Problem A
Asteroid Rangers
Problem ID: asteroids

The year is 2112 and humankind has conquered the solar system. The Space Ranger Corps have set up bases on any hunk of rock that is even remotely inhabitable. Your job as a member of the Asteroid Communications Ministry is to make sure that all of the Space Ranger asteroid bases can communicate with one another as cheaply as possible. You could set up direct communication links from each base to every other base, but that would be prohibitively expensive. Instead, you want to set up the minimum number of links so that everyone can send messages to everyone else, potentially relayed by one or more bases. The cost of any link is directly proportional to the distance between the two bases it connects, so this doesn't seem that hard of a problem.

There is one small difficulty, however. Asteroids have a tendency to move about, so two bases that are currently very close may not be so in the future. Therefore as time goes on, you must be willing to switch your communication links so that you always have the cheapest relay system in place. Switching these links takes time and money, so you are interested in knowing how many times you will have to perform such a switch.

A few assumptions make your task easier. Each asteroid is considered a single point. Asteroids always move linearly with a fixed velocity. No asteroids ever collide with other asteroids. Also, any relay system that becomes optimal at a time $t \geq 0$ will be uniquely optimal for any time s satisfying $t < s < t+10^{-6}$. The initial optimal relay system will be unique.

Input

Each test case starts with a line containing an integer n ($2 \leq n \leq 50$) indicating the number of asteroid bases. Following this are n lines, each containing six integers x, y, z, v_x, v_y, v_z. The first three specify the initial location of an asteroid ($-150 \leq x, y, z \leq 150$), and the last three specify the x, y, and z components of that asteroid's velocity in space units per time unit ($-100 \leq v_x, v_y, v_z \leq 100$).

Output

For each test case, display a single line containing the case number and the number of times that the relay system needs to be set up or modified.

Sample Input	Output for Sample Input
3 0 0 0 0 0 0 5 0 0 0 0 0 10 1 0 -1 0 0 4 0 0 0 1 0 0 0 1 0 0 -1 0 1 1 1 3 1 1 -1 -1 2 1 -1 -1	Case 1: 3 Case 2: 3

This page is intentionally left blank.

Problem B
Curvy Little Bottles
Problem ID: bottle

In her bike rides around Warsaw, Jill happened upon a shop that sold interesting glass bottles. She thought it might make an interesting project to use such bottles for measuring liquids, but this would require placing markings on the bottles to indicate various volumes. Where should those volume marks be placed?

Jill formalized the problem as follows. Assume a bottle is formed by revolving a shape that is the same as the graph of a polynomial P between $x = x_{\text{low}}$ and $x = x_{\text{high}}$ around the x-axis. Thus the x-axis is coincident with a vertical line through the center of the bottle. The bottom of the bottle is formed by a solid circular region at $x = x_{\text{low}}$, and the top of the bottle, at $x = x_{\text{high}}$, is left open.

The first sample input represents a bottle formed using the simple polynomial $4 - 0.25x$, with $x_{\text{low}} = 0$ and $x_{\text{high}} = 12$. The bottom of this bottle is a circle with a radius of 4, and the opening at the top is a circle with a radius of 1. The height of this bottle is 12. Volume markings are in increments of 25.

Given a polynomial P, x_{low}, x_{high}, and the volume increment between successive marks on the bottle, compute the distances up from x_{low} for the marks at successive volume increments. A mark cannot be made past the top of the bottle, and no more than the first 8 increments should be marked. Assume the value of P is greater than zero everywhere between x_{low} and x_{high}.

Input

Each test case consists of three lines of bottle data:

- Line 1: n, the degree of the polynomial (an integer satisfying $0 \leq n \leq 10$).
- Line 2: $a_0, a_1, \ldots, a_n$, the real coefficients of the polynomial P defining the bottle's shape, where a_0 is the constant term, a_1 is the coefficient of x^1, ..., and a_n is the coefficient of x^n. For each i, $-100 \leq a_i \leq 100$, and $a_n \neq 0$.
- Line 3:
  - x_{low} and x_{high}, the real valued boundaries of the bottle ($-100 \leq x_{\text{low}} < x_{\text{high}} \leq 100$ and $x_{\text{high}} - x_{\text{low}} > 0.1$).
  - inc, an integer which is the volume increment before each successive mark on the bottle ($1 \leq inc \leq 500$).

