Problem A. Accumulator Apex

Time limit: 3 seconds
Memory limit: 1024 megabytes

Allyn is playing a new strategy game called "Accumulator Apex". In this game, Allyn is given the initial value of an integer x, referred to as the accumulator, and k lists of integers. Allyn can make multiple turns. On each turn, Allyn can withdraw the leftmost element from any non-empty list and add it to the accumulator x if the resulting x is non-negative. Allyn can end the game at any moment. The goal of the game is to get the largest possible value of the accumulator x. Please help Allyn find the largest possible value of the accumulator x they can get in this game.

Input

The first line of the input contains two integers x and k ($0 \le x \le 10^9, 1 \le k \le 10^5$) — the initial value of the accumulator x and the number of lists. The next k lines contain the description of lists: an integer l_i ($l_i \ge 1$) followed on the same line by l_i elements of the list in the order from left to right. Each element of lists does not exceed 10^9 by the absolute value, and the total size of all lists does not exceed 10^5.

Output

The sole line of the output should contain the largest value of the accumulator x Allyn can get.

Examples

standard input	standard output
1 3 2 -1 2 2 -2 3 2 -3 4	4
1 2 3 -1 -1 4 4 1 -3 -4 8	4

Note

In the first input, we start with $x = 1$. Then, we can take the first integer from the first list and get $x = 0$ — adding the next integer 2 from the first list we get $x = 2$. After that, we can add the integers from the second list and obtain $x = 3$. Finally, we can add the integers from the third list and obtain $x = 4$.

In the second input, we can add the first integer from the second list and get $x = 2$. Then, by adding the elements from the first list, we get $x = 4$. We cannot add more integers to increase x.

Problem B. Blueprint for Seating

Time limit: 3 seconds
Memory limit: 1024 megabytes

An aircraft manufacturing company wants to optimize their products for passenger airlines. The company's latest research shows that most of the delays happen because of slow boarding.

Most of the medium-sized aircraft are designed with 3-3 seat layout, meaning each row has 6 seats: 3 seats on the left side, a single aisle, and 3 seats on the right side. At each of the left and right sides there is a window seat, a middle seat, and an aisle seat. A passenger that boards an aircraft assigned to an aisle seat takes significantly less time than a passenger assigned to a window seat even when there is no one else in the aircraft.

The company decided to compute an *inconvenience* of a layout as the total sum of distances from each of the seats of a single row to the closest aisle. The distance from a seat to an aisle is the number of seats between them. For a 3-3 layout, a window seat has a distance of 2, a middle seat — 1, and an aisle seat — 0. The inconvenience of a 3-3 layout is $(2 + 1 + 0) + (0 + 1 + 2) = 6$. The inconvenience of a 3-5-3 layout is $(2 + 1 + 0) + (0 + 1 + 2 + 1 + 0) + (0 + 1 + 2) = 10$.

Formally, a layout is a sequence of positive integers $a_1, a_2, \ldots, a_{k+1}$ — group i having a_i seats, with k aisles between groups, the i-th aisle being between groups i and $i + 1$. This means that in a layout each aisle must always be between two seats, so no aisle can be next to a window, and no two aisles can be next to each other.

The company decided to design a layout with a row of n seats, k aisles and having the minimum inconvenience possible. Help them find the minimum inconvenience among all layouts of n seats and k aisles, and count the number of such layouts modulo 998 244 353.

Input

The first line contains an integer t — the number of test cases you need to solve ($1 \le t \le 10^5$).

For each of the test cases, there is a single line containing n and k — the number of seats, and the number of aisles in a row ($2 \le n \le 10^9$; $1 \le k \le 10^5$; $k < n$).

The total sum of k in all t given test cases does not exceed 10^6.

Output

For each test case print two integers — the minimum inconvenience among all possible layouts, and the number of layouts with the minimum inconvenience modulo 998 244 353.

Example

standard input	standard output
8 4 1 3 2 4 2 5 2 6 1 6 2 1000000000 1 9 2	2 1 0 1 0 1 1 3 6 1 2 4 249999999500000000 1 6 3

Note

In the last test case of `9 2` the possible layouts with the minimum inconvenience of 6 are 3-4-2, 2-4-3, and 2-5-2.

