Problem A. Anniversary

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

In the year 2134, when the Earth population reached 10^{18}, the people decided to celebrate this event. N diminutive bulbs (the linear size of each bulb was less than 0.1 millimeters) were placed on a huge area specially allotted for the celebration. The bulbs were numbered with consecutive integers from 1 to N in some order.

In the beginning, all the bulbs were off. Afterwards, exactly 10^{18} steps were performed — one per each citizen of Earth. At the i-th step, the states of all bulbs with number X such that i divides X toggled at the same time. If a bulb is on, toggling its state would switch it off, and vice versa, toggling the state of an off bulb would switch it on. The interval between the consecutive steps was only 1 picosecond, so the whole celebration took around a week and a half.

Actually, all this stuff looked as pointless flickering, but the spectators were delighted by the colossal scale of the wonderful event. Finally, it was over. There was nothing more to look at, and — if you think soberly — nothing worthy ever happened.

In the meanwhile, after step 10^{18} some of the bulbs are still on. While the people are recovering from the shock, pondering why did they need such a celebration and who will cover its costs, we suggest that you count the number of bulbs which are still on and consume the precious power.

Input

The only line of input contains the integer N ($1 \leq N \leq 2^{63} - 1$).

Output

Print a single integer — the number of bulbs which are still on after step 10^{18}.

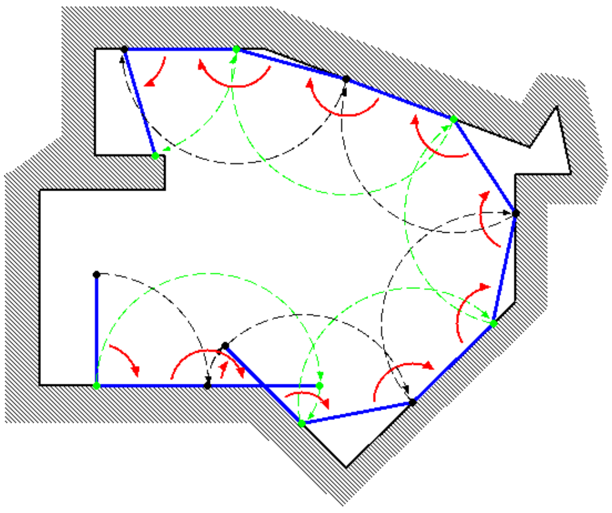
Example

standard input	standard output
1	1
9	3
100	10

Problem B. Bar Rotation

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

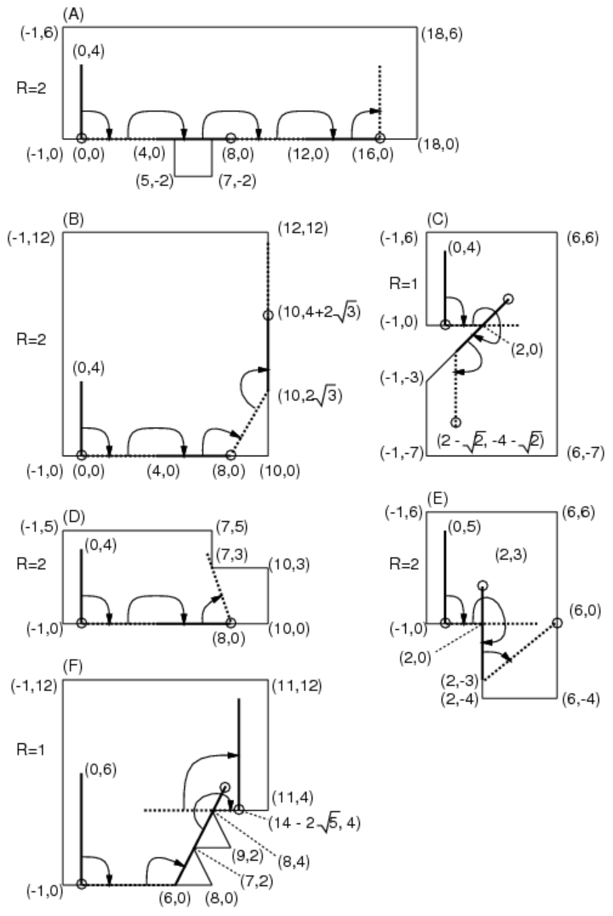
Let's think about a bar rotating clockwise as if it were a twirling baton moving on a planar surface surrounded by a polygonal wall (see figure below).



