Problem A. Odd/Even Strings

Source file name:	Oddeven.c, Oddeven.cpp, Oddeven.java, Oddeven.py
Input:	`Standard`
Output:	`Standard`

Consider a string containing lowercase letters only. We call the string *odd* if every letter in the string appears an odd number of times. Similarly, we call the string *even* if every letter in the string appears an even number of times.

Given a string, determine if it is odd or even.

Input

There is only one input line; it contains a string of lowercase letters only. The string will contain 1 to 60 letters (inclusive).

Output

Print `0` (zero) if the string is even, `1` (one) if the string is odd, and `0/1` (zero and one with a slash in between) if it is neither.

Example

Input	Output
geekkeeg	0
funnyn	1
zztop	0/1

Problem B. Fiborooji Sequence

Source file name:	Fiborooji.c, Fiborooji.cpp, Fiborooji.java, Fiborooji.py
Input:	`Standard`
Output:	`Standard`

The Fibonacci sequence is defined as follows:

$F_0 = 0$
$F_1 = 1$
$F_n = F_{n-1} + F_{n-2}$, for $n \geq 2$

The numbers in this sequence are 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ... The Fibonacci sequence has many applications including Mathematics, Computer Science, Nature, Economics, Art, and Music.

The *Fiborooji sequence* is a simplified version of the Fibonacci sequence. The *Fiborooji sequence* starts with two single-digit, non-negative numbers (at least one of the two numbers being nonzero). Each subsequent number is the sum of its preceding two numbers (like the Fibonacci sequence) but the numbers always stay single digit. This is done by keeping only the rightmost digit if the sum of the preceding two numbers end up having two digits. For example, if we have 8 and 7, the sum will be 15 but the next number will be 5 and not 15. As another example, if we have 4 and 6, the sum will be 10 but the next number will be 0 and not 10. Here is the *Fiborooji sequence* if we start with 2 and 1: 2, 1, 3, 4, 7, 1, 8, 9, 7, 6, ...

Applications of the *Fiborooji sequence* are being studied. One observation (discovery!) is that, regardless of the first two numbers (single digit) you start with, these two numbers will be repeated after a while which means the sequence will loop through the same numbers.

Given the first two single-digit numbers, find the length of the sequence when the starting two numbers repeat (i.e., reappear consecutively).

Input

There is only one input line; it contains two single-digit, non-negative numbers, at least one of the two numbers being non-zero.

Output

Print the length of the sequence when the starting two numbers repeat.

Example

Input	Output
3 4	14
0 6	22
1 0	62
5 5	5

Explanation

For the first example Input/Output:

The sequence will be 3, 4, 7, 1, 8, 9, 7, 6, 3, 9, 2, 1, 3, 4, which has a length of 14. Note that we started with "3 4" and, when "3 4" is repeated (reappear consecutively), the sequence has 14 elements.

Problem C. Soccer Standing Table

Source file name:	Soccer.c, Soccer.cpp, Soccer.java, Soccer.py
Input:	`Standard`
Output:	`Standard`

In soccer, a win is worth 3 points, draw (tie) is worth 1 point, and loss is worth zero points. Soccer standing tables show the following info for each team: number of matches played (MP), number of wins (W), number of draws (D), number of losses (L), and total number of points (Pts). For example, the entry:

```
USA 11 5 4 2 19
```

Indicates **USA** has played 11 matches, won 5 of them, drew 4 matches and lost 2, for a total number of points of 19 ($5 \cdot 3 + 4 \cdot 1$).

Even though the info is complete and helpful, some publications/websites list the numbers in different orders, e.g., the number of losses may be listed before the number of draws. And, if the columns are not labeled as to what they represent, that causes confusion.

Given five integers representing the entry for a soccer team, print these numbers in the order of `MP-W-D-L-Pts`.

Input

There is only one input line: it contains five integers, each between 0 and 300, inclusive. Assume that the input represents a valid entry, e.g., $MP = W + D + L$. Also assume that the input will not be all zeros, i.e., some games have been played.