Problem C. Cactus Transformation

Time limit:	3 seconds
Memory limit:	1024 megabytes

In the university, Caroline started to learn about cactus graphs. Her teacher wanted to check whether the students really understood the definition of a cactus or not and gave them the following problem as a home assignment:

You are given two cactuses with the same number of vertices and edges. Your task is to answer whether it is possible to transform the first cactus into the second one using only the following two-step operation at most 15 000 times:

- Pick an **arbitrary** edge from the **first** cactus and remove it (note that after this action, it's not necessary that graph is a cactus);
- Add an **arbitrary non-existing** edge into the **first** graph, so that the graph becomes a cactus.

Note that the operation consists of both actions, so you **must** apply both actions.

It's guaranteed that if it's possible to transform the first cactus into the second one, then it can be done by using at most 15 000 operations.

The teacher promised to give a perfect grade without an exam to anyone who solved the problem. Since the given cactuses are big and Caroline can't solve the problem independently in this short period of time, she asked you to help her write a program that solves the problem.

A *cactus* is a connected undirected graph in which every edge lies on at most one simple cycle. Intuitively, a cactus is a generalization of a tree where some cycles are allowed. Multiedges (multiple edges between a pair of vertices) and loops (edges that connect a vertex to itself) are not allowed in a cactus.

Two cactuses are called *same* if for any pair of vertices v and u ($1 \le v < u \le n$), either there exists an edge (v, u) in both cactuses or does not.

Input

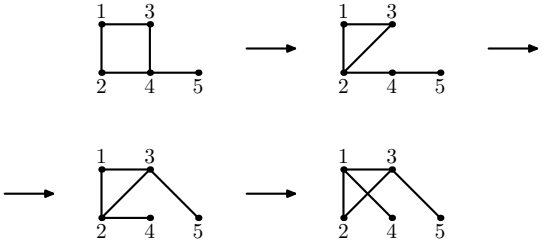
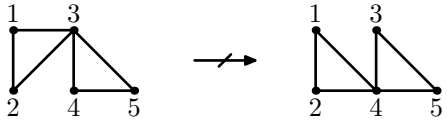
The first line contains two integers n and m ($3 \le n \le 1000$, $n-1 \le m \le \lfloor \frac{3(n-1)}{2} \rfloor$) — the number of vertices and edges in the cactuses. Each of the next $2 \cdot m$ lines contains two integers u and v ($1 \le u \ne v \le n$) — the edges of the cactuses. The first m lines represent the first cactus, while the second m lines represent the second cactus.

Output

If transforming the first cactus into the second one is impossible, output the single line with the word "NO".

Otherwise, in the first line output the single word "YES". In the second line output an integer c ($0 \le c \le 15\,000$) — the number of operations. Each of the following c lines should contain four integers w_i ($1 \le i \le 4$, $1 \le w_i \le n$). The first two integers (w_1, w_2) represent the vertices of the removed edge, while the last two integers (w_3, w_4) represent the vertices of the added edge.

Examples

standard input	standard output	Illustration
5 5 1 2 3 1 2 4 3 4 4 5 1 2 3 2 3 1 4 1 3 5	YES 3 3 4 2 3 5 4 3 5 2 4 1 4	
5 6 1 2 2 3 1 3 4 3 3 5 5 4 1 2 2 4 4 1 4 3 3 5 4 5	NO	

Problem D. Divisibility Test

Time limit: 3 seconds
Memory limit: 1024 megabytes

Daisy has recently learned divisibility rules for integers and she is fascinated by them. One of the tests she learned is a divisibility test by 3. You can find a sum of all digits of a decimal number and check if the resulting sum is divisible by 3. Moreover, the resulting sum of digits is congruent modulo 3 to the original number — the remainder modulo 3 is preserved. For example, $75 \equiv 7 + 5 \pmod 3$. Daisy is specifically interested in such *remainder preserving* divisibility tests.