Output

For each test case, display the case number and the volume of the full bottle on one line. On a second line, display the increasing sequence of no more than 8 successive distances up from the bottom of the bottle for the volume markings. All volumes and height marks should be accurate to two decimal places. If the bottle does not have a volume that allows at least one mark, display the phrase `insufficient volume`. No test case will result in a mark within 0.01 from the top of the bottle. The volume of the bottle will not exceed 1 000. All rounded distances for marks on a bottle differ by at least 0.05.

Sample Input	Output for Sample Input
1 4.0 -0.25 0.0 12.0 25 1 4.0 -0.25 0.0 12.0 300 0 1.7841241161782 5.0 10.0 20 0 1.0 0.0 10.0 10	Case 1: 263.89 0.51 1.06 1.66 2.31 3.02 3.83 4.75 5.87 Case 2: 263.89 insufficient volume Case 3: 50.00 2.00 4.00 Case 4: 31.42 3.18 6.37 9.55

Problem C
Bus Tour
Problem ID: bustour

Imagine you are a tourist in Warsaw and have booked a bus tour to see some amazing attraction just outside of town. The bus first drives around town for a while (a *long* while, since Warsaw is a big city) picking up people at their respective hotels. It then proceeds to the amazing attraction, and after a few hours goes back into the city, again driving to each hotel, this time to drop people off.

For some reason, whenever you do this, your hotel is always the first to be visited for pickup, and the last to be visited for dropoff, meaning that you have to suffer through two not-so-amazing sightseeing tours of all the local hotels. This is clearly not what you want to do (unless for some reason you are *really* into hotels), so let's fix it. We will develop some software to enable the sightseeing company to route its bus tours more fairly—though it may sometimes mean longer total distance for everyone, but fair is fair, right?

For this problem, there is a starting location (the sightseeing company headquarters), h hotels that need to be visited for pickups and dropoffs, and a destination location (the amazing attraction). We need to find a route that goes from the headquarters, through all the hotels, to the attraction, then back through all the hotels again (possibly in a different order), and finally back to the headquarters. In order to guarantee that none of the tourists (and, in particular, *you*) are forced to suffer through two full tours of the hotels, we require that every hotel that is visited among the first $\lfloor h/2 \rfloor$ hotels on the way to the attraction is also visited among the first $\lfloor h/2 \rfloor$ hotels on the way back. Subject to these restrictions, we would like to make the complete bus tour as short as possible. Note that these restrictions may force the bus to drive past a hotel without stopping there (this is not considered visiting) and then visit it later, as illustrated in the first sample input.

Input

The first line of each test case consists of two integers n and m satisfying $3 \leq n \leq 20$ and $2 \leq m$, where n is the number of locations (hotels, headquarters, attraction) and m is the number of pairs of locations between which the bus can travel.

The n different locations are numbered from 0 to $n-1$, where 0 is the headquarters, 1 through $n-2$ are the hotels, and $n-1$ is the attraction. Assume that there is at most one direct connection between any pair of locations and it is possible to travel from any location to any other location (but not necessarily directly).

Following the first line are m lines, each containing three integers u, v, and t such that $0 \leq u, v \leq n-1$, $u \neq v$, $1 \leq t \leq 3600$, indicating that the bus can go directly between locations u and v in t seconds (in either direction).

Output

For each test case, display the case number and the time in seconds of the shortest possible tour.

Sample Input	Output for Sample Input
5 4 0 1 10 1 2 20 2 3 30 3 4 40 4 6 0 1 1 0 2 1 0 3 1 1 2 1 1 3 1 2 3 1	Case 1: 300 Case 2: 6

Problem D
Fibonacci Words
Problem ID: fibonacci

The Fibonacci word sequence of bit strings is defined as:

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n-2) & \text{if } n \geq 2 \end{cases}$$

Here $+$ denotes concatenation of strings. The first few elements are:

n	$F(n)$
0	0
1	1
2	10
3	101
4	10110
5	10110101
6	1011010110110
7	10110101101101010110101
8	1011010110110101101011011010110110
9	10110101101101011010110110101101101011010110110101101

Given a bit pattern p and a number n, how often does p occur in $F(n)$?

Input

The first line of each test case contains the integer n ($0 \leq n \leq 100$). The second line contains the bit pattern p. The pattern p is nonempty and has a length of at most 100 000 characters.

Output

For each test case, display its case number followed by the number of occurrences of the bit pattern p in $F(n)$. Occurrences may overlap. The number of occurrences will be less than 2^{63}.

