National Programming Contest AKA Hackathon organized on the occasion of World Telecommunication and Information Society Day

Organized by

Bangladesh Submarine Cable Company Limited (BSCCL)

Hosted by

University of Information Technology and Science (UITS)

11 Problems

300 Minutes

20 Pages

Rules for the Contest:

a) Solutions to problems submitted for judging are called runs. Each run is judged as accepted or rejected by the judge, and the team is notified of the results.

b) Notification of accepted runs will **NOT** be suspended at the last one hour of the contest time to keep the final results secret. Notification of rejected runs will also continue until the end of the contest. But the teams will not be given any balloon and the public rank list will not be updated as usual in the last one hour.

c) A contestant may submit a clarification request to judges only through the CodeMarshal clarification system. If the judges agree that an ambiguity or error exists, a clarification will be issued to all contestants. Judges may prefer not to answer a clarification at all in which case that particular clarification request will be marked as IGNORED in the CodeMarshal clarification page.

d) Contestants are not to converse with anyone except members of their team and personnel designated by the organizing committee while seated at the team desk. **They cannot even talk with their team members when they are walking around the contest floor to have food or any other purpose.** Systems support staff or judges may advise contestants on system-related problems such as explaining system error messages.

e) While the contest is scheduled for a particular time length (five hours), the chief judge or the judging director has the authority to alter the length of the contest in the event of unforeseen difficulties. Should the contest duration be altered, every attempt will be made to notify contestants in a timely and uniform manner.

f) **A team may be disqualified by the** chief judge or the judging director for any activity that jeopardizes the contest such as dislodging extension cords, unauthorized modification of contest materials, distracting behavior or communicating with other teams. The **judges on the contest floor** will report to the **Judging Director** about distracting behavior of any team. **The judges can also recommend penalizing a team with additional penalty minutes for their distracting behavior.**

g) Eleven problems will be posed. So far as possible, problems will avoid dependence on detailed knowledge of a particular applications area or particular contest language. Of these problems at least one will be solvable by a first year computer science student, another one will be solvable by a second year computer science student and rest will determine the winner.

h) Contestants will have foods available in their contest room during the contest. So they cannot leave the contest room during the contest without explicit permission from the judges. **The contestants are not allowed to communicate with any contestant (even contestants of his own team) or coaches when they are outside the contest arena.**

i) Teams can bring **printed materials** with them and they can also bring five additional books. But they are not allowed to bring calculators or any machine-readable devices like CD, DVD, Pen-drive, IPOD, MP3/MP4 players, floppy disks etc. **Mobile phone MUST be switched off at all times and stored inside a bag or any other place that is publicly non visible during the entire contest time. Failure to adherence to this clause under any condition will very likely lead to strict disciplinary retaliation and possible disqualification.**

j) With the help of the volunteers, the contestants may have printouts of their codes for debugging purposes. **Passing of printed codes to other teams is strictly prohibited.**

k) The decision of the judges is final.

l) Teams should inform the volunteers/judges if they don't get verdict from the codemarshal within 5 minutes of submission. Teams should also notify the volunteers if they cannot log in into the CodeMarshal system. This sort of complains will not be entertained after the contest.

A Square Free Divisors of Factorials

Input: Standard Input
Output: Standard Output

Factorial n or n! is defined as below:

$$n! = 1*2*3*...*n$$

For example, 7! = 1*2*3*4*5*6*7 = 5040. Square free numbers are the numbers which do not have a square divisor greater than 1. For example 35 is a square free number, but 44 is not (as 4 is one of its divisors which is also an square number). 1 is also considered a square free number. The function SFDF(n) (Square free divisors of factorial) denotes how many square free divisors n! has. The function CSFDF(c) or cumulative SFDF is defined as:

$$CSDF(n) = \sum_{1}^{n} SFDF(i)$$

Given the value of n your job is to find the modulo 1000007 value of CSFDF(n).

Input

First line of the input file contains an integer T (T≤10000) which denotes the total number of test cases. Each of the next T lines contain a positive integer which denotes the value of n (n<10000001).

Output

For each line of input produce one line of output. This line contains the modulo 1000007 value of CSFDF(n).

Sample Input	Output for Sample Input
2 1 11	1 123

B	**Referral Scam** **Input:** Standard Input **Output:** Standard Output	

Dan designed an eWallet website for users to digitally store their shopping money.

- A user can refer any number of friends with a referral link.
- Each time a friend signs up and loads at least $100 dollars into their account, the referrer gets a $25 dollar bonus.
- A user can *get referred* by at most one friend.