Initially, an end of the bar (called "end A") is at $(0, 0)$, and the other end (called "end B") is at $(0, L)$, where L is the length of the bar. Initially, the bar is touching the wall only at the end A.

The bar turns fixing a touching point as the center. The center changes as a new point touches the wall.

Your task is to calculate the coordinates of the end A when the bar has fully turned by the given count R.



In Figure above, some examples are shown. In cases (D) and (E), the bar is stuck prematurely (cannot rotate clockwise anymore with any point touching the wall as the center) before R rotations. In such cases, you should answer the coordinates of the end A in that (stuck) position.

You can assume the following:

When the bar's length L changes by a ($|a| < 0.00001$), the final (x, y) coordinates will not change more than 0.0005.

Input

The format of the input is as follows:

```
L R N
X_1 Y_1
X_2 Y_2
...
X_N Y_N
```

L is the length of the bar. The bar rotates $2 \cdot \pi \cdot R$ radians (if it is not stuck prematurely). N is the number of vertices which make the polygon.

The vertices of the polygon are arranged in a counter-clockwise order. You may assume that the polygon is simple, that is, its border never crosses or touches itself.

N, X_i and Y_i are integer; R and L are decimal fractions given with no more than 1 digit after decimal point. Ranges of those values are as follows: $1.0 \le L \le 500.0$, $1.0 \le R \le 10.0$, $3 \le N \le 100$, $-1000 \le X_i \le 1000$, $-1000 \le Y_i \le 1000$, $X1 \le -1$, $Y_1 = 0$, $X_2 \ge 1$, $Y_2 = 0$.

Output

Print one line containing x- and y-coordinates of the final position of the end A, separated by a space.

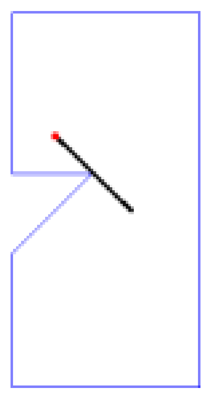
The value may contain an absolute error less than or equal to 0.001. You may print any number of digits after the decimal point.

Example

standard input	standard output
`4.0 2.0 8`	`16.0 0.0`
`-1 0`	
`5 0`	
`5 -2`	
`7 -2`	
`7 0`	
`18 0`	
`18 6`	
`-1 6`	
`4.0 2.0 4`	`9.999999999999998 7.4641016151377535`
`-1 0`	
`10 0`	
`10 12`	
`-1 12`	
`4.0 1.0 7`	`0.585786437626906 -5.414213562373095`
`-1 0`	
`2 0`	
`-1 -3`	
`-1 -8`	
`6 -8`	
`6 6`	
`-1 6`	
`4.0 2.0 6`	`8.0 0.0`
`-1 0`	
`10 0`	
`10 3`	
`7 3`	
`7 5`	
`-1 5`	
`5.0 2.0 6`	`6.0 0.0`
`-1 0`	
`2 0`	
`2 -4`	
`6 -4`	
`6 6`	
`-1 6`	
`6.0 1.0 8`	`9.52786404500042 4.0`
`-1 0`	
`8 0`	
`7 2`	
`9 2`	
`8 4`	
`11 4`	
`11 12`	
`-1 12`	

Note

Note that above sample input corresponds to the cases in second figure. For convenience, in Figure 3, we will show a corresponding photographic playback of an animation for the case (C).



Problem C. Cat and String

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

A cat maintains a string. It performs a sequence of operations and queries. There is one kind of operation and one kind of query.

