Problem A. New Financial Year

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

Naomi has spent the past year tracking her doughnut sales. With so many different flavours, some are bound to sell more than others. In order to maximize her sales for the coming year, she keeps track of certain information on each flavor. She tracks exactly N different flavours of doughnuts. For each flavour, she tracks O, the original price, P, the new price, and C, the relative change in price. The relative change in price is computed using the formula $C = 100\% \cdot (P - O)/O$.

Unfortunately, during one of her late nights, while analyzing her clipboard of data, she spilled coffee over the entire section of the page that keeps track of the original price of each doughnut flavour.

Can you help Lesley recover her data, specifically O, the original price of each doughnut flavour?

Input

The first line contains a single integer N, the number of doughnut flavours ($1 \le N \le 10^4$).

Each of the following N lines describes one flavour and contains two real numbers P and C ($1 \le P \le 1000.00$, $-1000.00 \le C \le 1000.00$), given to exactly 2 decimal places.

Output

Output N lines. On line i, print O, the original price of the i-th doughnut flavour, with absolute or relative error at most 10^{-5}.

Example

standard input	standard output

2
100.00 10.50
50.00 -50.75
	90.497737557
	101.522842640

Problem B. One More Problem About DFT

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

Let p be a prime number and $a = (a_0, a_1, \ldots, a_{n-1})$ be an array of n integers, where $p = Kn + 1$ for some positive integer K. We say that the array $\hat{a} = (\hat{a}_0, \hat{a}_1, \ldots, \hat{a}_{n-1})$ is the *Discrete Fourier Transform* of the array a if for every $k = 0, 1, \ldots, n-1$ the following holds:

$$\hat{a}_k = \left(\sum_{j=0}^{n-1} a_j w^{jk} \right) \bmod p$$

and we simply write $\hat{a} = \mathrm{DFT}(a)$. Here w denotes a primitive n-th root of unity modulo p, that is, we have $w^n \equiv 1 \pmod p$ and, for every i such that $0 < i < n$, $w^i \not\equiv 1 \pmod p$.

Note that there can be multiple choices for w, so the DFT won't be unique. Let us clarify how to uniquely find it for this problem. Let g be a generator modulo p, that is, for every x such that $0 < x < p$, there exists a positive integer r such that $0 \le r < p - 1$ and $x = g^r \bmod p$. You can find the smallest positive value for g that works and choose $w = g^K \bmod p$.

Now we define $\mathrm{DFT}^{(m)}(a) = \underbrace{\mathrm{DFT}(\mathrm{DFT}(\ldots \mathrm{DFT}(a)\ldots))}_{m \text{ times}}$, so your task is just to find $\mathrm{DFT}^{(m)}(a)$.

Input

The first line contains three space-separated integers: n ($2 \le n \le 3 \cdot 10^5$), p ($5 \le p \le 10^9 + 7$), and m ($0 \le m \le 10^{18}$), the parameters of the problem described above. It is guaranteed that p is prime and that n divides $p - 1$ evenly.

The second line contains n space-separated integers $a_0, a_1, \ldots, a_{n-1}$ ($0 \le a_i < p$), the array a.

Output

Output n space-separated integers $a'_0, a'_1, \ldots, a'_{n-1}$, the resulting array after doing the operation stated in the problem.

Example

standard input	standard output

`6 61 4`
`24 17 39 52 25 7`
	`10 2 1 42 46 8`

Note

In the sample test case, the smallest possible generator for $p = 61$ is $g = 2$. We have that $K = \frac{61-1}{6} = 10$, so we choose $w = 2^{10} \bmod 61 = 48$ to be the primitive 6-th root of unity modulo 61. The first iterations of the DFT are as follows:

- $\mathrm{DFT}^{(0)}(a) = (24, 17, 39, 52, 25, 7)$
- $\mathrm{DFT}^{(1)}(a) = (42, 55, 25, 12, 39, 32)$
- $\mathrm{DFT}^{(2)}(a) = (22, 42, 28, 7, 51, 41)$
- $\mathrm{DFT}^{(3)}(a) = (8, 9, 51, 11, 28, 25)$
- $\mathrm{DFT}^{(4)}(a) = (10, 2, 1, 42, 46, 8)$

Problem C. Robot

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

There is an infinitely large 2-dimensional chessboard, in which every cell has a unique integer coordinate (x, y). The starting cell has coordinate $(0, 0)$. If we start from this cell, walk x steps to the right, and then walk y steps upwards, we will arrive at cell (x, y). Note that x and y could be negative, which means walking in the opposite direction.