There are more examples like that that she learned for decimal integers (integers base 10):

- To test divisibility modulo 11, find an alternating sum of digits. Counting digits from the last (least significant) digit, add digits on odd positions (the last, 3rd to the last, etc) and subtract digits on even positions (2nd to the last, 4th to the last, etc) to get the sum that is congruent modulo 11 to the original number. For example, $123 \equiv 1 - 2 + 3 \pmod{11}$.
- To test divisibility modulo 4, keep the last **two digits**. Their value is congruent modulo 4 to the original number. For example, $876543 \equiv 43 \pmod 4$.
- To test divisibility modulo 7, find an alternating sum of groups of **three digits**. For example, $4389328 \equiv 4 - 389 + 328 \pmod 7$.

Similar tests can be found in other bases. For example, to test divisibility modulo 5 for octal numbers (base 8), find an alternating sum of groups of **two digits**. For example, $1234_8 \equiv -12_8 + 34_8 \pmod 5$.

Daisy wants to find such rules for a given base b. She is interested in three kinds of divisibility rules:

- **Kind 1** — take the last k digits of an integer in base b.
- **Kind 2** — take a sum of groups of k digits of an integer in base b.
- **Kind 3** — take an alternating sum of groups of k digits of an integer in base b.

It is not always possible to find such a divisibility rule. For example, in base 10 there is no such test for divisibility modulo 6, even though different approaches to testing divisibility by 6 exist.

Given base b and modulo n, Daisy wants to know the smallest group size k for which such a divisibility test exits.

Input

There are several tests in the input. The first line of the input contains an integer t — the number of tests. The next t lines describe the tests.

Each test consists of a line with two integers b and n — the base and the modulo ($b, n \geq 2$). The sum of all b values in the input does not exceed 10^6, and the sum of all n values in the input does not exceed 10^6.

Output

Write t lines — a line for each test in the input. On a line for a test write a single integer 0 if the divisibility test for a given b and n does not exist. Otherwise, write two integers a and k, where a is the kind of the divisibility test (1, 2, or 3) and k is the number of digits in a group for the test, such that k is the lowest among all possible divisiblity tests for the given b and n.

Example

standard input	standard output
6 10 3 10 11 10 4 10 7 8 5 10 6	2 1 3 1 1 2 3 3 3 2 0

Problem E. Evaluate It and Back Again

Time limit: 3 seconds
Memory limit: 1024 megabytes

Aidan and Nadia are long-time friends with a shared passion for mathematics. Each of them has a favorite number: Aidan's favorite number is p, and Nadia's is q.

To commemorate their friendship, their friends want to make a present: a plaque with an arithmetic expression whose value is equal to their favorite numbers. At first glance, it sounds impossible, but the answer is simple: Aidan reads *left-to-right*, while Nadia reads *right-to-left*, so the same expression can have different values for them.

For example, if `2023-12-13` is written on the plaque, then Aidan would calculate the result as $2023 - 12 - 13 = 1998$, and Nadia would calculate it as $31 - 21 - 3202 = -3192$.

Find an arithmetic expression that, when read left-to-right, evaluates to p, and, when read right-to-left, evaluates to q. Its length must be at most 1000 characters. It's guaranteed that such an expression exists for all valid inputs.

Input

The first line of the input contains two integers p and q ($-10^{18} \le p, q \le 10^{18}$).

Output

Print the expression without spaces or line breaks. It can only contain digits 0 through 9, '`+`', '`-`', and '`*`' characters.

The expression must contain at most 1000 characters. Leading zeros in numbers are not allowed (the only exception is the notation '`0`' representing the number 0) in both the expression and its reverse. Use of unary '`+`' or '`-`' is not allowed. The expression must be well-formed in both directions. The calculation uses the standard operator precedence.

Examples

standard input	standard output
1998 -3192	2023-12-13
413 908	12*34+5

Problem F. Fugitive Frenzy

Time limit: 5 seconds
Memory limit: 1024 megabytes

The city of F. can be represented as a tree. A famous fugitive is hiding in it, and today a faithful police officer decided to catch him at all costs. The police officer is stronger than the fugitive, but the fugitive is much faster than the former. That is why the pursuit proceeds as follows. At the moment $t = 0$ the police officer appears at the vertex with number s, and the fugitive spawns at any other vertex of his choice. After that, they take turns, starting with the police officer.

