Problem A. Soccer Match

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

As a big sports fan, you, the primary leader of the Pigeon Kingdom, are organizing a soccer match! A total of N players signed up for the match, and you plan to divide them into three groups: Red team, Blue team, and spectators. The number of players in the Red team and the Blue team **can** be different.

There are M pairs of friends among the N participants, where $M \geq 2KN$ for some given constant $K \geq 1$. The friendship is mutual, which means that if a is a friend of b, then b is a friend of a, and vice versa. To make the match more exciting, you want to make sure that each player in the Red team has at least $K + 1$ friends in the Blue team, and each player in the Blue team has at least $K + 1$ friends in the Red team. Can you find an arrangement satisfying such constraints?

Input

The first line contains one integer T ($1 \leq T \leq 50\,000$), denoting the number of test cases. For each test case:

The first line contains three integers, N, M, and K ($1 \leq N, M, K \leq 50\,000$ and $M \geq 2KN$), denoting the number of players, the number of pairs of friends, and the given constant, respectively.

Then M lines follow, each containing two integers u and v ($1 \leq u < v \leq N$), denoting that u and v are friends.

It is guaranteed that, in each test case, each pair of (u, v) appears at most once, and the sum of M over all test cases does not exceed $50\,000$.

Output

For each test case, output two lines:

The first line begins with one integer R, denoting the number of players in the Red team. Then R space-separated integers follow, each denoting the index of a player in the Red team.

The second line follows the same format. It begins with an integer B, denoting the number of players in the Blue team. Then B space-separated integers follow, each denoting the index of a player in the Blue team.

If there are multiple solutions, you can output any one of them. It can be shown that, under such constraints, a solution always exists.

Example

standard input	*standard output*
2 5 10 1 1 2 1 3 1 4 1 5 2 3 2 4 2 5 3 4 3 5 4 5 10 20 1 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10 1 10 1 4 4 7 7 10 3 10 3 6 6 9 2 9 2 5 5 8 1 8	3 2 3 4 2 1 5 3 2 8 10 2 1 9

Problem B. Gachapon

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

According to Wikipedia, "a gacha game is a video game that implements the gacha (toy vending machine) mechanic". Similar to loot boxes, gacha games induce players to spend in-game currency to receive a random virtual item.

One of these gacha games is called Step-up Gacha, which means that the player's chances of rolling a rare item are increased each time they roll. For example, the phenomenal game Genshin Impact ensures that you can always draw out four-star items or characters in any ten consecutive rolls.

It would be helpful if we give an abstraction to these rolling rules. Consider a game with 0-star, 1-star, ..., m-star items. Assume that the probability of drawing out an i-star item in a single roll is $\frac{a_i}{\sum_{j=0}^{m} a_j}$. A single draw is a level 0 rolling, and a rolling of level k consists of exactly b_k rounds of level $(k-1)$ rollings. The highest level of a rolling is n.

A level k rolling is legal if it ensures the following:

- at least one item with at least k stars is drawn,
- for all b_k level $(k-1)$ rollings it contains, at least one item with at least $(k-1)$ stars is drawn,
- ...and so on, down to each level 0 rolling (which is a single draw), for which at least one item with at least 0 stars is drawn trivially.

Let p_i be the expected number of i-star items drawn out from a legal n-level rolling, and let q be the probability that an n-level rolling is legal. Find the values p_i and q. To avoid unpleasant huge numbers and divisions by zero, for all $0 \le i \le m$, you should only output the value $(p_i \cdot q) \bmod 998\,244\,353$.

Input

The first line contains two integers m and n: the maximum number of stars and the highest level of a rolling ($1 \le n \le m \le 4000$).

The second line contains $m+1$ integers $a_0, a_1, \ldots, a_m$: the frequencies of rolling items with $0, 1, \ldots, m$ stars ($1 \le a_i \le 4000$).

The third line contains n integers $b_1, b_2, \ldots, b_n$: the number of previous level rollings in a rolling of level $1, 2, \ldots, n$ ($2 \le b_i \le 4000$).

Output

Output $m+1$ lines. The i-th line should contain a single integer: the value of $(p_{i-1} \cdot q) \bmod 998\,244\,353$.

Examples

standard input	*standard output*
2 1 1 1 1 3	554580197 1 1
2 1 89 10 1 10	989586456 1 299473306
3 2 1 1 2 1 2 3	58137752 260406016 517809313 758026833

Note

In the first example, the answers in rational form are: $\frac{8}{9}$, 1, 1.