There is a robot that starts from cell $(0, 0)$ and then executes a sequence of commands $c_1c_2 \dots c_n$, where each $c_i \in \{\mathtt{L}, \mathtt{R}, \mathtt{D}, \mathtt{U}\}$, meaning walking one step in the direction of Left, Right, Down, Up, respectively. For example, if the sequence of commands is LRLD, then the cells traveled are $(0, 0) \to (-1, 0) \to (0, 0) \to (-1, 0) \to (-1, -1)$. We call such sequence the *travel history* of the robot (in this example, the history contains five elements).

For every cell (x, y) in the travel history, if it is the i-th time the robot visits this cell, then the robot earns a score of

$$f(x, y, i) = i \cdot ((|x| + 1) \text{ xor } (|y| + 1)) + i.$$

The total score is the sum of the score of every cell in the travel history. In this example, the total score is $f(0, 0, 1) + f(-1, 0, 1) + f(0, 0, 2) + f(-1, 0, 2) + f(-1, -1, 1) = 1 + 4 + 2 + 8 + 1 = 16$.

For every i from 1 to n, please answer: if we remove c_i from the sequence of commands, what is the total score earned by the robot after executing the remaining sequence $c_1c_2 \dots c_{i-1}c_{i+1} \dots c_n$?

Input

The first line contains an integer n $(2 \le n \le 3 \cdot 10^5)$.

The second line contains a string $c_1c_2 \dots c_n$ of length n, denoting the sequence of commands.

Output

Output n lines. The i-th line must contain the total score if we remove command c_i.

Example

standard input	standard output
5 LRLDD	14 11 14 16 16

Problem D. The Hash Table

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

There is a hash table with m slots, numbered from 0 to $m - 1$. Initially the slots are empty.

There are n elements, numbered from 0 to $n - 1$, which should be inserted into the hash table in this order.

The hash function is $h(x) = x^2 \bmod m$, so the element number i will be inserted into the slot numbered $(i^2 \bmod m)$.

Because of the strange implementation, inserting an element into a slot costs T, where T is the number of elements this slot already contains. Please compute the total cost of inserting all these n elements into the table.

Input

The first line contains an integer t, denoting the number of test cases $(1 \le t \le 5)$.

Each test case is given on a single line with two integers, n and m $(1 \le n \le 10^9,\ 2 \le m \le 10^9)$.

Output

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

Example

standard input	standard output

3
5 4
1234 5678
5 4
	4
	229
	4

Note

In the first test case, the elements $0, 1, 2, 3, 4$ are inserted into slots $0, 1, 0, 1, 0$ respectively, incurring costs of $0 + 0 + 1 + 1 + 2 = 4$.

Problem E. Happiness

Input file:	`standard input`
Output file:	`standard output`
Time limit:	15 seconds
Memory limit:	512 mebibytes

You are at a local amusement park, and notice a spinning teacup ride. After observing it for a bit, you notice a few interesting things about the ride.

The ride consists of N circular teacups, all equally spaced around the circumference of a turntable: for $i = 0, 1, \ldots, N-1$, the i-th teacup is centered at the point which is initially $i \cdot \frac{360}{N}$ degrees clockwise from North. The turntable is large enough to accommodate all of the teacups without them banging into each other. In each teacup, there are N people, all equally spaced around the circumference of the teacup: for $i = 0, 1, \ldots, N-1$, the i-th person is located at the point which is initially $i \cdot \frac{360}{N}$ degrees clockwise from North. As teacups are large enough, consider people as points. In each teacup, for $i = 0, 1, \ldots, N-1$, the i-th person has *happiness value* v_i. Note that the people at the same initial position in different teacups have the same happiness value.