- During the police officer's move, she selects any vertex adjacent to the one where she is currently located and moves there. The police officer spends one minute moving. Also, the police officer may decide to stand still instead, in which case she waits one minute at the vertex at which she started her move. If at the end of the turn the police officer ends up at the same vertex as the fugitive, she instantly catches him and the chase ends.
- The fugitive's move is as follows. Let him be at vertex b, and the police officer at vertex p. Then the fugitive chooses any vertex $b' \neq p$ such that the path between the vertices b and b' does not contain vertex p and instantly moves there. In particular, he can always choose $b' = b$ to stay where he is. The fugitive's move takes no time.

Note that the fugitive managed to attach a radio bug to the police officer's badge a week ago, so the fugitive knows the location of the police officer at every moment (in particular, he knows the number s). On the contrary, the police officer knows nothing about the fugitive's movements and will only be able to detect him at the very moment she catches him.

The police officer aims to catch the fugitive as fast as possible, and the fugitive aims to be caught as late as possible. Since the chase can be thought of as a game with incomplete information, participants can use mixed (probabilistic) strategies — thus, the police officer acts to minimize the expected duration of the chase, and the fugitive — to maximize it.

Find the mathematical expectation of the duration of the chase with optimal actions of the police officer and the fugitive. It can be proven that it is always finite. In particular, with optimal strategies, the probability that the chase continues indefinitely is equal to zero.

Input

The first line contains an integer n — the number of vertices in the tree ($2 \le n \le 100$). The next $n - 1$ lines describe the city of F.: each of them contains a pair of integers u_i, v_i — the numbers of the ends of an edge ($1 \le u_i, v_i \le n$). These edges are guaranteed to form a tree.

The last line contains an integer s — the number of the vertex where the police officer initially appears ($1 \le s \le n$).

Output

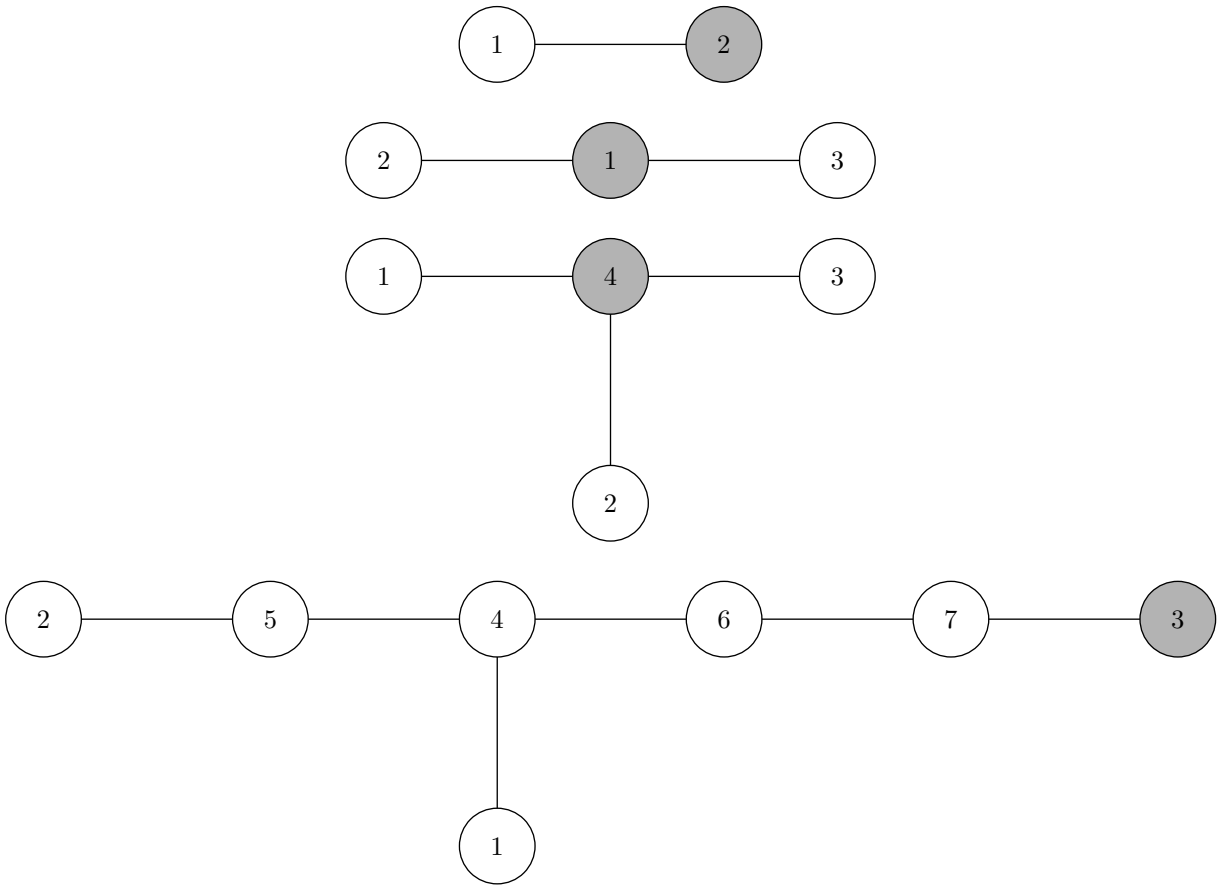
Print one real number — the mathematical expectation of the duration of the chase with the optimal strategies of the police officer and the fugitive. Your answer will be accepted if its absolute or relative error does not exceed 10^{-6}; formally, if p is your answer, and j is the jury's answer, this should hold: $\frac{|p-j|}{\max\{1,|j|\}} \le 10^{-6}$.