Output

Print the numbers in the order of `MP-W-D-L-Pts`. Assume that there will be exactly one valid answer.

Example

Input	Output
19 11 2 4 5	11 5 4 2 19
2 2 6 20 10	10 6 2 2 20
0 0 1 1 0	1 0 0 1 0

Problem D. Caterpillar Walk

Source file name:	Caterpillar.c, Caterpillar.cpp, Caterpillar.java, Caterpillar.py
Input:	`Standard`
Output:	`Standard`

Consider a city where each building is a rectangle with its base on the positive x-axis, its two walls parallel to the positive y-axis, and its roof parallel to x-axis. The walls of two buildings can touch each other, i.e., the buildings can be right next to each other. The buildings do not, however, have any area in common, i.e., no two buildings intersect/overlap.

Lulu (our beloved caterpillar) lives at coordinates $(0,\ 0)$ and works at coordinates $(100,\ 0)$. Every morning, Lulu crawls from home to office. Without any buildings, this is a distance of 100 but, unfortunately, the buildings force Lulu to crawl up and down of the building walls and they add to the distance Lulu has to travel. Note that Lulu starts at home, ends at the office, and is always crawling on x-axis, side of a wall going up (parallel to y-axis), top of a building (parallel to x-axis), or side of a wall coming down (parallel to y-axis).

Given the description of all the buildings, compute the total distance travelled by Lulu.

Input

The first input line contains an integer, n ($1 \leq n \leq 50$), indicating the number of buildings. Each of the next n input lines describes a building. A building is described by three integers: s ($1 \leq s \leq 98$), indicating the lower-left corner of the building, w ($1 \leq w \leq 98$), indicating the width of the building, and h ($1 \leq h \leq 100$), indicating the height of the building. Assume $s + w \leq 99$, i.e., no building will end beyond $(99,\ 0)$.

Output

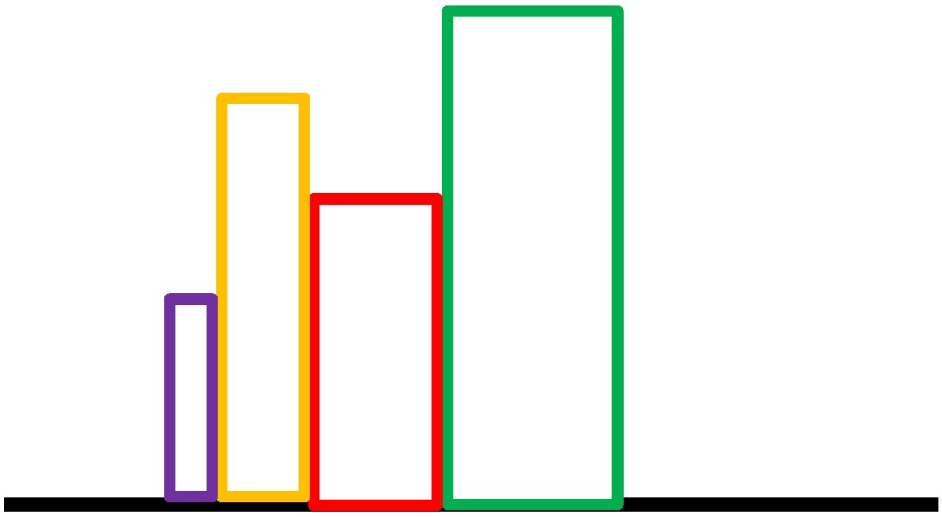
Print the total distance travelled by Lulu.

Example

Input	Output
2 40 3 7 60 10 12	138
2 43 10 12 40 3 7	124
3 40 3 7 43 10 12 53 20 8	124
3 70 20 8 40 3 7 43 10 12	140
4 12 2 10 14 4 20 18 6 15 24 8 25	160

Explanation

Picture illustrating the last Example Input:



Problem E. Tutorial Groupings

Source file name:	Tutorial.c, Tutorial.cpp, Tutorial.java, Tutorial.py
Input:	`Standard`
Output:	`Standard`

UCF has some very large classes such as COT 3100 (Introduction to Discrete Mathematics). Unfortunately, some students don't learn well in this environment. Luckily, UCF has so many knowledgeable students (referred to as TAs for the rest of this problem) that classes can be split up into smaller "tutorial" groups led by TAs, where students are more likely to achieve their potential.