Problem C. Survey

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

You are doing a survey and want to find respondents in your social group. Your social group has size n, and you have a budget of m dollars. You need to divide the m dollars into n shares. Each group member will get one of the shares uniformly at random. Note that the money contained in each share can be **any non-negative real number**.

Luckily, you know the *reward threshold* of each group member. If a person has a reward threshold of x, they will participate in the survey if and only if they have received a share of at least x dollars, otherwise they will just accept the payment and not participate in the survey. Since you want as many group members as possible to participate in the survey, you need to design a plan to divide the m dollars into n shares in order to maximize the expected number of members who will participate in the survey.

Input

The first line contains two integers n ($1 \le n \le 1000$) and m ($1 \le m \le 5000$), denoting the number of group members and your budget.

The next line contains n integers $x_1, x_2, \ldots, x_n$ ($0 \le x_i \le m$), denoting the reward threshold of each member.

Output

Print a single real number: the maximum expected number of members to participate in the survey.

The answer will be considered correct if the absolute or relative error between the output and the jury's answer is at most 10^{-9}.

Examples

standard input	*standard output*
5 5 1 2 3 3 4	1.200000000000
8 69 10 10 10 10 10 10 4 5	6.375000000000

Problem D. Station

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4.5 seconds
Memory limit:	1024 mebibytes

There are n bus stations and n bus lines along the main street of City A. The bus stations are labeled from 1 to n from left to right, and the importance of station i is a_i. The bus lines are also numbered from 1 to n. A bus of line k stops at stations whose importance is greater than or equal to k. Each bus line operates in both directions.

A tourist standing at station x can take any bus that stops at station x, pick a direction, and go to the **next** station y visited by that bus in that direction (of course, it is only possible if such station exists). The cost of such trip is l_x yuan if $y < x$, or r_x yuan if $y > x$. Tourists can take multiple bus trips to reach their destination.

Now there are q tourists, and the j-th tourist wants to travel from station s_j to station t_j. Your task is to find the minimum cost of the route for each tourist.

It is guaranteed that, for each i from 1 to $n-1$, the following are true: $l_i \le l_{i+1}$ and $r_i \ge r_{i+1}$.

Input

The first line of input contains a single integer T, the number of test cases ($1 \le T \le 3 \cdot 10^4$). The descriptions of test cases follow.

The first line of each test case contains two integers n and q: the number of stations and the number of tourists ($1 \le n, q \le 3 \cdot 10^5$).

The second line contains n integers $a_1, \ldots, a_n$, where a_i is the importance of station i ($1 \le a_i \le n$).

Then follow n lines, the i-th of which contains two integers l_i and r_i: the costs at station i ($1 \le l_i, r_i \le 10^9$, $l_i \le l_{i+1}$, $r_i \ge r_{i+1}$).

Then follow q lines, the j-th of which contains two integers s_j and t_j: the endpoints of a route for j-th tourist ($1 \le s_j, t_j \le n$).

The sum of n and the sum of q over all test cases do not exceed $3 \cdot 10^5$.

Output

For each tourist, output a line with the answer.

Example

standard input	*standard output*
1 9 6 1 7 3 4 9 9 1 2 2 1 11 1 11 5 11 7 10 8 6 8 4 8 3 9 1 10 1 1 9 5 1 3 1 7 6 2 6 1 1	33 9 6 8 17 0

Problem E. Number Guessing

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

Alice has a secret integer y, which is selected from $[1, 10^{18}]$. Bob wants to get the number, so he asks Alice some questions. In each question, Bob gives an integer x in $[1, 10^{18}]$ to Alice, and Alice returns 0 if $y < x$, 1 if $y = x$, and 2 otherwise.

Everything sounds great, but the communication between Alice and Bob has been tampered with by Eve! Eve wrote a random number generator $\mathcal{H}$ that generates pseudorandom numbers between 0 and $n - 1$. Every time Bob gives the number, Eve will call $\mathcal{H}$ to generate a pseudorandom number x, and gives Bob the bitwise exclusive-or sum of x and the result that Alice returns.

Bob found that the result returned by Alice has been tampered with by someone because he is getting conflicting results. By some means, Bob obtained $\mathcal{H}$'s source code, which is shown below.

Algorithm 1: random number generator $\mathcal{H}$

Data: constant n, constant P, seed of generator *seed*
Result: a number x selected from $[0, n - 1]$
1 $seed \leftarrow (seed \times n) \mod P$;
2 $result \leftarrow seed \mod n$;
3 **return** *result*;

Fig. 1: How $\mathcal{H}$ works.

The constants P and n are specified in the code, so Bob also knows their values. But he knows nothing about the *seed*, except that he knows the tamperer, Eve, will not modify the value during the interaction. Could you help Bob get the value of Alice's secret number in 100 queries?