Examples

standard input	standard output
2 1 2 2	1
3 1 2 1 3 1	2
4 4 3 4 1 4 2 4	3.66667
7 1 4 4 5 5 2 4 6 6 7 7 3 3	8.3525

Note

The trees from the examples are depicted below.



Problem G. Great City Saint Petersburg

Time limit: 5 seconds
Memory limit: 1024 megabytes

Saint Petersburg is the most beautiful city in the world unless it is raining. For the sake of this problem, we will assume it is raining every single day.

One of the streets in Saint Petersburg has an unusual shape — it is a narrow stripe of n sections 1 meter long each, where section i is at the height a_i meters from the ground. The stripe is 1 meter deep and bounded on the front and on the back by incredibly high buildings. Because of this, when it is raining, a certain amount of rain will accumulate, unable to flow out of the street from either its leftmost or rightmost end. Given the heights a_1, a_2, ..., a_n, you need to determine the amount of rain (in cubic meters) which will accumulate on the street.

Moreover, your colleagues from the metropolitan construction company will be visiting for q days and on day i they will be laying asphalt on all sections from l_i to r_i inclusive, thus increasing the height of each section l_i, $l_i + 1$, ..., r_i by 1 meter. You need to determine the total amount of water which accumulates on the street before the construction works, and also after every single day of the construction works.

Input

The first line contains the number of blocks n and the number of construction events q ($1 \le n, q \le 2 \cdot 10^5$). The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the height of each section before all the events. Each of the following q lines contains a pair of integers l_i, r_i ($1 \le l_i \le r_i \le n$), denoting the construction work from l_i to r_i inclusive.

Output

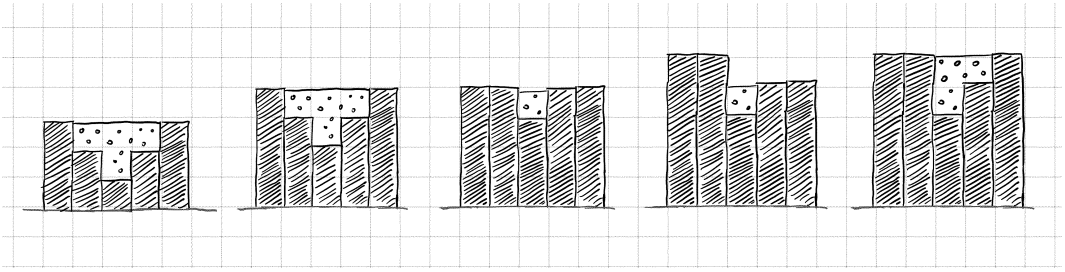
Print $q + 1$ integers — the amount of water on the street before all updates, and also after every update.