Over the course of Q milliseconds, one of two events happens every millisecond:

- The turntable rotates $k_i \cdot \frac{360}{N}$ degrees clockwise. The teacups compensate for that, so that their orientations relative to the exterior of the turntable are not affected.
- The teacup currently at $q_i \cdot \frac{360}{N}$ degrees clockwise from North rotates by $k_i \cdot \frac{360}{N}$ degrees clockwise around its center. The other teacups are not affected.

You want to calculate the total happiness of all people, initially and after every event. The total happiness is the sum of individual people's happiness. If a person with happiness value w is on a straight line formed by the centers of two teacups, their happiness is equal to w. Otherwise, their happiness is zero. As the total happiness may be large, please output it modulo 998 244 353.

Please write a program to track the total happiness of the participants!

Input

The first line contains two integers N and Q ($2 \le N \le 200\,000$, $1 \le Q \le 200\,000$).

The next line contains N integers, the values $v_0, v_1, \ldots, v_{N-1}$ ($1 \le v_i \le 1\,000\,000$).

The next Q lines describe the events. Each line is formatted as either "`1` k_i" ($1 \le k_i \le N$), indicating the turnable rotates $k_i \cdot \frac{360}{N}$ degrees clockwise, or "`2` q_i k_i" ($0 \le q_i < N$, $1 \le k_i \le N$) indicating the teacup currently at $q_i \cdot \frac{360}{N}$ degrees (from the top) rotates by $k_i \cdot \frac{360}{N}$ degrees clockwise about its center.

Output

Output $Q + 1$ lines, each one containing the total happiness after $0, 1, 2, \ldots, Q$ events.

Remember to output the happiness modulo 998 244 353.

Example

standard input	standard output

6 3
1 2 4 8 16 32
2 1 4
1 5
2 4 2
	189
	168
	210
	252

Problem F. Attractions On Plane

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

You are going on a trip on the Cartesian plane. Starting at $(0, 0)$ and going to $(X, 0)$ with constant speed, you will view attractions. Attractions are modeled as rectangles on the plane, with the base at (x_i, y_i), width w_i and height h_i. Unfortunately, attractions can overlap.

The distance from you to an attraction is the Euclidean distance from you to its closest point. An attraction is the *Star Attraction* if the distance from you to that attraction is the minimum among all attractions. If several attractions are at minimum distance, the one with the lower index is the Star Attraction (it had better ratings).

You want to know how much time each attraction will be the Star Attraction, in percentages.

Input

The first line will contain two integers, N and X ($1 \le N \le 200\,000$, $1 \le X \le 1\,000\,000$).

Each of the next N lines contain four integers, x_i, y_i, w_i, and h_i ($1 \le x_i, y_i \le 1\,000\,000$, $0 \le w_i, h_i \le 1\,000\,000$).

Output

Output N lines. On the i-th line, output the percentage of time that the i-th attraction is the Star Attraction. Your answer will be considered correct if its absolute or relative error is at most 10^{-8}.

Examples

standard input	standard output

2 10
1 2 5 1
2 1 1 5
	52.679491924
	47.320508076

4 7
1 3 0 0
3 4 0 0
5 5 0 0
3 4 0 0
	53.571428571
	35.714285714
	10.714285714
	0.000000000

Problem G. Measuring WAC-ness

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

Consider a string of length N. Let string S be that string repeated K times. You are interested in how wack the string is, so your task is to find the WAC-ness of this string.

The WAC-ness of a string is the number of times "`WAC`" appears as a subsequence of that string.

A subsequence of a string is a string that can be derived from the given sequence by deleting zero or more characters without changing the order of the remaining characters. Two subsequences are different if at least one of the remaining indices are different. For example, in the string "`AABC`", the subsequence formed by indices 1, 3, and 4 is distinct from the subsequence formed by indices 2, 3, and 4.