Sample Input	Output for Sample Input
6 10 7 10 6 01 6 101 96 10110101101101	Case 1: 5 Case 2: 8 Case 3: 4 Case 4: 4 Case 5: 7540113804746346428

This page is intentionally left blank.

Problem E
Infiltration
Problem ID: infiltration

Good morning, agent W-12. Your mission, should you choose to accept it, is as follows.

We are infiltrating the ever so insidious Association of Chaos and Mischief (ACM) in order to take down their command structure. Unfortunately, they appear to be prepared for such an eventuality, and have given their command structure an annoyingly complex design which makes our infiltration quite difficult.

The ACM command structure is divided into several cells. For each pair of cells A and B, either A controls B or B controls A. But this "control" relation can be cyclic, so it could happen that A controls B and B controls C and C controls A.

We can send in agents to infiltrate any particular cell, which gives us control over that cell and the cells that it controls, but not any other cells. So in the example above, infiltrating A would give us control over A and B, but not C.

For a successful infiltration of the ACM, we must obtain control over all of its cells, otherwise the cells that are out of our control will discover us and start causing some of their trademark chaos and mischief. As you know, we're on a tight spending leash from higher authority these days, so we need to execute this mission as efficiently as possible. Your mission is to figure out the minimum number of cells we need to infiltrate in order to succeed.

This mission briefing will self-destruct in five hours. Good luck!

Input

The first line of a test case contains the number n of cells the ACM has ($1 \leq n \leq 75$). Each of the next n lines contains a binary string of length n where the i^{th} character of the j^{th} line is 1 if cell j controls cell i, and 0 otherwise ($1 \leq i, j \leq n$).

The i^{th} character of the i^{th} line is 0 and for $i \neq j$, either the i^{th} character of the j^{th} line is 1 or the j^{th} character of the i^{th} line is 1, but not both.

Output

For each test case, display its case number followed by the minimum number m of cells that must be infiltrated to obtain complete control of the ACM. Then display m numbers $c_1, \ldots, c_m$ in any order, indicating the list of cells to infiltrate (cells are numbered from 1 to n). If more than one set of m cells gives complete control, any one will be accepted.

Sample Input	Output for Sample Input
2 00 10 3 010 001 100 5 01000 00011 11001 10100 10010	Case 1: 1 2 Case 2: 2 1 2 Case 3: 2 2 3

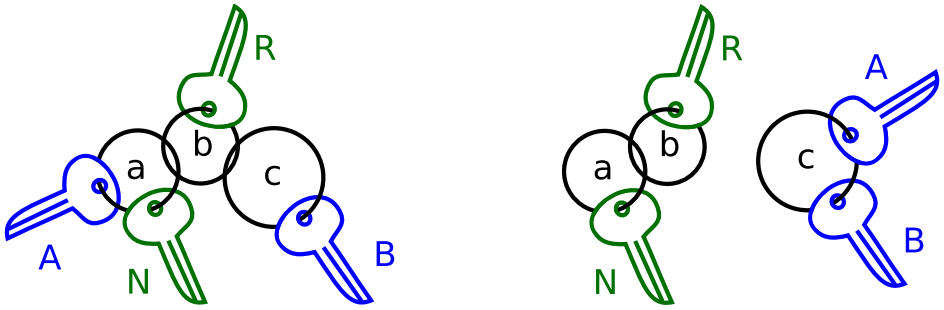
Problem F
Keys
Problem ID: keys

Adam carries a bunch of keys attached to key rings, some of which may be connected to each other. The rings are common key rings, so a key can be attached to or detached from a ring by sliding along the spiral. In the same way, two rings can be connected or disconnected. Adam wants to give some of the keys to Brenda. Since manipulating the keys and rings is often an annoying task (and also dangerous to one's fingernails), Adam is looking for a way to minimize the number of key and ring operations.

Every key attachment, key detachment, ring connection, or ring disconnection is considered one operation. Since manipulating two rings is significantly easier than sliding a key, we first want to minimize the number of keys being detached and attached. Among solutions with the same minimal number of key operations, you need to find the one with the minimal number of ring connections and disconnections.

When all the operations are complete, Adam and Brenda must each carry one connected group of rings and keys. The only exception is when either of them would have no keys at all—in such a case, no ring is needed. Each key must be attached to exactly one ring. Some rings (but not keys) may be considered leftovers and may remain disconnected from the two groups.

