character of the string up to the last character of the string. For example, the 4^{th} suffix of "`development`" is "`lopment`", the 7^{th} suffix of "`development`" is "`ment`" (0-based indexing).

Input

The first line contains an integer T $(1 \leq T \leq 75)$, where T is the number of test cases.

The first line of each test case contains two integers n and q $(1 \leq n, q \leq 10^4)$, where n is the number of strings, and q is the number of queries.

Then n lines follow, each line contains a string s, giving the strings. All strings are non-empty consisting of lowercase English letters.

Then q lines follow, each line contains an integer x $(1 \leq x \leq m)$, giving the queries. Where m equals the length of the longest string among the given n strings.

The total length of strings in each test case does not exceed 10^5.

Output

For each query, print a single line containing the answer.

Example

standard input	standard output
1 5 4 abccba abddba xa abdcba aneverknow 1 2 4 3	4 3 1 2

Note

As input/output can reach huge size it is recommended to use fast input/output methods: for example, prefer to use `scanf/printf` instead of `cin/cout` in C++, prefer to use `BufferedReader/PrintWriter` instead of `Scanner/System.out` in Java.

Problem H. Eyad and Math

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Eyad was given a simple math problem, but since he is very bad at math he asked you to help him.

Given 4 numbers, a, b, c, and d. Your task is to find whether a^b is less than c^d or not.

It is guaranteed that the two numbers above are never equal for the given input.

Input

The first line contains an integer T ($1 \le T \le 10^5$), where T is the number of test cases.

Then T lines follow, each line contains four integers a, b, c, and d ($1 \le a, b, c, d \le 10^9$).

Output

For each test case, print a single line containing "<" (without quotes), if a^b is less than c^d. Otherwise, print ">" (without quotes).

Example

standard input	standard output

2
9 2 5 3
3 4 4 3
	<
	>

Problem I. Move Between Numbers

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

You are given n magical numbers $a_1, a_2, \ldots, a_n$, such that the length of each of these numbers is 20 digits.

You can move from the i^{th} number to the j^{th} number, if the number of common digits between a_i and a_j is **exactly** 17 digits.

The number of common digits between two numbers x and y is computed is follow:

$$Common(x, y) = \sum_{i=0}^{9} min(countX_i, countY_i).$$

Where $countX_i$ is the frequency of the i^{th} digit in the number x, and $countY_i$ is the frequency of the i^{th} digit in the number y.

You are given two integers s and e, your task is to find the minimum numbers of moves you need to do, in order to finish at number a_e starting from number a_s.

Input

The first line contains an integer T $(1 \leq T \leq 250)$, where T is the number of test cases.

The first line of each test case contains three integers n, s, and e $(1 \leq n \leq 250)$ $(1 \leq s, e \leq n)$, where n is the number of magical numbers, s is the index of the number to start from it, and e is the index of the number to finish at it.

Then n lines follow, giving the magical numbers. All numbers consisting of digits, and with length of 20 digits. Leading zeros are allowed.

Output

For each test case, print a single line containing the minimum numbers of moves you need to do, in order to finish at number a_e starting from number a_s. If there is no answer, print -1.

Example

standard input	standard output
`1`	`3`
`5 1 5`	
`11111191111191111911`	
`11181111111111818111`	
`11811171817171181111`	
`11111116161111611181`	
`11751717818314111118`	

Note

In the first test case, you can move from a_1 to a_2, from a_2 to a_3, and from a_3 to a_5. So, the minimum number of moves is 3 moves.

Problem J. Boxes Game

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Ibraheem and Husam are playing a game with group of boxes, lined next to each other on a straight line, such that each box contains a card with some value written on it. Ibraheem and Husam already know the values of all cards before the game starts.

Ibraheem and Husam take turns to play the game, Ibraheem starts first. In each turn, a player can take one box from either ends of the line, add the value of the card inside this box to his score, then remove this box from the line.

Knowing that both players play optimally, your task is to calculate the value x - y, such that x is Ibraheem's score at the end of the game, and y is Husam's score at the end of the game.

Input

The first line contains an integer T ($1 \leq T \leq 500$), where T is the number of test cases.

The first line of each test case contains an integer n ($1 \leq n \leq 10^3$), where n is the number of boxes in the line.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($-10^3 \leq a_i \leq 10^3$), giving the values of the cards inside the boxes.

Output

For each test case, print a single line containing its answer.

Example

standard input	standard output

5
4
3 1 100 5
1
1000
4
-1 100 4 -5
1
-4
1
0
	97
	1000
	92
	-4
	0

Problem K. Malek and Summer Semester

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

Malek registered n courses for the summer semester. Malek has a success rate m, which means he has to succeed at least in $ceil(n \times m)$ courses out of the n courses, in order to consider the summer semester as a successful semester. Malek is considered successful in the i^{th} course, if his grade on this course was greater than or equal to 50.

$ceil(x)$ is the smallest integer greater than or equal to x. For example, $ceil(0.95) = 1$, $ceil(4) = 4$, and $ceil(7.001) = 8$.

Malek will give you his grades in the n courses, and your task is to tell him whether the summer semester was a successful semester or not.

Input

The first line contains an integer T $(1 \le T \le 100)$, where T is the number of test cases.

The first line of each test case contains an integer n and a real number m $(1 \le n \le 100)$ $(0 < m < 1)$, where n is the number of courses, and m is the required success rate.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le 100)$, where a_i is the grade of the i^{th} course.

The success rate m is given with exactly two digits after the decimal point.

Output

For each test case, print a single line containing “`YES`” (without quotes), if the summer semester was a successful semester for Malek. Otherwise, print “`NO`” (without quotes).

Example

standard input	standard output

2
5 0.60
45 46 48 48 50
5 0.75
100 99 98 97 100
	NO
	YES

Problem L. Roads and Tracks

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

You are given a road of length n, that has m parallel tracks numbered from 1 to m.

You will start moving from the beginning of the road, and at track number 1. You want to pass the whole road (i.e. finish at position $n+1$) and ending at any track.

If you are at position i and track number j, the cost to move **forward** to position $i+1$ and track j is a_{ij}.

You can move between tracks at the same position. If you are at position i, it will take b_{ij} seconds to move from track j to $j+1$ (or from track $j+1$ to j). Note that moving between tracks does not change your position on the road.

Your task is to find what is the minimum cost required to pass the whole road starting from the beginning of the road and at track 1, and ending at any track. If there are multiple solutions, choose the one with minimum time.

Input

The first line contains an integer T, where T is the number of test cases.

The first line of each test case contains two integers n and m $(1 \le n \le 25000)$ $(2 \le m \le 25)$, where n is the length of the road, and m is the number of tracks.

Then n lines follow, each line contains m integers, where the j^{th} integer in the i^{th} line is the cost of passing through the road from position i to position $i+1$ at track j $(0 \le a_{ij} \le 10^9)$.

Then n lines follow, each line contains $m-1$ integers, where the j^{th} integer in the i^{th} line is the number of required seconds to switch between tracks j and $j+1$ at position i $(0 \le b_{ij} \le 10^9)$

Output

For each test case, print a single line containing two space separated integers x and y, where x is the minimum cost required to pass the road, and y is the minimum possible time to pass the road with the minimum cost x.

Example

standard input	standard output

1
5 3
1 1 1
2 2 2
2 1 3
1 7 1
2 1 2
2 1
1 3
1 2
1 8
2 1
	6 4

Note

As input/output can reach huge size it is recommended to use fast input/output methods: for example, prefer to use `scanf/printf` instead of `cin/cout` in C++, prefer to use `BufferedReader/PrintWriter` instead of `Scanner/System.out` in Java.