In order for a tutorial group to be successful, the knowledge level of students within that group has to be relatively close to one another. More specifically, the difference between the minimum and maximum knowledge levels of students in a tutorial group can't exceed a given value k. In addition, for a tutorial group to be successful, it cannot have more than s students, for a given value s. (Of course, each group must have at least one student!) Finally, it must be possible to order the tutorial groups as T_1, T_2, ..., T_m, such that for any integer i, $1 \leq i < m$, the maximum knowledge level of a student in tutorial T_i is strictly lower than the minimum knowledge level of a student in tutorial T_{i+1}. Since TAs are so plentiful and knowledgeable, as many tutorial groups as desired can be formed.

As an example, consider a class where the students have knowledge levels 10, 6, 5, 9 and 12, such that the difference between the minimum and maximum levels of students in a tutorial can't be greater than 5 and there can be no more than 3 students in a group. For this example, we can split the class into tutorial groups in the following 13 ways:

1. {5}, {6}, {9}, {10}, {12}
2. {5}, {6}, {9}, {10, 12}
3. {5}, {6}, {9, 10}, {12}
4. {5}, {6}, {9, 10, 12}
5. {5}, {6, 9}, {10}, {12}
6. {5}, {6, 9}, {10, 12}
7. {5}, {6, 9, 10}, {12}
8. {5, 6}, {9}, {10}, {12}
9. {5, 6}, {9}, {10, 12}
10. {5, 6}, {9, 10}, {12}
11. {5, 6}, {9, 10, 12}
12. {5, 6, 9}, {10}, {12}
13. {5, 6, 9}, {10, 12}

Given all of these restrictions, your COT 3100 instructor (Dr. Travis Meade) would like to know how many ways the tutorials can be formed.

Given the knowledge level of each student in a class, the maximum difference between knowledge levels allowed in a tutorial group and the maximum number of students in a tutorial group, calculate the number of different ways to form tutorial groups. Since this number could be quite large, report it modulo 10^9+7.

Input

The first input line contains three space separated integers: n ($1 \leq n \leq 10^4$), indicating the number of students in the class, k ($1 \leq k \leq 10^9$), indicating the maximum difference between the minimum and maximum knowledge levels of students in a single tutorial, and s ($1 \leq s \leq 100$), indicating the maximum number of students allowed in a tutorial.

The following input line contains n space separated integers. The i^{th} of these integers is a_i ($1 \leq a_i \leq 10^9$), indicating the knowledge level of the i^{th} student. Assume that all the knowledge levels are distinct, i.e., no duplicates.

Output

Print the number of different ways to form tutorial groups, modulo 10^9+7. Note that two ways of forming tutorial groups are different if there is a set (tutorial group) in one but not in the other.

Example

Input	Output
6 2 5 3 6 7 12 15 16	4
7 10 7 2 3 4 6 7 8 9	64
5 5 3 10 6 5 9 12	13

Problem F. Food Poisoning

Source file name:	Food.c, Food.cpp, Food.java, Food.py
Input:	`Standard`
Output:	`Standard`

Your friend, Mikey, has had issues recently; every week he gets food poisoning. Mikey is pretty sure that the food poisoning is from *exactly one* restaurant; he doesn't know which one though. Mikey wants to devise a strategy to find out. He wants to find out as quickly as possible (least number of weeks).

Mikey is willing to eat at a subset of his usual restaurants each week to find the bad restaurant (the subset can be as small or as large as needed). However, Mikey doesn't want to get food poisoning too many times, so he wants to find the bad restaurant while getting poisoned *at most k times*.