Lucy is a hacker looking to game the system. She realized that Dan's website has following security holes:

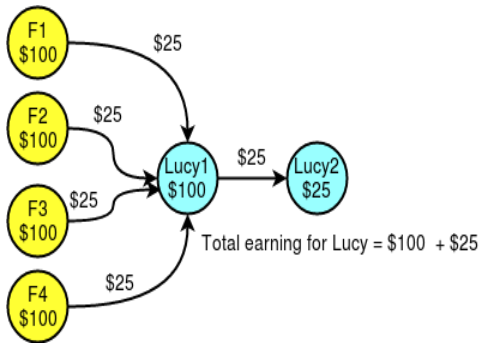
- It can't detect the difference between the money earned by referral and the money loaded by the user.

- Due to lack of verification system, it can't prevent a single person to create multiple accounts under different names.

Let's say Lucy has four friends ***F1***, ***F2***, ***F3*** and ***F4***. Lucy created an account called ***Lucy1*** and referred her friends. Then she created another account called ***Lucy2*** and referred her first account ***Lucy1***.

Now when each of the four friends signed up and loaded $100 dollars in their account, ***Lucy1*** got $25 * 4 = $100 as referral bonus. But as ***Lucy1*** was referred by ***Lucy2***, now ***Lucy2*** also gets $25 bonus!

So total earning for Lucy in two accounts is $100+25=$125.



You are given an integer x. Your task is to find the minimum number of "real" friends Lucy must refer to earn at least $x.

Input

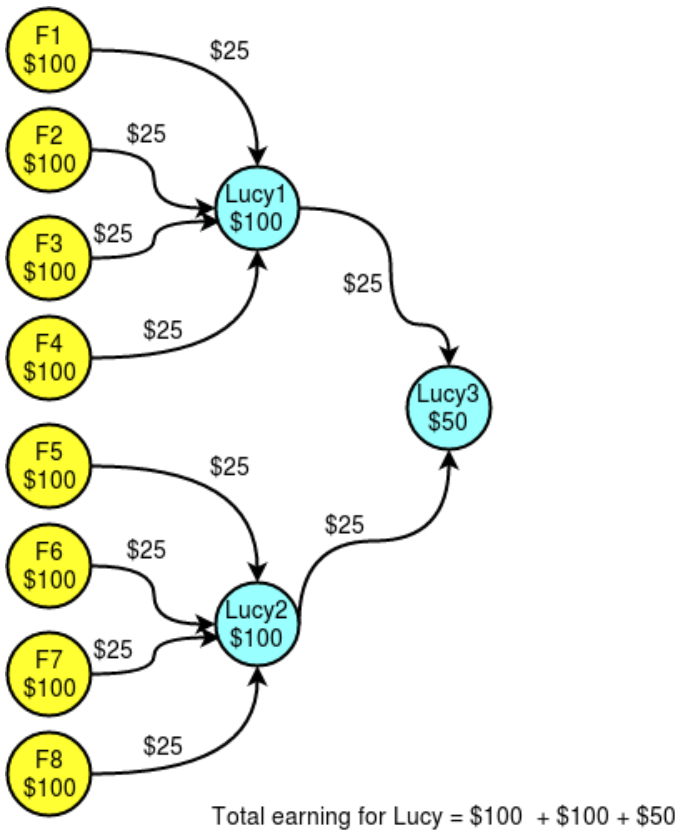
The first line contains **t (1≤t≤5*10^4)**, number of test cases. Each test case contains a single line containing an integer **x (1≤x≤10^17)**.

Output

For each case, print the answer on a single line.

Sample Input	Output for Sample Input
2 45 250	Case 1: 2 Case 2: 8

Explanation of test case 2:



C Multiplying Very Sparse Matrix

Input: Standard Input
Output: Standard Output

Multiplying matrix by hand is a tedious job but we can do it very easily by writing a program. if two matrix **A** and **B** are to multiplied then the number of columns of the first matrix must be equal to the number of rows of the 2nd matrix. A sparse matrix is a matrix in which most of the elements are zeroes. For this problem we define Very Sparse Matrix **(VSM)** as a matrix whose at most **2%** elements are non-zero. Given the description two large **VSMs** your job is to multiply them quite efficiently.