The left side of the following figure shows an initial configuration consisting of four keys on three rings. Adam wishes to give Brenda the two keys labeled N and R. This can be accomplished by two key operations and one ring operation, resulting in the configuration shown on the right side of the figure.



Input

Each test case contains one or more lines, each containing a two letter string. Lowercase letters (a - z) represent key rings and uppercase letters (A - Z) represent keys. The two letters on a line specify either a key attached to a ring or two rings connected together. The end of each test case is denoted by a line containing the digit zero.

Keys denoted by letters A through M remain with Adam, and keys denoted by letters N through Z are given to Brenda.

No line contains two uppercase letters. No pair of letters are specified more than once in the same test case. Each key is connected to exactly one ring. There are no "circles" in the ring configurations (disconnecting any two rings will increase the number of connected groups). All existing keys and rings are mentioned at least once.

Output

For each test case, display the case number followed by the minimal number of key attach/detach operations and the minimal number of ring connect/disconnect operations.

If there is no way to split the keys as requested, display the case number and the word `impossible` instead of the two integers.

Sample Input	Output for Sample Input
ab bc aA aN Rb cB 0 aA bB Cc 0 aA aZ 0 aA bB cC xX yY ax xb by yc 0	Case 1: 2 1 Case 2: 0 2 Case 3: impossible Case 4: 0 7

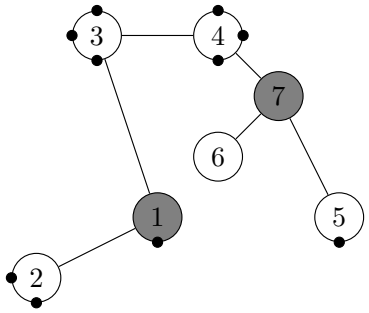
Problem G
Minimum Cost Flow
Problem ID: minflow

You have been hired to construct a system to transport water between two points in an old factory building using some existing components of the old plumbing. The old components consist of *pipes* and *junctions*. Junctions are points where pipes may have previously been joined. We say *previously* joined, because some of the old pipes were damaged and have been removed, effectively leaving open holes in the junctions to which they were connected. If water should enter one of these junctions, it would pour out of an open hole and eventually flood the building—clearly an undesirable event.

You can remedy this situation by installing new pipes between some of the open holes and installing plugs to close other open holes as necessary. When you install a new pipe connecting two holes (which must be in two different junctions), the two holes are no longer open and water will be able to flow through the new pipe. The cost of installing a new pipe is equal to the distance between the centers of the two junctions the pipe connects. The cost of installing a plug in an open hole is 0.5. You are not concerned about open holes in junctions that will never be reached by water.

Two of the junctions are special. One, called the *source*, is the point where water will be pumped into the new system. The other, called the *destination*, is where the water is needed. After any plugs and new pipes have been added to the system, water will be pumped into it at the source with a pressure sufficient to reach a specified height (in the absence of leaks, of course). You are allowed to select the pressure arbitrarily, and are guaranteed that the pressure will not change during the operation of the system. Naturally the pressure must be sufficient to force water up to the heights of both the source and the destination. Your task is simply to find the most inexpensive way of getting water from the source junction to the destination junction without flooding the building.

The figure below corresponds to the first sample input case, where black dots represent open holes, junction 1 is the source, and junction 7 is the destination. (The position of a black dot on its circle has no significance and is used for illustration purposes only.)



Water flows through the system according to the laws of physics. If the pressure is sufficient to fill a junction with water, then that junction will remain filled with water. If there are pipes extending horizontally or downward from a junction, then water will also flow through those pipes. Water will also flow upward through pipes connected to a junction up to the height determined by the water pressure. Of course, if the water reaches an open hole in a junction, it will flow through the hole and flood the building.

In the first sample input case, you can connect junctions 1 and 5 at a cost of 3, plug the open holes in junction 2, and set the pressure so that the water flows up to junction 7 only. The water will fill junctions 1, 2, 5, 6 and 7, and will flow no higher. A different (more expensive) solution would be to simply plug

all the holes at a total cost of 5, and let the water flow through all the junctions. You cannot solve this case by connecting junctions 1 and 6 and plugging holes in junctions 2 and 5, since junction 6 has no open holes to which a new pipe can be connected.