As the answer can be very large, please output the answer modulo 998 244 353.

Input

The first line will contain two integers, N and K ($1 \le N \le 200\,000$, $1 \le K \le 200\,000$), the length of the original string and the number of times that string is repeated to form S. The second and final line will contain the original string of N characters, consisting of uppercase letters of the English alphabet.

Output

Output the WAC-ness of the string S modulo 998 244 353.

Examples

standard input	standard output

`5 1`
`WABCA`
	`1`

`5 2`
`WABCA`
	`5`

Problem H. Easter Gift

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

Wesley got an array of N elements $(a_1, a_2, \ldots, a_N)$ for Easter, and is eager to sort it (so that $a_1 \le a_2 \le \ldots \le a_N$). Bored, Wesley decided to make it harder on himself by only allowing himself to swap two elements if the absolute difference between them is less than or equal to K. Note that the elements can be anywhere; as long as their absolute difference is less than or equal to K, Wesley can swap them.

Unfortunately, Wesley quickly realized that it might not be possible to sort the array. He then wonders: what is the minimum value of K required to be able to sort the array?

Input

The first line contains an integer N, the number of elements in the array ($1 \le N \le 2 \cdot 10^5$).

The next line contains N integers $a_1, a_2, \ldots, a_N$, the array itself ($1 \le a_i \le 10^{18}$).

Output

Output the minimum value of K required to be able to sort the array. If the elements are already sorted, you should output 0.

Example

standard input	standard output
8 1 4 4 2 7 14 12 10	2

Problem I. Returning Lights To Box

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

Wesley needs to take down his holiday lights. He has a line of N lights, some of which may be on, and Wesley needs all the lights to be off before he can unplug them (or else he will receive a deadly electrical shock).

Each light has a corresponding switch that can be used to turn the light on or off, and Wesley can use at most one of these switches every second, starting from the first second. However, these lights are finicky, and in the next M seconds they will toggle their state on their own! Specifically, at the end of i-th second, the b_i-th light will flip its state: turn on if it was off, or turn off if it was on. Wesley wants to take the lights down as soon as possible, so he would like to know what is the earliest time possible for all the lights to be off, assuming he uses switches in an ideal manner. In particular, output the least i such that all lights can be turned off by the end of the i-th second by some sequence of switch usages. Note that if all lights are initially off, then the least such i is 0.

Input

The first line contains two integers N and M, the number of lights and the number of unsolicited changes the lights will make ($1 \le N, M \le 2 \cdot 10^5$).

The second line will contain N integers $a_1, a_2, \ldots, a_N$ ($0 \le a_i \le 1$), the initial state of the lights. Here, $a_i = 1$ indicates that the i-th light is initially on, and $a_i = 0$ tells it is off.

The third and final line will contain M integers $b_1, b_2, \ldots b_M$, which denotes that the b_i-th light flips its state at the end of i-th second ($1 \le b_i \le N$).

Output

Output a single integer, the earliest time (in seconds) it will take for Wesley to turn all the lights off. Note that if all the lights can be turned off before M seconds have passed, Wesley will ignore any future toggles and take them down immediately.

Examples

standard input	standard output

3 3
1 1 1
1 2 3
	2

5 8
0 1 0 1 1
1 2 2 1 4 3 2 1
	4

Problem J. Into Cactus

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

Given a tree, add as many edges as possible so that the resulting graph is a cactus graph.

A cactus graph is a graph where each edge is contained in at most one simple cycle. This graph may not contain self-loops or parallel edges.

Input

The first line contains an integer N, the size of the tree ($1 \le N \le 200\,000$).

Each of the next $N - 1$ lines contains two integers u and v ($1 \le u, v \le N$, $u \ne v$), indicating that there is an edge between nodes u and v. It is guaranteed that the resulting graph is a tree.

Output

On the first line, output K, the maximum number of edges that can be added to the graph. On each of the next K lines, output two integers a and b ($1 \le a, b \le N$, $a \ne b$), indicating that you are going to add an edge between nodes a and b. The resulting graph must be a cactus graph.