Note that if Mikey eats at a subset of restaurants in a given week and gets food poisoning, then the bad restaurant is in that subset; if he doesn't get food poisoning, then the bad restaurant is not in that subset. Note also that the subset of restaurants for different weeks do not have to be the same size.

Given the number of restaurants and the maximum allowable number of times to get food poisoning, determine (in the worst case) the least number of weeks needed to find out which restaurant is giving Mikey food poisoning.

Input

There is only one input line; it contains two integers: n ($1 \leq n \leq 10^5$), indicating the number of restaurants, and p ($1 \leq p \leq 10^4$), indicating the maximum number of times Mikey is willing to get food poisoning.

Output

Print the least number of weeks Mikey will need to eat before definitively determining which restaurant is giving him food poisoning, given an optimum strategy, but worst luck. Note that there will always be an answer.

Example

Input	Output
8 1	7
7 2	3

Explanation

For the First Example Input/Output:

Mikey can eat in one restaurant each week; that will tell him if that restaurant is causing the food poisoning. After at most 7 weeks, he will know the bad restaurant since there are only 8 restaurants.

Problem G. Toboggan Ride

Source file name:	Toboggan.c, Toboggan.cpp, Toboggan.java, Toboggan.py
Input:	`Standard`
Output:	`Standard`

You are riding a toboggan on the straight line which can be modeled by the positive x-axis. You start at $x = 0$ and must get a boost to get started. Since there is friction, after that initial boost, eventually you would slow to a stop. Luckily, there are several positions along the positive x-axis where you will receive another boost! As long as you can make it to the next boosting position, you'll be able to finish the ride.

Of course, not only do you want to finish the ride, you also want to finish it within some time limit. Luckily, you are allowed to manipulate the boosts you receive. Unfortunately, you must set all boosts to the exact same value. Specifically, you must pick some constant, c, and then c meters/second will be added to your velocity instantaneously, as soon as you arrive at each boost point.

If you start at a position with an initial velocity of v meters/second, without any more boosts, you will travel for v seconds exactly, losing a velocity of 1 meter/second every second. At the end of the v seconds, you will be stationary and will have traveled a distance of $\frac{v^2}{2}$ meters.

If you start at a position with an initial velocity of v and travel for t seconds, where $t < v$, then you will travel a distance of $t \cdot \left(v - \frac{t}{2}\right)$ meters.

Here is an example of a possible toboggan ride:

Boost value = 10 m/s Boost locations: $x = 0,\ 32,\ 110$ End of Ride Location: $x = 238$

Starting at $x = 0$ and traveling for 4 seconds, we go a distance of $4 \cdot \left(10 - \frac{4}{2}\right) = 32$ m, arriving at the second boost point with a velocity of 6 m/s (since we lose 1 m of velocity each second).

The boost at $x = 32$ m changes our velocity from 6 m/s to 16 m/s.

Next, we travel for 6 seconds and go a distance of $6 \cdot \left(16 - \frac{6}{2}\right) = 78$ m, arriving at $x = 32$ m $+ 78$ m $= 110$ m , where we get our final boost. At this point in time, our velocity is 16 m/s $- 6$ m/s $= 10$ m/s .

We receive the final boost at $x = 110$ m, which changes our velocity from 10 m/s to 20 m/s. We travel another 8 seconds and go a distance of $8 \cdot \left(20 - \frac{8}{2}\right) = 128$ to finish the ride at $x = 238$ m.

Thus, in order to finish the ride in $4 + 6 + 8 = 18$ seconds, we must set our boost to at least 10 m/s.

Given the length of the toboggan ride (in meters), the locations where a boost will be provided, and an amount of time to finish the ride (in seconds), determine the least amount of boost in meters/second necessary to finish the ride by the desired goal time.

Input

The first input line contains three space separated integers: L ($1 \leq L \leq 10^9$), indicating the length of the toboggan ride in meters, n ($1 \leq n \leq 100$), indicating the number of boosts, and t ($1 \leq t \leq 10^9$), indicating the amount of time in seconds to complete the ride.