- The operation allows to insert a character at the end of the string.
- The query considers the substring, denoted by T, of the string from the l-th character to the r-th one, and asks the maximum length of substring of T which appears at least twice in T. If no substring appears at least twice in T, the maximum length is 0.

Input

The first line contains a non-empty string of lowercase English letters, followed by a single space and and a positive integer m. We use len to denote the length of this string. Each of the following m lines denotes either an operation or a query.

We define a temporary variable tmp, and it is initially set to 0.

We use the format "`1` c" to describe an operation: the new character is $(c - \text{'a'} + tmp) \bmod 26 + \text{'a'}$.

We use the format "`2` l r" to describe a query: the indices of substring T are $(l-1+tmp) \bmod len+1$ and $(r-1+tmp) \bmod len+1$. We guarantee that $1 \leq (l-1+tmp) \bmod len+1 \leq (r-1+tmp) \bmod len+1 \leq len$. After such query, tmp becomes equal to the query answer.

The initial length of the string and m are no more than $5 \cdot 10^4$.

Output

For each query, print the answer on a separate line.

Example

standard input	standard output
aabda 6 2 1 4 1 a 2 1 5 1 b 2 6 5 2 7 4	1 2 3 2

Problem D. Decimal Suffix

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

You are given two integers A and B. Find all bases X such that the decimal representation of B is a suffix of the representation of A in positional notation with base X.

Input

The first line of input contains two integers A and B ($1 \leq A \leq 10^9$, $10 \leq B < 100$), given in decimal notation.

Output

Print all X with the property mentioned in the problem statement, in arbitrary order. If no such X exist, print -1 instead.

Example

standard input	standard output
`71 13`	`68 4`

Note

Note that $71_{10} = 13_{68}$ and $71_{10} = 1013_4$. The string 13 is a suffix of both representations.

Problem E. Exotic Matter Reserves

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	768 mebibytes

As an eligible Ingress Resistance Agent, you should know your power source, the Exotic Matter. We call it XM, which is the driving force behind all of our actions in Ingress. XM allows us to construct items through hacking portals, to attack enemy portals, make links and create fields.

We try to collect XM from the ground. XM concentration comes from location based services, meaning that areas with a lot of foot traffic have higher amounts comparead to places that don't. You can collect XM by moving through those areas. XM will be automatically harvested by your Scanner when it is within your interaction circle.

Alice wants to select a location such that she can collect XM as much as possible. To simplify the problem, we consider the city as a grid map with size $n \times m$ numbered from $(0, 0)$ to $(n-1, m-1)$. The XM concentration inside the block (i, j) is $p(i, j)$. The radius of your interaction circle is r. We can assume that XM of the block (i, j) are located in the centre of this block. The distance between two blocks is the Euclidean distance between their centres.

Alice stands in the centre of one block and collects the XM. For each block with the distance d smaller than r to Alice, and whose XM concertation is $p(i, j)$, Alice's scanner can collect $p(i, j)/(1 + d)$ XM from it.

Help Alice to determine the maximum XM which she can collect once he stands in the centre of one block.

Input

There are no more than 100 test cases.

For each case, the first line consists of two integers n and m ($1 \le n, m \le 500$) and one real number r ($0 \le r \le 300$) given with no more than 9 digits after the decimal point. Each of the following n lines contains m integers $xm_{i,j}$ ($0 \le xm_{i,j} \le 100$) corresponding to the XM concentrations inside each block.

Output

For each case, output a separate line containing the maximum XM which Alice can collect with absolute error 10^{-3} or better.

Example

standard input	standard output

3 3 1
1 3 6
7 9 4
2 8 1
3 3 2
1 3 6
7 9 4
2 8 1
5 5 1.5
4 3 2 9 1
3 4 3 2 8
9 4 3 2 1
2 3 0 1 2
6 3 4 3 1
	9.000
	24.142
	17.956

Problem F. Funny Pub

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 mebibytes

You've just entered a funny pub for a drinking party with your dear friends.