Examples

standard input	standard output
5 4 3 2 1 2 3 1 5 2 4 1 2 5 5	4 4 1 1 3
7 3 1 1000000000 1 1 1 1000000000 1 1 3 4 5 5 7	2999999997 2999999996 2999999994 2999999996

Note

The picture illustrates the amount of water accumulating on the street in the first example.



Problem H. Hypercatapult Commute

Time limit: 3 seconds
Memory limit: 1024 megabytes

A revolutionary new transport system is currently operating in Byteland. This system requires neither roads nor sophisticated mechanisms, only giant catapults.

The system works as follows. There are n cities in Byteland. In every city there is a catapult, right in the city center. People who want to travel are put in a special capsule, and a catapult throws this capsule to the center of some other city. Every catapult is powerful enough to throw the capsule to any other city, with any number of passengers inside the capsule. The only problem is that it takes a long time to charge the catapult, so it is only possible to use it once a day.

The passenger may need to use the catapults multiple times. For example, if the passenger wants to travel from city A to city B, they can first use one catapult to move from A to C, and then transfer to another catapult to move from C to B.

Today there are m passengers. Passenger i wants to travel from city a_i to city b_i. Your task is to find the way to deliver all the passengers to their destinations in a single day, using the minimal possible number of catapults, or say that it is impossible.

Input

The first line of the input contains two integers n and m ($1 \le n \le 1000$, $0 \le m \le 10^5$) — the number of cities and the number of passengers. The next m lines contain pairs of numbers a_i and b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$).

Output

In the first line print the number k — the minimal number of catapults you need to use.

In the next k lines, print descriptions of each catapult launch, in the order they need to be performed. Each description should consist of two integers c_i, d_i, the index of the city to launch from, and the index of destination city.

Note that you don't need to print what passengers should be put into the capsule on each launch, but it should be possible for each passenger to reach their destination city using the plan you provide.

If it is impossible to deliver all passengers, print the single number -1.

Examples

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

Problem I. Innovative Washing Machine

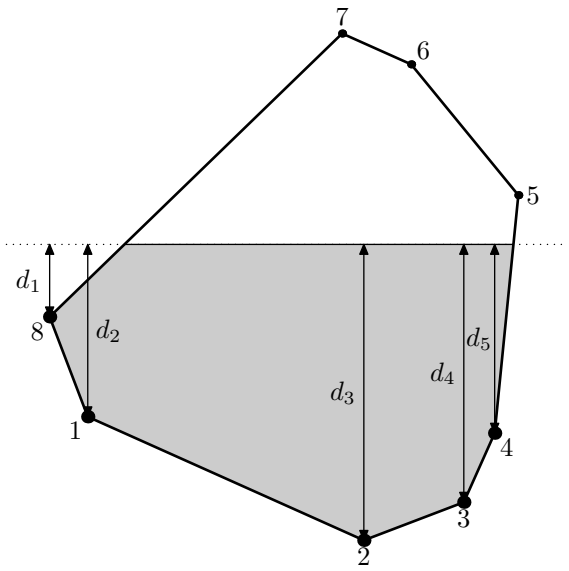
Time limit:	3 seconds
Memory limit:	1024 megabytes

You are asked to help a team that participates in "Innovation Workshop" — an event where teams of students invent and prototype their innovative ideas. One of the teams developed a new innovative washing machine that significantly reduces the usage of energy needed for laundry.

The innovative idea was to use a convex polygon instead of a circle for the shape of a washing machine drum. You are given this polygon. A drum is rotating around some fixed point inside the polygon with a constant speed of 1 turn in 1 second.

Currently, the prototype is built and testing is started. There are s litres of water in the drum. At each moment of time, water under the influence of gravity occupies a region with area s at the bottom of the drum.