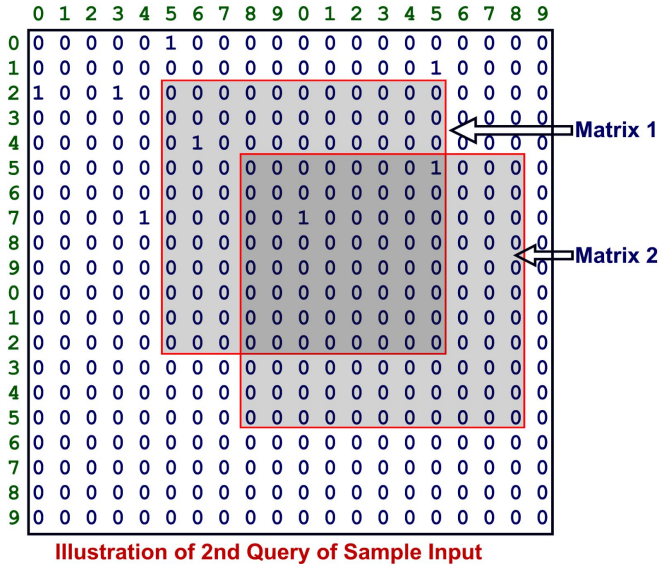


Illustration of 2nd Query of Sample Input

Input

First line of the input file contains an integer **N (0 < N ≤ 2000)**. This integer denotes that an **(N * N)** master-grid will follow. All matrices of this problem will be a rectangular region from this master-grid. Each of the next **N** lines contains **N** randomly generated integers separated by a single space. The integers are either 0 or 1 and it is guaranteed that not more than 2% integers are 1. The rows of the master-grid are numbered from **0** to **N-1** and the columns of the master-grid are also numbered from **0** to **N-1**.

First line after the master grid contains an integer **Q (0 < Q ≤ 200)** which denotes the total no of query. Next **2Q** lines contains **Q** queries so each query consists of **2** lines. The description of each query is given below:

First line of each query consists of four integers $\mathbf{start\_row_1}$, $\mathbf{start\_scol_1}$, $\mathbf{end\_row_1}$, $\mathbf{end\_col_1}$ ($0 \le \mathbf{start\_row_1} \le \mathbf{end\_row_1} < \mathbf{N}$ and $0 \le \mathbf{start\_col_1} \le \mathbf{end\_col_1} < \mathbf{N}$) These four integers denote that the first (topmost) and last (bottom most) row of the first matrix is numbered $\mathbf{start\_row_1}$ and $\mathbf{end\_row_1}$ respectively and first (leftmost) and last (rightmost) column of the first matrix is numbered $\mathbf{start\_col_1}$ and $\mathbf{end\_col_1}$ respectively. Second line of each query consists of four integers $\mathbf{start\_row_2}$, $\mathbf{start\_scol_2}$,

end_row$_2$, **end_col$_2$** ($0 \le$ **start_row$_2$** $\le$ **end_row$_2$** $<$ **N** and $0 \le$ **start_col$_2$** $\le$ **end_col$_2$** $<$ **N**). These four integers denote that the first (topmost) and last (bottom most) row of the second matrix is numbered **start_row$_2$** and **end_row$_2$** respectively and first (leftmost) and last (rightmost) column of the second matrix is numbered **start_col$_2$** and **end_col$_2$** respectively. The queries are also randomly generated. You can assume that number of column of first matrix and number of row of second matrix will always be same.

Output

For each query you need to multiply the two matrices specified by the queries. But you do not need to produce the result matrix as output. You only need to output an integer which is the summation of all the elements of the answer matrix.

Sample Input

```
20
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3
0 0 5 5
0 0 5 5
2 5 12 15
5 8 15 18
0 0 19 19
0 0 19 19
```

Output for Sample Input

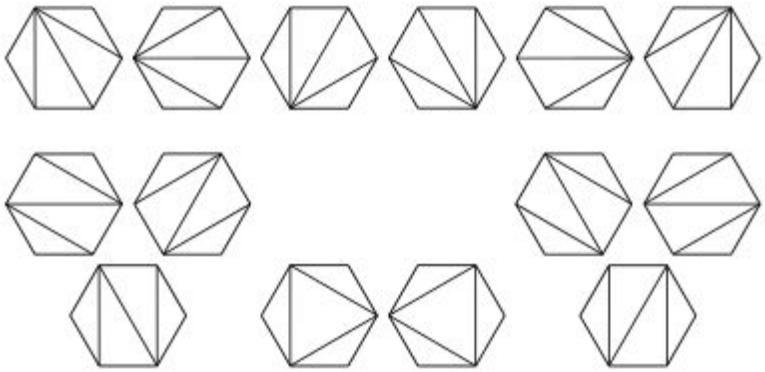
```
1
0
3
```

D Polygon Shopping

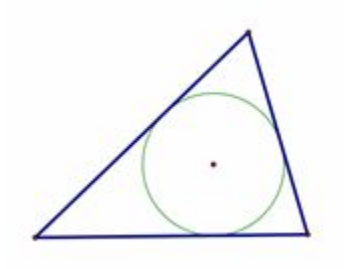
Input: Standard Input
Output: Standard Output