The second input line contains n space separated distinct integers b_1, b_2, b_3, ..., b_n ($0 = b_1 < b_2 < b_3 < \ldots < b_n < L$), representing the locations where the n boosts occur. Note that the first value in this list is zero (0).

Output

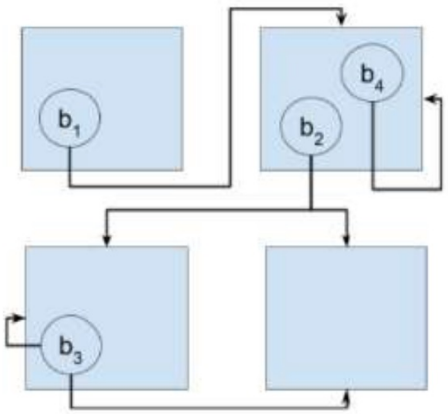
Print a single floating-point number representing the minimum boost necessary to finish the ride in the desired amount of time. Any answer with an absolute error of 10^{-6} will be accepted.

Example

Input	Output
238 3 18 0 32 110	10.0
1000 5 20 0 200 315 816 900	27.829407683424986

Problem H. Discord Daisy Chain

Source file name:	Discord.c, Discord.cpp, Discord.java, Discord.py
Input:	`Standard`
Output:	`Standard`



You are working on several discord bots in one larger server. Each bot has a channel that it "listens" to (in nontechnical terms, a bot is in a channel and receives all the messages sent to that channel). When the channel the bot listens to gets a message, the bot will send the same message to a number of other channels (dependent on the bot), causing bots in those channels to receive the message and send the same message to other channels.

Note that all the bots in a channel receive a message sent to the channel and all the bots in the channel will forward the message to their receiving channels.

You have the list of bots, and now you're curious if there is a channel such that if you post a message inside of it, all the other channels will have at least one copy of the message. For example, in the configuration illustrated in the picture, if we post a message inside the channel containing bot b_1, all the other channels will have (receive) at least one copy of the message.

Since there may be several such channels, you wonder how many are there?

Given the description of the server (number of channels and bot descriptions), determine the number of channels that can allow for a message to be sent to all the other channels.

Input

The first input line contains two integers: c ($1 \leq c \leq 10^5$), indicating the number of channels in the server, and b ($1 \leq b \leq 10^5$), indicating the number of bots in the server.

Each of the next b input lines starts with an integer, l_i ($1 \leq l_i \leq c$), indicating the channel the i^{th} bot is listening to (i.e., the channel the bot is in), and m_i ($1 \leq m_i \leq c$) indicating the number of channels the i^{th} bot will send the message to. The line will contain m_i more integers, a_1, a_2, ..., a_{l_i} ($1 \leq a_j \leq c$), indicating the channels to which the i^{th} bot will forward its messages to.

It is guaranteed that the total number of receiving channels in the input will not exceed $2 \cdot 10^5$.

Output

Print the number of channels that can be used to send a message to all the other channels.

Example

Input	Output
4 4 1 1 2 2 2 3 4 3 2 3 4 2 1 2	1
5 4 1 5 1 2 3 4 5 2 4 3 1 4 2 3 3 1 2 3 4 2 1 2	4

Explanation

For the First Example Input/Output:

This data set corresponds to the configuration illustrated in the above picture.

There are 4 channels and 4 bots.
Bot 1 is in Channel 1 and forwards messages to Channel 2.
Bot 2 is in Channel 2 and forwards messages to Channels 3 and 4.
Bot 3 is in Channel 3 and forwards messages to Channels 3 and 4.
Bot 4 is in Channel 2 and forwards messages to Channel 2.

Messages posted in Channel 1 will get to all the other channels. This is the only such channel.