Assume existing pipes and any new pipes do not interfere with each other or with any junctions, except those to which they are connected. That is, even if a straight line from junction A to junction B passes through junction C, any pipe from A to B will not touch C.

Input

The first line of each test case contains two integers N and M, where N ($2 \leq N \leq 400$) is the number of junctions in the building (numbered 1 through N) and M ($0 \leq M \leq 50\,000$) is the number of existing usable pipes. Each of the next N lines contains four integers x_i, y_i, z_i, and k_i satisfying $-10\,000 \leq x_i, y_i, z_i \leq 10\,000$ and $0 \leq k_i \leq 400$, $i = 1, 2, ..., N$. The i^{th} line describes junction i: (x_i, y_i, z_i) is the location of the i^{th} junction where the z-axis is the vertical axis; k_i indicates the number of open holes in the junction. Each of the next M lines contains two integers a_j and b_j satisfying $1 \leq a_j < b_j \leq N$. The j^{th} line indicates that pipe j connects junctions a_j and b_j. At most one pipe connects any pair of junctions, and no two junctions share the same coordinates. The source is junction 1, and the destination is junction N.

Output

For each case, display the case number. Then if suitable new pipes and plugs can be used to construct the desired system, display the minimum cost of connecting the source junction to the destination junction, accurate to four decimal places. If it is impossible to connect the source to the destination, display the word `impossible`.

Sample Input	Output for Sample Input
7 6 2 0 1 1 0 0 0 2 1 0 4 3 3 0 4 3 5 0 1 1 3 0 2 0 5 0 3 0 1 2 1 3 3 4 4 7 5 7 6 7 4 1 2 0 0 0 3 0 1 0 4 1 0 1 5 1 1 1 1 2	Case 1: 4.0000 Case 2: impossible

Problem H
Room Service
Problem ID: room

You are working for a company designing cute, funny robot vacuum cleaners. At a high level, the robots' behavior is divided into three modes:

1. Exploration
2. Vacuuming
3. Rampant Killing

Unfortunately, while consumer testing shows that the last two modes are working perfectly, the exploration mode still has bugs. You've been put in charge of debugging.

At the beginning of the exploration mode, the robot is placed into a convex polygonal room. It has sensors that should tell it where all the walls are. Your job is to write a program that verifies that these readings are correct. To do this, the robot needs to physically touch every wall in the room.

Your problem is this: given the shape of a convex polygonal room with N walls and a starting point P inside it, determine the shortest route that touches each wall and then returns to P. Touching a corner counts as touching both incident walls.

Input

Each test case starts with a line containing the number of vertices N of the polygon ($3 \leq N \leq 100$) and the integer coordinates P_x and P_y of the robot's starting point ($-10\,000 \leq P_x, P_y \leq 10\,000$). This is followed by N lines, each containing two integers x, y ($-10\,000 \leq x, y \leq 10\,000$) defining a vertex of the polygon. Vertices are given in counterclockwise order, all interior angles are less than 180 degrees, the polygon does not self-intersect, and the robot's starting point is strictly inside the polygon.

Output

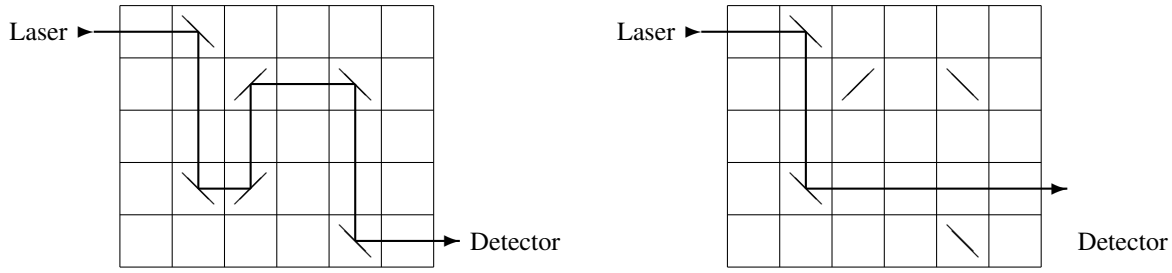
For each test case, display the case number and the length of the desired route, accurate to two decimal places.

Sample Input	Output for Sample Input
4 0 0 -1 -1 1 -1 1 1 -1 1 3 10 1 0 0 30 0 0 20	Case 1: 5.66 Case 2: 36.73

This page is intentionally left blank.