Soruba went shopping one day. When looking around he entered a shop that sells decorations. The shop sold many different kind of decorations. Suddenly, Soruba saw a shelf filled with polygons. He got closer to inspect and saw that they were not ordinary polygons.

1. The polygons were convex and regular, containing N sides of length 1.
2. They were triangulated. Meaning, they were made of triangles with non-intersecting area. See image below.
3. All of them were unique. No triangulation is repeated.
4. Each triangle has circle inside of it which is touching all three sides. See image below.



Picture of Triangulated Polygons



Incircle inside Triangle

The shopkeeper explained that these decorations were made for a Math Professor that wanted to explain Polygon Triangulation to his students. The circles inside were not part of triangulation and made only to support the structure.

Upon inquiring what they are doing on self, the shopkeeper further explained that they were donated to the shop by the Professor. The shopkeeper is going to sell them on a discount (he won't charge for the frame, only for the support structure), since they are second-hand items.

Soruba wants to buy the whole set. The shopkeeper says he doesn't remember the discounted price but what he remembers is that each of the polygon has a cost equal to the sum of all the diameters of circles supporting it. So the total cost will be sum of cost of each polygon.

So given the value of N, find the cost of the Polygon set.

Input

The first line of input will contain single positive integer T (T≤20) which is number of test case. For each case, there will be single line containing a positive integer N ($3 \leq N \leq 20$).

Output

For each case, print the case number first, and then the cost of the Polygon Set rounded to 3 decimal place. See sample I/O for more details.

Sample Input	Output for Sample Input
2 3 4	Case 1: 0.577 Case 2: 2.343

E Star Systems

Input: Standard Input
Output: Standard Output

The famous astronomer Professor Shonku is studying planets in a faraway galaxy. In his work, he performs measurements on millions of planets. That means his calculations have to be quite efficient, and in this he needs your help.

Professor Shonku has determined that the planets follow some of the laws of the planets in our solar system but differ in some interesting ways. Here is what he knows about the planets under observation:

1. Each planet orbits its sun on an elliptical path given by a formula of the form shown below but unlike our solar system, the sun is at the center (0, 0) of this ellipse.
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$
2. The speed of the planet on its path follows a variant of Kepler's law. Given a fixed time t (for example 1 hour), the elliptical segment traversed within time t will have the same area, independent of the initial location of the planet on its path. Here, the elliptical segment is defined as the region bounded by the arc of the ellipse between the start and end positions of the planet and the lines from the sun to these two positions.
3. Planets orbit in counterclockwise direction, and a full orbit takes an integral number of hours.

Professor Shonku is trying to determine the time it takes a planet to orbit its sun by monitoring the path of the planet. He knows that the time for a full orbit is at least 10000 hours and at most 200000 hours. Since this means that an orbit can take up to 23 years, Professor Shonku cannot wait to observe a full orbit. He needs your help in computing the orbit duration from several observations.

Input

First line of the input file contains a positive integer T ($T \leq 25$) which denotes the number of test cases. Each test case consists of three lines. Each of the three lines contains two floating-point numbers, which are the coordinates (x_1, y_1), (x_2, y_2), and (x_3, y_3) of three observations of the location of a planet in the order which they were taken. Each number is given with 14 digits to the right of the decimal point. These observations were taken at times t_1, t_2 and t_3 respectively. The durations $t_2 - t_1$ and $t_3 - t_2$ are exact multiples of 1 hour, and are between 1000 and 2000 hours inclusive. Moreover, the parameters a and b of the elliptical path satisfy ($0.1 \leq a/b \leq 10$).

Output

Display the orbit duration in hours. If the input allows for multiple possible orbit durations, display the smallest one. **You can assume that there will always be a valid solution.**

Sample Input	Output for Sample Input
2 17285.21988721138800 463.67872970687591 17100.78077891304200 665.93761652637556 16684.03664426841700 975.03398761156586 15753.53919213167100 4582.18602410143470 12940.97854473381300 5982.95269561511300 9522.90812162402840 7062.28250041092180	106578 40741

Warning: This problem is a bit unstable floating-point wise, so be careful.