Input

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

Each test case starts by a line containing two integers n and P: the two constants in $\mathcal{H}$. It is guaranteed that $3 \le n \le 4$, $10 \le P \le 10^{18}$, and P is a prime number.

Interaction Protocol

To ask a question, print a single line formatted as "`?` x" ($1 \le x \le 10^{18}$). Then, you should read a single line with the answer. In each test case, you can ask at most 100 questions.

If the answer is a non-negative integer, it is the number that Bob received, and you can continue guessing. If the answer is -1, it means you exceeded the number of queries or made an invalid query. Exit immediately after receiving -1, and you will see a "`Wrong Answer`" verdict. Otherwise, you can get an arbitrary verdict because your solution will continue to read from a closed stream.

If you have got the secret number y, print a single line formatted as "`!` y". Then, you should read a single line with the answer.

If the answer is -1, it means your guess is wrong, and you should exit immediately. If the answer is 1, it means your guess is correct, and you should read and solve the next test case. Note that printing this secret number does not count as one of the 100 queries.

It is guaranteed that the interactor is deterministic, which means that the interactor does not modify the values of y and *seed* dynamically based on your queries.

Do not forget to print the end-of-line character and flush the output every time your solution prints a line. Otherwise, you will likely get an "`Idleness Limit Exceeded`" verdict.

Example

standard input	*standard output*
1 3 998244353 ? 4 ? 2 ? 1 ! 1	 0 0 1 1

Note

In the example, we have $seed = 0$, so Bob always receives the correct answer from Alice. It is reasonable for Bob to directly use binary search in this test case, but it does not apply to other possible situations.

Problem F. Build a City

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

Recently, you have become addicted to a game called "Build a City".

The game can be played on a two-dimensional map. On the map, there are $n + 1$ human settlements labelled from 0 to n, whose coordinates are (x_i, y_i) for $i = 0, 1, \ldots, n$. It is guaranteed that $x_0 = y_0 = 0$, which means that settlement 0 is located at the origin.

Your city is a rectangle made of walls, with sides parallel to coordinate axes. The city is said to contain a settlement when this settlement is inside or on the border of the rectangle.

At first, your city contains settlement 0 and is a degenerate rectangle at the origin. In one move, you can acquire an unoccupied settlement and build some walls to enclose it in the city. Specifically, the new city is the smallest rectangle with sides parallel to the coordinate axes, such that it contains what the old city contained, plus the newly acquired settlement. Your goal is to enclose all human settlements in your city.

In one move, your city's infrastructure department can only build walls of total length up to m. Note that existing walls can be reused, but cannot be placed elsewhere. So, the length of walls built in each move is the perimeter of the new rectangle minus the length of the common part of the original rectangle and the new rectangle. If the newly acquired settlement is already within the city, the length of walls to be built is zero.

You now want to know if there exists an order in which to acquire the settlements, so that the total length of walls to be built in each move does not exceed m.

Input

The first line of input contains one integer T, the number of test cases ($1 \le T \le 5 \cdot 10^5$). For each test case:

The first line contains two integers n and m, the number of unoccupied settlements and the maximum length of walls that can be built in one move ($1 \le n \le 5 \cdot 10^5$ and $1 \le m \le 4 \cdot 10^9$).

The i-th of the following n lines contains two integers x_i and y_i, the coordinates of settlement i ($1 \le x_i, y_i \le 10^9$).

It is guaranteed that the sum of n in all test cases does not exceed $5 \cdot 10^5$.

Output

For each test case, output a line containing the word "`Yes`" if such an order exists, or the word "`No`" otherwise.

Example

standard input	*standard output*
3 3 6 1 1 4 1 2 2 4 9 1 4 2 3 3 2 4 1 10 14 10 8 1 6 2 5 4 2 5 5 8 9 2 7 6 8 6 5 7 4	Yes No Yes

Problem G. Trans

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

Bob is interested in popcount and some strange transforms. Currently, he is attacking the following problem:

There is an array of 2^n integers $a_0, a_1, a_2, \ldots, a_{2^n-1}$. The task is, for each i ($0 \le i \le 2^n - 1$), to calculate

$$b_i = \sum_{j=0}^{2^n-1} (\text{popcount}(i \text{ and } j) \bmod 2) \cdot a_j,$$

where "popcount(x)" denotes the number of ones in the binary representation of x, and "and" denotes the bitwise AND operation.

Although Bob is very smart, he still can't solve the problem fast. Can you help him calculate all b_i?