Problem I. K-gap Subsequence

Source file name:	Kgap.c, Kgap.cpp, Kgap.java, Kgap.py
Input:	`Standard`
Output:	`Standard`

A subsequence of a given sequence of integers is a subset of the values in the sequence in the same order. A k-gap subsequence of a sequence of integers is a subsequence such that consecutive elements in the subsequence differ by at least k. For example, the sequence

3, **12**, **8**, **4**, 2, 5, 1, **9**, 8, 6, **5**, **1**, **7**

has a 4-gap subsequence of 3, 12, 8, 4, 9, 5, 1 and 7 (highlighted in bold above) since

$|3-12|$, $|12-8|$, $|8-4|$, $|4-9|$, $|9-5|$, $|5-1|$ and $|1-7|$ are all 4 or greater.

Given a sequence and a value of k, determine the length of the longest k-gap subsequence.

Input

The first input line contains two space separated integers: n ($1 \leq n \leq 3 \cdot 10^5$), indicating the length of the sequence, and k ($1 \leq k \leq 10^9$), indicating the value of k for the input case.

The following input line contains n space separated integers. The i^{th} of these integers is a_i ($1 \leq a_i \leq 10^9$), the i^{th} value of the input sequence.

Output

Print the length of the longest k-gap subsequence of the input sequence.

Example

Input	Output
10 2 1 2 3 2 1 3 1 3 5 6	7
5 12 3 7 14 20 32	3
13 4 3 12 8 4 2 5 1 9 8 6 5 1 7	8

Problem J. Ultimate Commitment Forever

Source file name:	Ultimate.c, Ultimate.cpp, Ultimate.java, Ultimate.py
Input:	`Standard`
Output:	`Standard`

UCF has been continuously placing high in various university rankings and, as such, more students would like to attend UCF. To provide the best education environment, new buildings are added to the campus master plan and, as such, we see construction at various sites.

For the purposes of this problem, treat UCF's campus as a grid with Cartesian Coordinates. Each location is a lattice point and one can travel one unit north, south, east, or west from a lattice point to the adjacent lattice points. There is construction at some lattice points and these can't be crossed when traveling between locations.

The Manhattan distance between any two locations $(x_1,\ y_1)$ and $(x_2,\ y_2)$ is $|x_1 - x_2| + |y_1 - y_2|$. When traveling between two locations, you only allow yourself to travel the Manhattan distance between the two locations; this implies that each step should get you closer to the destination. For example, if the destination is to the northwest of the starting location (i.e., you are traveling in the northwest direction), the only steps you allow yourself to take are moving one unit north or one unit west (the only exception is when the constructions force you to "go around").

You'd like to know the number of different ways you can make several journeys, assuming that you're only willing to travel the Manhattan distance between the locations and are avoiding all intersections that are under construction. Since this number could be quite large, calculate it modulo $10^9 + 7$.

Given a list of locations that are under construction, as well as several trips denoted by pairs of ordered pairs of locations, determine the number of ways to make each of those trips, avoiding any construction intersections while traveling the Manhattan distance between the locations.

Input

The first input line contains an integer, n $(0 \leq n \leq 10)$, representing the number of locations under construction. Each of the next n input lines contains a pair of space separated integers, x $(0 \leq x \leq 5 \cdot 10^5)$ and y $(0 \leq y \leq 5 \cdot 10^5)$, indicating that the location $(x,\ y)$ is under construction. Assume that all construction locations are distinct, i.e., no duplicates.

The next input line contains an integer, t $(1 \leq t \leq 10^4)$, representing the number of trips you plan on taking on campus. Each of the following t input lines contains four space separated integers: x_1, y_1, x_2, y_2 $(0 \leq x_1,\ y_1,\ x_2,\ y_2 \leq 5 \cdot 10^5)$, representing a trip to evaluate from $(x_1,\ y_1)$ to $(x_2,\ y_2)$. It is guaranteed that neither the starting nor ending point of the trip will be one of the construction locations, and the starting and ending locations will be distinct as well. Assume that all the trips will have a Manhattan distance greater than 0.