F	**Randomized Nim** **Input:** Standard Input **Output:** Standard Output	

Alice has **n (n≤8)** piles, each containing **1** to **8** stones. In each move she can choose a pile and remove a positive number of stones from it. Probability of choosing any pile is same. Also probability of removing any number of stones from a pile is also same.

Find the expected number of moves it will take for Alice to remove all the stones.

Input

The first line contains ***T (1≤T≤50000)***, number of test cases. Each test case contains two lines. The first line contains ***n***, the number of piles. The second line contains ***n*** integers, the number stones in each piles.

Output

Print the expected number of moves as a reduced fraction a/b.

Sample Input	Output for Sample Input
4 1 1 1 2 2 1 2 2 2 2	Case 1: 1/1 Case 2: 3/2 Case 3: 5/2 Case 4: 3/1

Explanation

Let's look at test case 2. There is only one pile of size 2. Alice has two possible options:

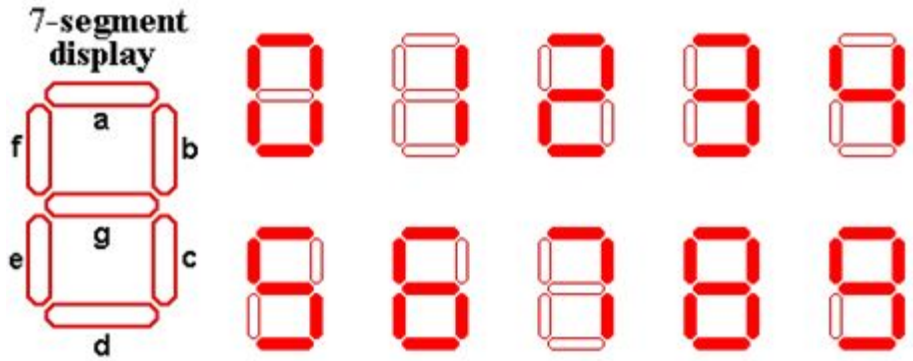
- Remove both stones in a single move.
- Make two moves, remove one stone in each moves.

In the first option, Alice needs just one move but in the 2nd option she needs two moves. The probability of choosing any option is same. So the expected number of moves is (1+2)/2 = 3/2.

G Seven Segment Display

Input: Standard Input
Output: Standard Output

We all are very much familiar with the idea of a seven-segment display, aren't we?



In the above picture you can see all seven segments of a seven-segment display named as- 'a', 'b', 'c', 'd', 'e', 'f' and 'g'. If a particular segment is lighted we call it as ON segment, otherwise it is called OFF segment.

Digit Display	ON Segments	OFF Segments
0	a, b, c, d, e and f	g
1	b and c	a, d, e, f and g
…	… …	… …
8	a, b, c, d, e, f and g	---
9	a, b, c, d , f and g	e

In this problem, you can switch OFF any ON segment of a display. For example, if you switch OFF 'g' and 'd' segment of 3, it becomes 7. Note that, you cannot switch ON any OFF segment of a seven-segment display.

You have x_0 Zeros(0), x_1 Ones(1), x_2 Twos(2), …, x_9 Nines(9) digit and a non-empty string of digits. Find the maximum length of the String that can be displayed from the beginning with minimum cost. Your first target is to maximize length and then minimize total cost. It takes 1$ to switch OFF one ON segment.

Input

Input starts with an integer **T** , denoting the number of test cases.

The first line of each case starts with 10 space separated integers x_0, x_1, x_3, …,x_9 . Next line contains a nonempty string of digits. There is a blank line between two consecutive input cases.

Constrains:
$1 \leq T \leq 1500$
$0 \leq x_0, x_1, x_3, \ldots, x_9 \leq 10000$
$1 \leq$ String Length ≤ 10000
You may safely assume that all inputs are valid and the string is composed of digits only.

Output

For each of the test cases, you need to print desired answers. You must follow the exact format as sample output.