Input

The first line contains a single integer n ($1 \le n \le 20$).

The second line contains 2^n integers describing the array a ($1 \le a_i \le 10^9$).

Output

Print one line with 2^n integers, the i-th of them being the value b_i.

Example

standard input	*standard output*
2 1 2 3 4	0 6 7 5

Problem H. Blind Box

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

You are the owner of a store.

The store has launched a new blind box campaign. Each blind box contains n cards, and there is a positive integer written on each card. The cards in each box are ordered in such a way that the number on the i-th card is greater than or equal to the number on the $(i-1)$-th card for every $i > 1$. Additionally, the integer on each card does not exceed m.

The store has **all** possible blind boxes satisfying the conditions above, and every two blind boxes in the store are different. Two boxes are considered different if and only if there is an index i such that the numbers on the i-th cards in the two boxes are different.

You sell blind boxes at a fixed price. After buying and opening a blind box, customers will ask you for a cashback, and the amount equals the product of the numbers on the n cards in the box. Please calculate the minimum price of each blind box to ensure that, after selling all blind boxes, your net income is non-negative.

Input

The first line of input contains two integers n and m: the number of cards in each box and the maximum value on a card ($1 \le n, m \le 10^5$).

Output

Print a single integer: the minimum price to ensure a non-negative net income. The price may be fractional, but you have to output this price modulo 998 244 353. Formally, let the minimum price be an irreducible fraction $\frac{x}{y}$. They you have to print $x \cdot y^{-1} \bmod 998\,244\,353$, where y^{-1} is an integer such that $y \cdot y^{-1} \bmod 998\,244\,353 = 1$.

Examples

standard input	*standard output*
2 2	332748120
5 5	499122514

Note

Explanation of the first example:

There are three different blind boxes: $(1, 1)$, $(1, 2)$, and $(2, 2)$.

The amounts of cashback are 1, 2, and 4, respectively.

So, the minimum price should be $\frac{7}{3}$.

And the answer in the second example is $\frac{42525}{126} = \frac{675}{2}$.

Problem I. EIP1559

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

You are an avid Ethereum researcher. Recently Ethereum passed a resolution to change the gas rate of a transaction from a value *gasPrice* to a pair (*maxFee*, *maxPriorityFee*). The exact gas price of a transaction is calculated by $gasPrice = \min(maxFee, maxPriorityFee + baseFee)$, while *baseFee* is a parameter that can change over time.

You maintain a dynamic collection of transactions. At some moments, you want to know, for a specific *baseFee*, what is the largest *gasPrice* of a transaction in the collection.

Specifically, you need to maintain a collection of transactions that supports the following three operations:

1. Add a transaction with the gas rate (*maxFee*, *maxPriorityFee*) to the collection.
2. Remove a single transaction with the gas rate (*maxFee*, *maxPriorityFee*) from the collection. It is guaranteed that there is at least one transaction that satisfies the condition.
3. For a specific *baseFee*, find the maximum value of *gasPrice* in the collection when the current base fee is *baseFee*. It is guaranteed that there is at least one transaction in the collection.

Input

The first line contains an integer t ($0 \le t \le 10^6$) representing the number of operations. For the following t lines, the first integer *type* on each line represents the type of the current operation.

If $type = 1$, the next two integers are *maxFee* and *maxPriorityFee*. You should add a transaction with gas rate (*maxFee*, *maxPriorityFee*) to the collection.

If $type = 2$, the next two integers are *maxFee* and *maxPriorityFee*. You should remove a single transaction with gas rate (*maxFee*, *maxPriorityFee*) from the collection.

If $type = 3$, the next integer is *baseFee*. You should output the maximum value of *gasPrice* in the collection when the current base fee is *baseFee*.

It is guaranteed that all the values of *maxFee*, *maxPriorityFee*, and *baseFee* are integers in range $[0, 10^6]$.

Output

For each operation with $type = 3$, output a line with an integer representing the current largest *gasPrice* when the current base fee is *baseFee*.

Example

standard input	*standard output*
9 1 200000 20000 1 150000 40000 1 120000 50000 1 130000 30000 3 80000 3 100000 3 140000 2 150000 40000 3 100000	120000 140000 160000 130000

Problem J. Three Countries

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

Today, you want to measure the accessible area of Teyvat.

Mondstadt, Liyue, and Inazuma are the three countries in Teyvat. The territories of these countries can be regarded as three circles c_1, c_2, and c_3, respectively. It is **possible** that some of the circles overlap.

Let S_i be the set of points in c_i. The area of Teyvat, S, is defined as the convex hull of points in $S_1 \cup S_2 \cup S_3$.