Now you are to make orders for glasses of hard and soft drink as requested by the participants. But unfortunately, most staffs in this funny pub are part-time workers; they are not used to their work so they make mistakes at a certain probability for each order.

You are worrying about such mistakes. Today is a happy day for the participants, the dearest friends of yours.

Your task is to write a program calculating the probability at which the staff brings correct drinks for the entire orders. Cases in which the staff's mistakes result in a correct delivery should be counted into the probability, since those cases are acceptable for you.

Input

The input consists of no more than 270 test cases. Each test case begins with a line containing an interger N $(1 \leq N \leq 8)$. The integer N indicates the number of kinds of drinks available in the pub.

The following N lines specify the probabilities for the drinks in the format shown below.

$p_{11}\ p_{12} \cdots p_{1N}$

$p_{21}\ p_{22} \cdots p_{2N}$

...

$p_{N1}\ p_{N2} \cdots p_{NN}$

Each real number p_{ij} is given with no more than 3 digits after the decimal point and indicates the probability where the staff delivers a glass of the drink j for an order of the drink i. It holds that $p_{ij} \geq 0$ and $p_{i1} + p_{i2} + \ldots + p_{iN} = 1$ for $1 \leq i, j \leq N$. At the end of each test case, a line which contains N integers comes. The i-th integer n_i represents the number of orders for the drink i by the participants $(0 \leq n_i \leq 4)$.

The last test case is followed by a line containing a zero.

Output

For each test case, print a line containing the test case number (beginning with 1) followed by the natural logarithm of the probability where the staff brings correct drinks for the entire orders. Print the results with eight digits to the right of the decimal point. If the staff cannot bring correct drinks in any case, print "`-INFINITY`" instead. Use the format of the sample output.

Example

standard input

```
3
0.7 0.1 0.2
0.1 0.8 0.1
0.0 0.0 1.0
4 3 2
8
0.125 0.125 0.125 0.125 0.125 0.125
0.125 0.125
0.125 0.125 0.125 0.125 0.125 0.125
0.125 0.125
0.125 0.125 0.125 0.125 0.125 0.125
0.125 0.125
0.125 0.125 0.125 0.125 0.125 0.125
0.125 0.125
0.125 0.125 0.125 0.125 0.125 0.125
0.125 0.125
0.125 0.125 0.125 0.125 0.125 0.125
0.125 0.125
0.125 0.125 0.125 0.125 0.125 0.125
0.125 0.125
0.125 0.125 0.125 0.125 0.125 0.125
0.125 0.125
2 2 2 2 2 2 2 2
2
1 0
1 0
2 2
0
```

standard output

```
Case 1: -1.89723999
Case 2: -8.14438201
Case 3: -INFINITY
```

Problem G. Game Efficiency

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

You are deeply disappointed with the real world, so you have decided to live the rest of your life in the world of MMORPG (Massively Multi-Player Online Role Playing Game). You are no more concerned about the time you spend in the game: all you need is efficiency.

One day, you have to move from one town to another. In this game, some pairs of towns are connected by roads where players can travel. Various monsters will raid players on roads. However, since you are a high-level player, they are nothing but the source of experience points. Every road is bi-directional. A path is represented by a sequence of towns, where consecutive towns are connected by roads.

You are now planning to move to the destination town through the most efficient path. Here, the efficiency of a path is measured by the total experience points you will earn in the path divided by the time needed to travel the path.

Since your object is moving, not training, you choose only a straightforward path. A path is said straightforward if, for any two consecutive towns in the path, the latter is closer to the destination than the former. The distance of two towns is measured by the shortest time needed to move from one town to another.

Write a program to find a path that gives the highest efficiency.

Input

The first line contains a single integer T that indicates the number of cases ($1 \le T \le 111$).