Vertices of the polygon that are underwater are under pressure. By Pascal's law, we know that pressure is proportional to depth. Let's define by $d_1, d_2, \ldots, d_k$ depths of the vertices that are underwater at some moment of time, k is the number of underwater vertices. Let's define the *pressure imbalance* as the average difference between underwater vertex depth and the maximum underwater vertex depth, i.e. $\frac{1}{k}\sum_{i=1}^{k}\left(\max_{j=1}^{k} d_j - d_i\right)$. Note that the order of d_i is not important.



The polygon from the third test case is rotated. Vertices 1, 2, 3, 4, 8 are underwater.

To select the optimal shape of the drum, the team wants to know the expected value of pressure imbalance for the moment of time selected uniformly from segment $[0, 1]$ (in seconds). Please help the team to calculate this value.

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The next lines contain descriptions of test cases.

The first line contains two integers n, s ($3 \le n \le 2 \cdot 10^5$, $s \ge 1$) — the number of vertices in the polygon and the number of litres of water inside the drum. It is guaranteed that s is less than the area of the polygon.

Each of the next n lines contains two integers x_i, y_i ($|x_i|, |y_i| \le 10^8$) — coordinates of polygon vertices.

It is guaranteed that the given points form a convex polygon. The area of the polygon is positive and no two consecutive segments are collinear. The vertices of the polygon are given in counterclockwise order.

The sum of n for all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print a single real number — the expected value of pressure imbalance for a random uniform moment of time.

Your answer will be accepted if its absolute or relative error does not exceed 10^{-5}; formally, if p is your answer, and j is the jury's answer, this should hold: $\frac{|p-j|}{\max\{1,|j|\}} \le 10^{-5}$.

Example

standard input	standard output
4 4 2 0 0 2 0 2 2 0 2 3 1 1 -1 0 1 -1 -1 8 18 -2 1 -2 -3 -1 -4 0 -4 3 -3 4 -1 4 0 -1 2 4 1 99999998 99999999 99999999 99999998 100000000 99999999 99999999 100000000	0.3729232286 0.1379212354 1.3663189952 0.2636965438

Problem J. Joy of Pokémon Observation

Time limit: 3 seconds
Memory limit: 1024 megabytes

The Pokémon Conservation Society protects Pokémon and their habitats all around the globe. In recent research, data about h habitats was collected.

Each habitat may be inhabited by several Pokémon species. Researchers know how many limbs each species has. Pokémon are swift and extremely good at hiding, so researchers were only able to detect the total number of limbs in each of the habitats.

Researchers understand that it might not be possible to find the population of each species, but would like to understand how much uncertainty is left. How many different combinations of Pokémon would have the observed number of limbs?

Input

The first line contains a single integer h ($1 \le h \le 1\,024$) — the number of habitats. The next h lines contain the description of each habitat.

Each line starts with two integers t and s ($0 \le t \le 10^9$, $1 \le s \le 3$), where t is the total number of limbs, and s is the number of species in the habitat. They are followed by s integers l_i ($1 \le l_i \le 16$) — the number of limbs for each species.

Output

Output the number of possible combinations of Pokémon in each habitat. Output should contain h lines with a single integer.

Examples

standard input	standard output
3 6 1 3 6 2 2 3 6 3 1 2 3	1 2 7
4 1000000000 3 1 1 1 0 3 2 4 5 17 2 2 4 34 3 5 3 2	500000001500000001 1 0 25

Note

For the sake of example we will use LaTeX Pokémon: Q has one limb, ∠ has two limbs, ∃ has three limbs. In the first example all three habitats have 6 limbs.

In the first example the first habitat has only one Pokémon species — ∃. So it is likely the young family containing ∃∃.

In the second habitat there are two Pokémon species: ∠ and ∃. So it is either ∠∠∠ or ∃∃.

The third habitat may contain any of the three Pokémon species: Q, ∠ and ∃. There are seven possible combinations: ∃∃, ∠∠∠, Q∠∃, QQ∠∠, QQQ∃, QQQQ∠, QQQQQQ.

In the second example the first habitat has three Pokémon species, but all of them have only one limb: ∂, Q and ρ. There are 10^9 limbs and $\sum_{i=0}^{i \le 10^9} (i+1)$ combinations.

In the second habitat no limbs were detected. So there are unfortunately no Pokémon left in the area.