If there are several solutions with the maximum possible K, output any one of them.

Example

standard input	standard output

6
6 4
3 1
3 6
4 5
2 3
	2
	2 1
	3 5

Problem K. Colorful Components

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

There are n nodes, the i-th of which has color c_i. For a given integer k ($1 \le k \le n$), please count the number of ways to build exactly $n-1$ undirected edges between the nodes, such that:

(1) The n nodes form a connected graph.

(2) If we destroy every edge that connects two nodes of different colors, then every connected component in the remaining graph has at most k vertices.

Two ways of building edges are considered different if and only if there exist two nodes i and j such that $1 \le i < j \le n$ and there is an edge between them in one of the two ways but not in the other.

Since the number could be large, you only need to output the answer modulo 10^9+7.

Input

The first line contains two integers, n and k ($1 \le k \le n \le 300$).

The following n lines contain integers $c_1, c_2, \ldots, c_n$ denoting the colors of the nodes, one integer per line ($1 \le c_i \le n$).

Output

Output the answer modulo 10^9+7.

Examples

standard input	standard output

5 3
1
1
3
1
5
	125

4 2
2
1
1
1
	7

Problem L. Almost Free Falling

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

Bob likes skydiving very much, so during his last internship at an IT company, he decided to live the experience of jumping from an aircraft and fall through the air before opening his parachute. But, before he imagined, he was already at the ground, everything happened so fast!

Now, he wants to know how much height he traveled through the air starting from the point where he jumped at some specific moments, and the total time he was falling.

We can describe this more formally: at time $t = 0$, suppose that Bob is a point-mass with mass m with initial position at the point where he jumped, with initial velocity zero (he jumped from rest) and with his parachute closed. Naturally, the gravity from the Earth will apply a downwards force equal to his weight, $m \cdot g$, where $g = 9.81\,\text{m/s}^2$. And to make it more realistic, in the period of time when he hasn't opened his parachute, the air will apply an upwards force (air resistance) equal to $b_1 v(t)$, where b_1 is a given constant and $v(t)$ is his instantaneous velocity. Then, at the moment when he opens his parachute at time $t = t_1$, that force will disappear and immediately a new upwards force equal to $b_2 v(t)$ will appear, where $b_2 > b_1$. That means that the effect of opening the parachute will apply more resistance to Bob's falling, or in other words, he will fall more slowly.

Help Bob with the task of calculating how much distance he has fallen at the moments he needs to know and the total time of falling. It is guaranteed that the parachute opens strictly before he has reached the ground (in other words, he doesn't die), and all the times in the queries are strictly less than the total time of falling.

Input

The first line contains five integers: m $(40 \le m \le 120)$, b_1, b_2 $(1 \le b_1 < b_2 \le 100)$, t_1 $(90 \le t_1 \le 180)$, and h_f $(1 \le h_f \le 70\,000)$, indicating Bob's mass in kilograms, the constant when the parachute is closed in $\text{N} \cdot \text{s/m}$, the constant when the parachute is opened in $\text{N} \cdot \text{s/m}$, the time when the parachute opens in seconds, and the total height from the starting point to the ground in meters, respectively. Recall that force is measured in Newtons, and $1\,\text{N} = 1\,\text{kg} \cdot m/\text{s}^2$.

The second line contains an integer q $(1 \le q \le 20)$, the number of queries from Bob.

Each of the following q lines contains an integer t_i in seconds, indicating a moment where Bob wants to know his traveled distance.

Output

Print q lines: for each query, the distance traveled, in meters. And after that, print one line indicating the total time of falling, in seconds.

The absolute or relative error of the answers should not exceed 10^{-4}.

Examples

standard input	standard output

45 57 95 173 2347
6
385
293
326
65
104
161
	2320.3231578947
	1892.8136842105
	2046.1594736842
	497.2936288089
	799.3383656510
	1240.7883656510
	390.7408540039

45 75 85 99 811
3
57
89
94
	331.9704000000
	520.3224000000
	549.7524000000
	143.5652618043

Problem M. Number of Colorful Matchings

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

You are given a graph G with n black nodes and n white nodes, where every edge can only connect a black node and a white node (in other words, the graph is bipartite).