Output

For each trip to evaluate, print a single line with the number of ways the trip can be made while avoiding construction locations and traveling only the Manhattan distance between the two locations, modulo $10^9 + 7$. Two paths for making a trip are different if they differ in a particular step, e.g., `NN`, `NW`, `WW`, `WN` are all different.

Example

Input	Output
3 2 4 4 2 5 1 2 0 0 4 4 3 2 3 100	40 1
2 1 1 2 2 2 0 0 3 3 3 7 7 5	4 15

Problem K. Really Nerdy Game

Source file name:	Really.c, Really.cpp, Really.java, Really.py
Input:	`Standard`
Output:	`Standard`

Ali is playing a board game by himself. The board's track consists of N locations (cells, spots) laid out in a circle, i.e., the board cell N is followed by the board cell 1. At every step in the game, Ali rolls a die with k sides and moves his piece forward (clockwise) as many cells (spots) as it is indicated by the die. The die's sides have the value 1, 2, 3, ..., k (each occurring exactly once), and each side is equally likely to come up.

Ali starts his piece at location 1, and when Ali moves his piece past location N, Ali's piece continues its movement to locations 1, 2, etc. (again, the board is circular).

There are several spots on the board that are instant win cells, and there are several spots on the board that are instant loss cells. Reaching either one of these spots ends the game immediately. Any spot that is not an instant win or an instant loss allows the game to continue.

Ali wants to know the probability that he will end up winning. Since Ali is very precise, Ali wants to know the fraction of winning as p/q. Since p and q can be quite large, compute the answer as $p \cdot \left(q^{-1}\right) \mod 10007$ instead, where q^{-1} denotes the multiplicative inverse of q modulo 10007. (Math Refresher on Multiplicative Inverse is on the next page.)

Given the layout of the board and the die used, determine the probability of winning.

Input

The first input line contains four integers: n ($1 \leq n \leq 50$), indicating the number of spots on the board, d ($1 \leq d \leq 120$), indicating the number of sides on the die, w ($0 \leq w < n$), indicating the number of winning spots on the board, and l ($0 \leq l < n \breve{} w$), indicating the number of losing spots on the board.

Each of the next w input lines contains an integer (between 2 and n, inclusive) indicating a winning location. Each of the next l input lines contains an integer (between 2 and n, inclusive) indicating a losing location.

Assume that all the winning locations are distinct, all the losing locations are distinct, no location will be both a winning and losing spot, and location 1 will not be winning or losing.

Output

Print the integral probability of winning p/q as $p \cdot \left(q^{-1}\right) \mod 10007$, where q^{-1} denotes the multiplicative inverse of q modulo 10007.

Example

Input	Output
4 6 1 1 2 4	8578
5 2 1 0 2	1

Explanation

The answer to the first case is 4/7.

Math Refresher (Definition of Multiplicative Inverse):

Another name for Reciprocal.

What you multiply by a number to get 1.

Example: $8 \cdot \frac{1}{8} = 1$

In other words: when we multiply a number by its "Multiplicative Inverse", we get 1.

But not when the number is 0 because 1/0 is undefined!

Problem L. Brownies vs. Candies vs. Cookies

Source file name:	Brownies.c, Brownies.cpp, Brownies.java, Brownies.py
Input:	`Standard`
Output:	`Standard`

Everyone is welcome to the UCF Programming Team practices, and many students take advantage of this opportunity. The main benefit is that these students improve their problem solving and programming skills. Another benefit is that the students enjoy the refreshments Dr. Orooji brings every week! Dr. O usually brings candies but sometimes he brings cookies or brownies. Brownies are very popular and don't usually last long, so Dr. O has to come up with some clever trick to make the brownies last longer (so that the students stay for the entire practice!). Well, the easiest solution is to cut the brownies in half; that will double the number of brownies.

Given the original number of brownies and the students wanting brownies, you are to keep track of the brownie count as Dr. O cuts them in half.