The first line of each test case contains two integers n and m that represent the numbers of towns and roads respectively. The next line contains two integers s and t that denote the starting and destination towns respectively. Then m lines follow. The i-th line contains four integers u_i, v_i, e_i, and t_i, where u_i and v_i are two towns connected by the i-th road, e_i is the experience points to be earned, and t_i is the time needed to pass the road.

Each town is indicated by the town index number from 0 to $(n-1)$. The starting and destination towns never coincide. n, m, e_i's and t_i's are positive and not greater than 1000.

Output

For each case output in a single line, the highest possible efficiency. Print the answer with four decimal digits. The answer may not contain an absoleute error greater than 10^{-4}.

Example

standard input	standard output

2
3 3
0 2
0 2 240 80
0 1 130 60
1 2 260 60
3 3
0 2
0 2 180 60
0 1 130 60
1 2 260 60
	3.2500
	3.0000

Problem H. Highways Removal

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

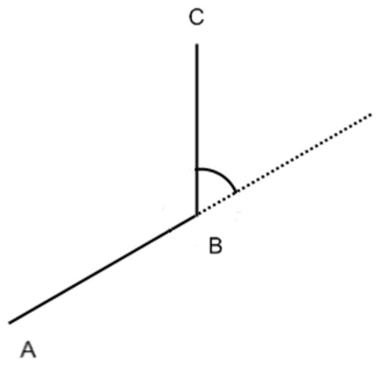
On one remarkable Cartesian plane there was an extremely conservative country. There were N cities in the country numbered from 1 to N. Compared to the infinite plane, the cities are so small that we shall consider them as points on the plane. The coordinates of the i-th city are (X_i, Y_i). No two cities reside in the same point. No three cities lie on a straight line.

Some pairs of cities are connected with bidirectional highways. Each highway is a line segment connecting two cities. There are exactly three outgoing highways for each city. No highway connects a city to itself. No pair of cities is connected with more than one highway.

That's how it went in this country from the very old times, and nobody ever thought of changing anything. But the trouble crept quietly — a liberal king got to the throne. And immediately issued a decree concerning the highways system of the country. The decree stated that some of the highways will be removed in order to meet the following conditions:

- Each city must have 2 outgoing highways.
- The turning angle between two highways outgoing from the same city must be strictly less than 60 degrees.
- No two highways intersect anywhere except the cities.

The turning angle between two highways is calculated in the following way. Say there are highways from city B to cities A and C. The turning angle between them is equal to the exterior angle at point B in triangle ABC (see the picture).



Implementation of the reform was delegated to the minister of transportation. He has to decide which of the highways will be removed and which will be kept. To please the king, the minister wants to keep such set of highways (of all possible sets satisfying the decree) which minimizes the maximum turning angle between highways outgoing from the same city.

Input

The first line of input contains an even integer N ($4 \le N \le 200$). Each of the followng N lines contains 5 integers. The first two integers in the i-th of these lines are X_i and Y_i ($-10^5 \le X_i, Y_i \le 10^5$). The following three numbers are the numbers of cities with which city i is connected initially. No two cities lie in the same point. No three cities lie on a straight line. Each city is connected with exactly three other cities. No pair of cities is connected with more than one highway. No highway connects a city to itself. Any two turning angles (not necessarily at one city) in the country before the reform differ by at least

10^{-5} degrees. Any turning angle in the country before the reform differs from 60 degrees by at least 10^{-5} degrees.

Output

If there is no way to fulfill the decree conditions, print a single line with text "`Minister's life is short :(`" (without outer quotes). Characters "`'`", "`:`" and "`(`" have ASCII-codes 39, 58 и 40, correspondingly.

Otherwise output the solution of the problem which the minister should choose. Print N integers. If the i-th of them is equal to j, it means that the highway between cities i and j should be removed.