Each edge in G has a color: either blue or red. No two edges of the same color can connect the same pair of vertices (in other words, there are no same-color parallel edges).

For every k from 0 to n, please count the number of perfect matchings in G that contain exactly k red edges and $n-k$ blue edges. Recall that a perfect matching is a subset of n edges in which no two edges can share a common endpoint. Since the number could be large, you are only required to output the answers modulo 2.

Input

The first line contains a non-negative integer n ($1 \le n \le 300$).

Each of the next n lines contains n characters with no spaces. Together, these lines describe the adjacency matrix of red edges. The j-th character on the i-th line is "1" if there is one red edge connecting the i-th black node and the j-th white node, and "0" otherwise.

The next n lines describe the adjacency matrix of blue edges, in the same format as above.

Output

Output $n+1$ lines containing your answers for $k = 0, 1, 2, \ldots, n$ respectively. Remember that you only need to output the answer modulo 2.

Example

standard input	standard output
2 11 10 00 11	0 0 1

Note

In the example, there exist three perfect matchings:

1. red $(1,1)$, blue $(2,2)$
2. red $(1,2)$, blue $(2,1)$
3. red $(1,2)$, red $(2,1)$

Problem N. 100 Boxes Per Hour...

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

At work, you sort 100 boxes per hour. Each box has one of three colours: red, blue, or green. The boxes come through a slot, so you cannot see what boxes have yet to come. You want to take as many boxes home as possible, but you only have two bins to store them in. Unfortunately, although the bins are not limited in capacity, you absolutely cannot have two boxes of different colours be in the same bin.

Thankfully, a coworker told you the distribution of the box colours but forgot to specify the colours. All you have are three integers that add up to 100. When you receive a box, you can do three things. First, you can choose to empty either bin you have, discarding all boxes inside it. After that, you can place the box into a bin that is either empty or contains only boxes of the specified colour. Alternatively, you can simply discard the current box.

You suspect that these boxes are valuable, so you want to keep as many as possible. You will work for $T = 100$ hours in total, collecting boxes. Each hour, you start with two empty boxes. Can you collect at least 43 out of 100 boxes at the end of each hour?

In each test, the order of the boxes is not random in any way: it is fixed in advance, before the start of the contest.

Interaction Protocol

First you will get a line with an integer T, the number of hours you will work. For all tests in this problem, $T = 100$.

For each hour you work, you will receive a line with three integers: A, B, and C, indicating that there are A boxes of some colour, B boxes of another colour, and C of the remaining colour. It is guaranteed that $0 \le A, B, C \le 100$ and $A + B + C = 100$.

Then, you will start your hour of work. You will be given a line with a single character, one of "`R`", "`G`", or "`B`", indicating the colour of the current box. Then you may empty either bin, with "`EMPTY 1`" or "`EMPTY 2`". You may empty either bin as many times as you'd like. Then, either output "`PLACE` x" where x is the bin to place the box into, or output "`DISCARD`" to discard the current box. In any case, after printing each line, remember to output a newline character and flush the output.

If at any point you output an invalid instruction or put two boxes of different colours in one bin, you will receive a line with a single integer, `-1`. At this point, your program should terminate, and you will then get the outcome "`Wrong Answer`".

Example

standard input	standard output
1	
5 2 3	
G	
	PLACE 2
G	
	PLACE 1
B	
	EMPTY 1
	PLACE 1
B	
	EMPTY 2
	EMPTY 1
	PLACE 2
G	
	DISCARD
G	
	DISCARD
R	
	PLACE 1
G	
	DISCARD
B	
	PLACE 2
R	
	PLACE 1

Note

The sample interaction shows a valid exchange, but does not comply with the constraints, as it has $T = 1$ and $A + B + C = 10$. It will not appear in any of the test cases.