Sample Input	Output for Sample Input
4 1 1 1 1 1 1 1 1 1 1 0123456789 1 1 1 1 1 1 1 1 1 1 83811 0 0 0 2 1 1 1 0 1 0 37108123456789 0 0 0 0 0 0 0 0 1 0 12348	Case 1: 10 0 Case 2: 2 0 Case 3: 4 5 Case 4: 1 5

Explanation:
Case 1: You can display the string without any change.
Case 2: You can display only first two digits.
Case 3: You can display first 4 digits only. One of the possible way is shown as follows:
3: Use one of the 3 digit display with 0$ (no change required)
7: Convert 3 into 7 and it requires 2$.
1: Convert 4 into 1 and it requires 2$.
0: Convert 8 into 0 and it requires 1$.
So, you can display first 4 digits and it requires total 0$+2$+2$+1$=5$.
Case 4: Convert 8 into 1 and it requires 5$.

H Too Long, Don't Read

Input: Standard Input
Output: Standard Output

In the middle of a working day as Tuna was feeling very sleepy, he decided to take a nap in a sleeping cubicle. Soon after he lied down, he could hear a sound of an alarm from another nearby cubicle. Tuna had no doubt that it was due to his friend Sardine, who had this bad habit of continuing to snooze her alarm, instead of just stopping it once and for all. After few minutes Tuna realized that he cannot sleep anymore anyway, so instead he tried to solve the following problem.

Let's imagine that the timeline is **T** seconds (current time is T=0). Sardine set her alarm at some time **t,** which is unknown to Tuna, but it is guaranteed to be an integer in the range of **[1,T]**. What Tuna knows for sure (from past experience), is Sardine likes to listen to the alarm sound for arbitrary seconds, but it cannot be more than **L** seconds (because that is the duration of her alarm tone). Note that each time the alarm starts ringing, it has to keep ringing for at least 1 second. After Sardine snoozes the alarm, it takes **D** seconds for the alarm to start ringing for the next time. The alarm is going to ring for maximum **R** number of times.

For example, let's say T = 13, L = 6, D = 3. First time the alarm rings is when t = 1, and Sardine chooses to listen to it for 4 seconds. So she snoozes the alarm when t = 5. So the next time the alarm will start ringing is when t = 8, and so on. Note that, If the alarm is ringing at the end of the timeline (i.e. at the T'th second), then the alarm just stops ringing automatically, and doesn't start ringing again for the entire day.

Tuna is wondering, what is the expected number of times the alarm is going to ring in this **T** second long timeline. He is very very sleepy, so he's asking for your help.

Input

Input starts with an integer **C (1 ≤ C ≤ 100)** denoting the number of test cases and C cases follow.
Each case has 4 integers, **T, L, D** and **R (1 ≤ T ≤ 10000, 1 ≤ L, D ≤ 1000** and **1 ≤ R ≤ 100)**.

Output

For each case, output the case number first. Followed by the expected number of times the alarm is going to ring (rounded to 7 decimal places).

Sample Input	Output for Sample Input
2 1 1 1 1 3 1 1 2	Case 1: 1.0000000 Case 2: 1.3333333

I Light up the Museums

Input: Standard Input
Output: Standard Output

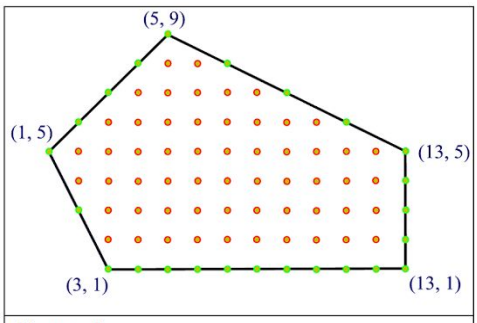


Figure 1:

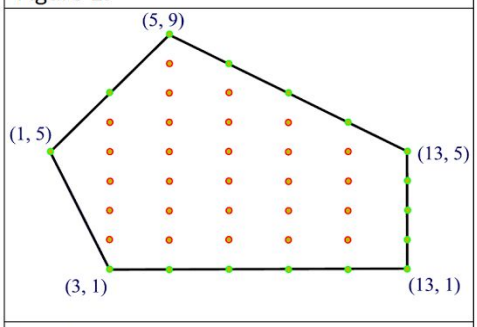


Figure 2:

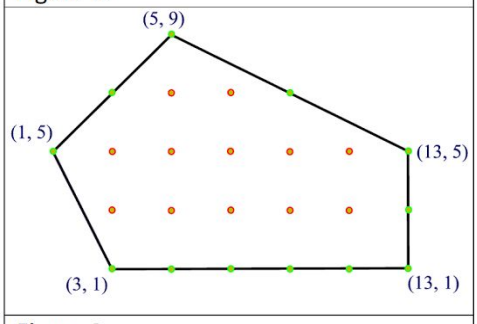


Figure 3:

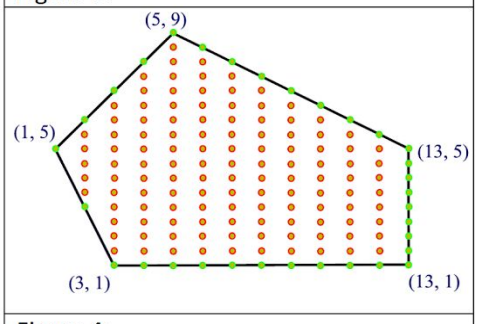


Figure 4:

The Museums of Ekaterinburg, Russia are going through a lot of renovation and construction works because world's most talented programmers are going to visit the museums soon. The renovation works include (a) Painting (b) Building new buildings (c) Fitting lights in these new Buildings.

Newly built museums have shapes of non-intersecting simple convex-polygons and they are described with the Cartesian coordinates of the corners of the polygon in clockwise or counter-clockwise order. Lights are placed in the museum following some specific rules. These are described below:

1. Lights are placed in rows and columns. The rows are always parallel to x axis and columns are parallel to y-axis.

2. The distance between two consecutive lights on the same row are equal and we call this distance d_{row}. Similarly, the distance between two consecutive lights on the same column are also equal and we call this distance d_{col}.

3. Lights must be placed in such a way that a light is placed in each corners of the museum. This is shown in the figures on the left. Lights can be placed on the boundary but never outside the museum.

4. The authorities of all the Museums want the lighting to be as uniform as possible but light bulbs do not always project lights equally in all directions. That is why The value of d_{row} and d_{col} may or may not be equal. The value $\frac{d_{row}}{d_{col}}$ is called light placement aspect ratio or LPAR. The LPAR of Figure 1, Figure 2, Figure 3 and Figure 4 is $\frac{1}{1}$, $\frac{1}{2}$, $\frac{1}{1}$ and $\frac{1}{2}$ respectively. So the value of d_{row} and d_{col} actually decides how many lights will be needed for the whole museum but LPAR cannot determine how many lights will be needed (Figure 1 and 3 has same LPAR and Figure 2 and 4 has same LPAR but still the bulb requirement is different).

5. Lights are of two types (i) Internal lights (Lights that are fit strictly inside the museum - shown as red circles in figures above) (ii) Boundary lights (Lights that are fit strictly on the boundary - shown as green circles in the figures above).

6. All the internal Lights fit in a museum must be identical.

Recently, you have been given the charge of this project. But there was someone else in the charge of this project before you and he had bought different sets of internal lights for different museums. Each

set contains an specific amount of identical internal lights. But the problem is that there is no record of which set of light was bought for which museum or what was the possible value of d_{row} and d_{col} for buying this amount of lights. But the thing you know is that the value of LPAR can always be expressed with a fraction $\frac{a}{b}$, where a and b are integers and ($1 \leq a, b \leq 10$). Given the description of the shape of the Museum and number of internal lights in each set bought, you will have to find out which set of lights was actually bought for the museum. You will also have to find the possible value of LPAR.

Input

First line of the input file contains a positive integer T ($0<T<10$) which denotes the number of test cases to follow. The description of a test case is given below:

First line of each test case contains two integers N ($3 \leq N \leq 10$) and S ($3 \leq S \leq 10$). Here N is the number of sides of the convex polygon P that represents the shape of the museum, and S is the total number light sets. Each of the next N lines contains two integers ($0 \leq x_i, y_i \leq 500$). These integers actually denote the Cartesian coordinates the vertices of polygon P in clockwise or counter-clockwise order. Each of the next S lines contains an integer B_j ($3 \leq B_j \leq 10^{12}$), here B_j is the number of lights in the j-th set.

Output

For each test case produce zero or more lines of output. If for any possible value of d_{row} and d_{col} exactly B_j lights is needed for the museum you have to print 2 values: the value of LPAR in format a/b (Here a and b are integers and relative prime and ($1 \leq a, b \leq 10$)) and value of B_j. If more than one values of B_j is suitable report all of them in similar fashion in the order they appear in the input. If one value of B_j is suitable for the Museum for more than one value of LPAR report them all in similar fashion in increasing order of LPAR. Look at the output for sample input for details. **Print a blank line after the output of each test case.**

Sample Input	Output for Sample Input
2 5 3 1 5 3 1 13 1 13 5 5 9 57 100 27 5 3 1 5 3 1 13 1 13 5 5 9 57 100 27	1/1 57 1/2 27 1/1 57 1/2 27

*Figure 1, 2 are related with the sample input.