Input

The first input line contains a positive integer, n, indicating the number of programming team practices. This is followed by the data for these practices. The first input line for each practice contains two integers (separated by a space): the number of students (between 1 and 30 inclusive) in the practice and the number of brownies (between 60 and 600 inclusive) Dr. O has brought that day. The next input line for the practice contains a positive integer, m, indicating how many groups of students approach the refreshment table to take brownies. This is followed by the number of students in each group, one number per line. Assume that the input values are valid, e.g., the number of students in a group will be at least 1 and it will not be greater than the number of students in the practice.

If a group of students is approaching the refreshment table and Dr. O notices that the number of remaining brownies is less than or equal to the number of students in the group, Dr. O cuts the brownies in half to be sure they won't be all gone after each student in the group grabs one brownie. Note that, if needed, Dr. O will cut the brownies more than once (as many times as needed). For example, if there are 3 brownies left and 24 students are approaching the table, Dr. O has to cut the brownies four times ($3 \rightarrow 6 \rightarrow 12 \rightarrow 24 \rightarrow 48$) to be sure the brownies won't be all gone after each student in the group grabs one.

Output

At the beginning of each practice, output "`Practice #p: s b`" where p is the practice number (starting with 1), s is the number of students in this practice, and b is the number of brownies. Then, on each of the following output lines, print the number of students in a group approaching the refreshment table and the number of brownies left after each of these students has grabbed one brownie (note that cutting in halves may occur before grabbing).

Example

Input	Output
2 20 60 8 15 10 20 18 9 12 2 10 15 100 4 1 2 3 5	Practice #1: 20 60 15 45 10 35 20 15 18 12 9 3 12 12 2 10 10 10 Practice #2: 15 100 1 99 2 97 3 94 5 89

Problem M. Turing's Challenge

Source file name:	Turing.c, Turing.cpp, Turing.java, Turing.py
Input:	`Standard`
Output:	`Standard`

Knuth was looking through some of Turing's memoirs and found a rather interesting challenge that Turing had left for one of his successors. Naturally, Knuth has slyly decided to ask you, his best student, to write a computer program to solve the challenge, but plans on taking credit for the work. Since you know that co-authoring a paper with Knuth is to computer scientists what co-authoring a paper with Erdos is to mathematicians, you've decided to take the bait. Help Knuth solve Turing's problem!

The challenge is as follows:

Given positive integer values for X and N, define the set T as follows:

$$T = \{T_i \mid 1 \leq i \leq N+1\}, \text{ where } T_i = \binom{N}{i-1} \cdot X^{i-1}$$

The goal of the challenge is to pick a set S of maximal sum with $S \subseteq \{i \mid 1 \leq i \leq N+1\}$, such that $\prod_{i \in S} T_i \equiv 2 (\text{mod } 4)$.

In other words, we seek a subset of terms in the binomial expansion of $(1+X)^N$ such that the product of the terms leaves a remainder of 2 when divided by 4 and the sum of the *indices* of those terms is maximal.

The goal of Turing's challenge is to determine this maximal sum.

As an example, consider $X = 3$ and $N = 5$. The corresponding terms are $T_1 = 1$, $T_2 = 15$, $T_3 = 90$, $T_4 = 270$, $T_5 = 405$, and $T_6 = 243$.

The product, $T_1 \cdot T_2 \cdot T_4 \cdot T_5 \cdot T_6 = 1 \cdot 15 \cdot 270 \cdot 405 \cdot 243 = 398580750 \equiv 2 (\text{mod } 4)$ thus the solution to this specific challenge is $1 + 2 + 4 + 5 + 6 = 18$, since no other product of terms with a higher sum of indices is congruent to 2 (mod 4).

Input

The first input line contains a positive integer, q ($1 \leq q \leq 500$), indicating the number of queries. Each of the next q lines will contain a pair of space-separated integers, where the first integer is X $\left(1 \leq X < 2^{31}\right)$, and the second integer is N $\left(1 \leq N < 2^{31}\right)$, for that query.

Output

For each query, output on a line by itself, the desired maximal sum of indices. If no such subset of terms exists, output 0 instead.

Example

Input	Output
3 3 5 1 4 4 6	18 9 0