In the third habitat all Pokémon have an even number of limbs, so it is not possible to have 17 limbs.

Problem K. Kim's Quest

Time limit: 3 seconds
Memory limit: 1024 megabytes

In the long-forgotten halls of Kombinatoria's ancient academy, a gifted mathematician named Kim is faced with an unusual challenge. They found an old sequence of integers, which is believed to be a cryptic message from the legendary Kombinatoria's Oracle, and Kim wants to decipher its hidden meaning.

Kim's mission is to find specific patterns within the sequence, known as *Harmonious Subsequences*. These are extraordinary subsequences where the sum of every three consecutive numbers is even, and each subsequence must be at least three numbers in length.

Given a sequence a_i $(1 \le i \le n)$ of length n, its *subsequence* of length m is equal to $a_{b_1}, a_{b_2}, \ldots, a_{b_m}$ and is uniquely defined by a set of m indices b_j, such that $1 \le b_1 < b_2 < \ldots < b_m \le n$. Subsequences given by different sets of indices b_j are considered different.

There's a twist in Kim's quest: the number of these Harmonious Subsequences could be overwhelming. To report the findings effectively, Kim must calculate the total number of these subsequences, presenting the answer as a remainder after dividing by the number 998 244 353.

Input

The first line contains a single integer n — the length of the sequence $(3 \le n \le 2 \cdot 10^5)$.

The second line contains n space-separated integers a_i — the elements of the sequence $(1 \le a_i \le 2 \cdot 10^5)$.

Output

Output one number — the number of Harmonious Subsequences, modulo 998 244 353.

Examples

standard input	standard output
3 1 2 3	1
5 2 8 2 6 4	16
5 5 7 1 3 5	0
11 3 1 4 1 5 9 2 6 5 3 6	386
54 2 1 1 1 1 2 1 2 2 2 2 1 1 1 2 1 1 2 2 1 2 2 2 2 2 2 2 1 1 1 2 2 1 1 1 1 2 2 1 1 2 2 2 2 2 1 1 1 2 2 1 2 1 1	0

Note

In the provided input data for the fifth sample, the sequence of numbers is split into three separate lines for clarity, but it should be understood that in the actual test data, the sequence is given in one line. The actual number of Harmonious Subsequences in this example is $4\,991\,221\,765 = 5 \times 998\,244\,353$, hence the output is zero as a result of finding its remainder after dividing by the number 998 244 353.

Problem L. LOL Lovers

Time limit: 3 seconds
Memory limit: 1024 megabytes

There are n food items lying in a row on a long table. Each of these items is either a loaf of bread (denoted as a capital Latin letter 'L' with ASCII code 76) or an onion (denoted as a capital Latin letter 'O' with ASCII code 79). There is at least one loaf of bread and at least one onion on the table.

You and your friend want to divide the food on the table: you will take a prefix of this row (several leftmost items), and the friend will take the rest. However, there are several restrictions:

1. Each person should have at least one item.
2. The number of your loaves should differ from the number of your friend's loaves.
3. The number of your onions should differ from the number of your friend's onions.

Find any correct division and print the number of items you take or report that there is no answer.

Input

The first line contains one integer n ($2 \le n \le 200$) — the number of food items on the table. The second line contains a string of length n consisting of letters 'L' and 'O'. i-th symbol represents the type of the i-th food item on the table: 'L' stands for a loaf of bread, and 'O' stands for an onion. It is guaranteed that this string contains at least one letter 'L' and at least one letter 'O'.

Output

Print one integer — a number k such that, if you take k leftmost items and your friend takes the remaining $n-k$ items, each of you and your friend get at least one item, your number of loaves is different from your friend's, and your number of onions is different from your friend's. If there are several possible answers, print any of them. If there are no possible answers, print the number -1.

Examples

standard input	standard output
3 LOL	-1
2 LO	1
4 LLLO	1
4 OLOL	-1
10 LLOOOOLLLO	5

Note

In the first example, in any division the left and the right part contain one loaf of bread.

In the second example, the division is 'L' and 'O', and in these two strings the number of loaves is different (1 and 0) and the number of onions is different (0 and 1).

In the third example, any number 1, 2 or 3 is a correct answer.