J	Wormholes! Not Again **Input:** Standard Input **Output:** Standard Output	

The scientist has came up with another beautiful research on wormholes. I think you have forgotten about wormholes once again. Let me explain again.

A wormhole is a subspace tunnel through space and time connecting two star systems. Wormholes have a few peculiar properties:

1. Wormholes are bi-directional.
2. The time it takes to travel through a wormhole is negligible.
3. A wormhole has two endpoints, each situated in a star system.
4. A star system may have more than one wormhole end point within its boundaries.
5. Between any pair of star systems, there is at most one wormhole.
6. There are no wormholes with both endpoints in the same star system.

All wormholes have a constant time difference between their endpoints. For example, a specific wormhole may cause the person traveling through it to end up 15 years in the future. Another wormhole may cause the person to end up 42 years in the past.

Last time, the scientist wanted to use wormholes to study the Big Bang. Since warp drive has not been invented yet, it is not possible for him to travel from one star system to another one directly. This Can be done using wormholes, of course.

The scientist can start his journey from any star system. Then he wants to reach a cycle of wormholes somewhere in the universe that causes him to end up in the past. By traveling along this cycle a lot of times, the scientist is able to go back as far in time as necessary to reach the beginning of the universe and see the Big Bang with his own eyes. **A cycle of wormholes consists of three or more non-repeating star systems.**

Recently the scientist discovered that there is no wormholes cycle in our universe that can lead us to the Big Bang. But, as he is one of the best physicists ever born, he invented a device that can change a wormhole. With this device he is going to decrease the time difference between two endpoints of a wormhole, such that it becomes a wormhole cycle which can be used to see Big Bang. But the device need huge amount of power to run and it is proportional to decreasing amount of time difference. So he wants to decrease the time difference as minimum as possible.

Write a program to help the scientist to find for each wormhole the minimum time difference in years, that has to be decreased such that, the wormhole becomes part of a cycle that Big Bang can be seen using that cycle. Note that, you are not changing the time differences rather you are proposing the value. So for each wormhole's calculation you will work with the same unchanged star network or wormhole network.

Input

Input starts with an integer T (≤ 100), denoting the number of test cases.

Each case starts with two integer numbers n and m, in a single line. These indicate the number of star systems ($1 \leq n \leq 100$) and the number of wormholes ($0 \leq m \leq 10000$). The star systems are numbered from 1 to n. For each wormhole a line containing three integer numbers x, y and z is given. These numbers indicate that this wormhole allows someone to travel from the star system numbered x to the star system numbered y (and vice-versa), thereby ending up z ($0 \leq z \leq 1000$) years in the future or past, a negative integer denotes past, positive integer denotes future. Note that, wormholes are bi-directional.

Output

For each case, print the case number first. Then for each wormhole, print the minimum number of years to be decreased to be part of Big Bang cycle. If it is not possible, print 'N'.

Sample Input	Output for Sample Input
2 3 3 1 2 1 2 3 2 3 1 3 4 4 1 2 1 2 3 1 1 3 1 3 4 2	Case 1: 7 7 7 Case 2: 4 4 4 N

K Trip to World Finals

Input: Standard Input
Output: Standard Output

Moonty used to dream about participating in ACM ICPC World Finals from his childhood. Dream come true for him, as he is going to attend the ACM ICPC World Finals 2017 in Rapid City, South Dakota, USA.

Moonty is planning for his trip to USA as he is not only traveling to Rapid City, but also to Silicon Valley (another dream and secret one). He is very much excited about his trip. As he has started his packing for the trip, he faced a serious problem. According to his air ticket, he can take a luggage weighing at most 23 kg. If it exceeds the 23 kg limit, he will have to pay extra money. But he is trying to minimize the trip cost.

Write a program to help Moonty (though he can write one but he is busy with his trip planning). He will give you N of his items' weight, you will have to tell him whether he will have to pay or not.

Input

Input starts with an integer **T (≤ 100)**, denoting the number of test cases.

Each case starts with a single integer numbers **N**, number of item **(1 ≤ N ≤ 100)**. Next line will consists of **N** integer numbers $\mathbf{W_i}$, weight of all the items **(1 ≤ W_i ≤ 50)** in kg.

Output

For each case, print the case number and "**Pay**" if he has to pay extra money. Otherwise, print "**Do Not Pay**".

Sample Input	Output for Sample Input
2 5 1 2 3 4 5 2 10 15	Case 1: Do Not Pay Case 2: Pay