Example

standard input	standard output

4
0 0 2 3 4
41 0 1 3 4
0 42 4 2 1
43 44 2 3 1
	Minister's life is short :(

8
0 100 2 6 8
0 201 3 1 5
102 303 2 4 8
204 305 3 5 7
306 207 2 4 6
308 109 1 5 7
210 0 6 8 4
111 0 1 3 7
	6 5 8 7 2 1 4 3

Problem I. ICPC

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

The gloomy cult of ICPC is existing for quite a long time. The witnesses confess that the servants of the cult practice extremely strange things. They divide into groups of three, worship the mysteriously glowing display, frantically knock on some keys grouped on a weird panel. The communication seems to be bidirectional — sometimes the "deity" returns signs of favor, which cause the joyful shouting of the servants. Although actually the majority of slothful servants rarely achieves this. More often the "deity" is unsatisfied, which causes confusion and even despondency in the ranks of the servants.

The nature of the "deity" is widely rumoured. Some say that the "deity" exists in several guises and each of them is controlled by an initiated person (maybe he's not even human, but a creature of a higher order) — an Admin. One highly acclaimed Admin is the so-called Admin with Big Beard. But let's get back to the problem.

Once, N servants of the ICPC cult gathered and complained that the traditional means of worship are outworn and pestering, and they aren't grim enough. So it was decided to replace the frantic knocking on the keyboard with dark energy interchange. A plan of the interchange was developed — the set of M pairs (A, B), which means that servant A should transfer the dark energy to servant B (and correspondingly servant B should receive the dark energy from servant A).

The servants began acting according to the plan, but the energy wouldn't transfer. As you surely understand, it can't possibly be that the ICPC servants possess no dark energy, so there had to be something else. In a while, the servants realized that they had to arrange in a special way. They drew a magic circle and stood at its border.

Of course, the existence of the magic circle is not enough. The process of energy transfer has to be associated with the circle. As all of us know, the process of energy transfer is associated with the circle if each servant transfers energy to the servants immediately succeeding him in the circle (in clockwise order) and receives energy from the servants immediately preceding him.

Let us define it more formally. Denote with $next(X)$ the servant immediately succeeding servant X in the circle (in clockwise order) and with $prev(X)$ the servant immediately preceding X. Also, let us define for $i > 1$ the values $next^i(X) = next(next^{i-1}(X))$ and $prev^i(X) = prev(prev^{i-1}(X))$. Put $next^1(X) = next(X)$ and $prev^1(X) = prev(X)$. The process of energy transfer is associated with the magic circle if and only if for each servant the following conditions are met:

- He transfers energy to servants $next^i(X)$, $1 \le i \le A$, where A is the total number of servants who receive energy from X;
- He receives energy from servants $prev^i(X)$, $1 \le i \le B$, where B is the total number of servants who transfer energy to X.

The goal has never been so close, but arranging on the circle in the proper way turned out to be a difficult task. Help the ICPC cult brethren!

Input

The input contains several scenarios. The first line of input contains their number T, afterwards their descriptions follow.

The description of each scenario begins with a line containing integers N and M ($N \ge 3$), which is followed by M lines. Each of them contains two integers A and B ($1 \le A, B \le N,\ A \ne B$), denoting that servant with number A should transfer the energy to servant with number B. Each pair (A, B) will be at most once in a single scenario. Both (A, B) and (B, A) will not be simultaneously in a single scenario.

The servants are numbered from 1 to N. Sum of N's over all scenarios in a single input does not exceed 10^5. Sum of M's over all scenarios in a single input does not exceed $2 \cdot 10^5$.

Output

Print a single line for each scenario in the input. If the solution for the scenario does not exist, the line should contain text "`Epic fail`" (without quotes). Otherwise, the line should contain a permutation of numbers from 1 to N — the order in which the servants should stand to achieve the process of energy transfer associated with the magic circle. The order should be clockwise. If there are several solutions, output any.

Example

standard input	standard output

3
6 10
1 6
2 3
2 4
2 5
3 1
3 6
4 3
4 5
5 3
6 2
3 0
4 3
1 2
2 3
3 1
	1 6 2 4 5 3
	1 2 3
	Epic fail