Formally, S is the smallest set of points satisfying the following two conditions:

- $S \supseteq S_1 \cup S_2 \cup S_3$,
- $\forall p_1, p_2 \in S, \forall \alpha \in [0, 1], \alpha p_1 + (1 - \alpha) p_2 \in S$.

You are given the circles c_1, c_2, and c_3. Your task is to calculate the area of S.

Input

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

Each test case is given on three lines. The i-th of these lines contains three integers, x, y, and r, which are the coordinates of the center and the radius of i-th circle ($1 \le x, y, r \le 100$).

Output

For each test case, output a single real number representing the area of S.

Your answer will be considered correct if its absolute or relative error when compared with the jury's answer is no more than 10^{-6}.

Example

standard input	*standard output*
3 1 1 1 2 1 1 3 1 1 1 1 1 2 2 1 3 3 1 1 1 100 85 27 100 53 82 100	7.14159265359 8.79844690308 58923.76801932990

Problem K. Surround the Cat

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

This is an interactive problem.

Consider a hexagonal tiling on the plane, as shown in the picture below.

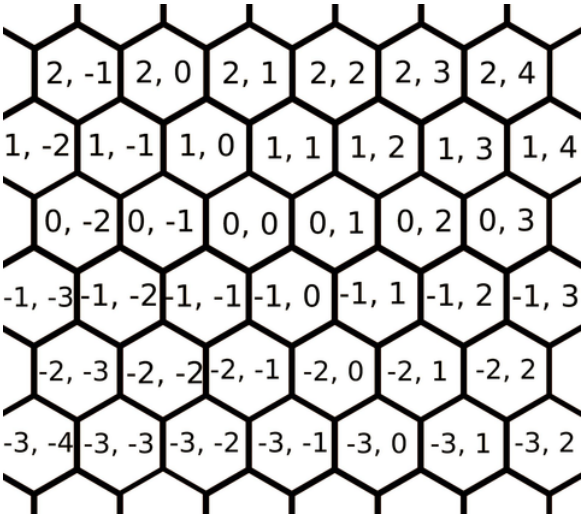
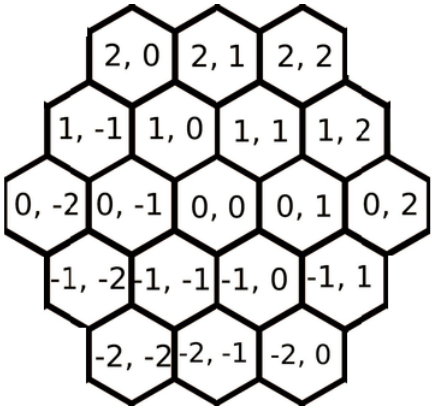


Fig. 1: What the plane looks like.

The house you built for your cat is a regular hexagon of side $N = 10$. Its six corners are located at $(9, 0)$, $(9, 9)$, $(0, 9)$, $(-9, 0)$, $(-9, -9)$, and $(0, -9)$.

For example, for $N = 3$ the house would look like this:



Now the cat wants to escape. Thus, you need to place some rocks in the house to surround the cat so that it can not escape.

Initially, the cat is located at $(0, 0)$. Every second, you can choose a location in the house and place a rock. Note that you cannot place a rock at the cat's current location. After that, the cat will choose a location that is adjacent to its current location and does not contain a rock, and move there.

Once the cat reaches the boundary of the house, it will escape and win. On the other hand, if the cat cannot choose a valid move, you win. Try to surround the cat so that it will not escape.

Interaction Protocol

Use standard input to read the locations of the cat. Use standard output to write the locations where

you place the rocks. Each location is given on a single line containing two space-separated integers: the coordinates of the location.

Initially, the cat is at $(0, 0)$, and gives you "`0 0`" as input. You respond by printing the location of the first rock, then read the next location where the cat moved, and so on.

If you print an invalid location (it is not in the house, or currently contains the cat), terminate your program to get the "`Wrong Answer`" verdict. Placing a rock in a location that already contains a rock is allowed, but does not have any additional effects. If the cat arrived at the boundary of the house, you lose: terminate your program to get the "`Wrong Answer`" verdict. If the cat has no valid moves, you win: instead of reading the cat's next location, just terminate your program gracefully.

Don't forget to print the newline character and flush the output after printing each location: otherwise, you will likely get the "`Idleness Limit Exceeded`" verdict.

Example

standard input	*standard output*
0 0 1 0 0 0 1 0 0 0 1 0 0 0	1 1 0 1 -1 0 -1 -1 0 -1 9 9 1 0