Problem I
A Safe Bet
Problem ID: safe

Safe Ltd. is a company that manufactures high-quality safes. Its latest invention is an optical closure mechanism that uses a laser beam passing through a rectangular grid with several mirrors.



When the laser is activated, a beam enters the top row of the grid horizontally from the left. The beam is reflected by every mirror that it hits. Each mirror has a 45 degree diagonal orientation, either ／ or ＼. If the beam exits the bottom row of the grid horizontally to the right, it is detected and the safe opens (see the left side of the figure above). Otherwise the safe remains closed and an alarm is raised.

Each safe has a missing mirror, which prevents the laser beam from traveling successfully through the grid (see the right side of the figure above). The safe has a mechanism that enables the user to drop a single mirror into any empty grid cell. A legitimate user knows the correct position and orientation of the missing mirror (／ in row 4 column 3 above) and can thus open the safe. Without this knowledge the user has to guess correctly, which can be difficult for safes with large grids.

Your job is to determine if particular safes are actually secure. A secure safe does not open right away without inserting a mirror, and there is at least one valid location and orientation for the missing mirror. There may indeed be multiple such locations and orientations.

Input

Each test case describes a single safe and starts with a line containing four integer numbers r, c, m, and n ($1 \leq r, c \leq 1\,000\,000$ and $0 \leq m, n \leq 200\,000$). The mechanism's grid has r rows and c columns. Each of the next m lines contains two integer numbers r_i and c_i ($1 \leq r_i \leq r$ and $1 \leq c_i \leq c$) specifying that there is a ／ mirror in row r_i column c_i. The following n lines specify the positions of the ＼ mirrors in the same way. The $m + n$ positions of the mirrors are pairwise distinct.

Output

For each test case, display its case number followed by:

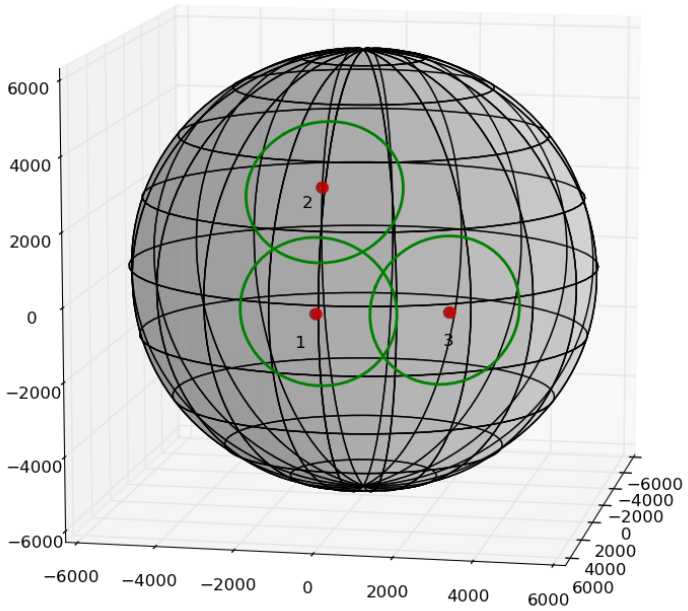
- `0` if the safe opens without inserting a mirror.
- k r c if the safe does not open without inserting a mirror, there are exactly k positions where inserting a mirror opens the safe, and (r, c) is the lexicographically smallest such row, column position. A position where both a $\diagup$ and a $\diagdown$ mirror open the safe counts just once.
- `impossible` if the safe cannot be opened with or without inserting a mirror.

Sample Input	Output for Sample Input
5 6 1 4 2 3 1 2 2 5 4 2 5 5 100 100 0 2 1 77 100 77 100 100 0 0	Case 1: 2 4 3 Case 2: 0 Case 3: impossible

Problem J
Shortest Flight Path
Problem ID: shortest

Commercial flights are statistically quite safe (in terms of number of deaths per passenger-kilometer, only going to the moon is safer). But there are still reasons for precautions and safety regulations. An early such rule was the so-called "60-minute rule," which required that a two-engine plane must always be within 60 minutes of the nearest adequate airport along its entire flight path. A variety of similar rules have existed, but at their core, they remain the same: the flight path can not take the airplane more than a certain maximum allowed distance from the nearest airport. With these restrictions, planes cannot always use a direct route for flying from one airport to another.

In this problem we will compute the shortest flight path between two airports while adhering to a maximum allowed distance rule. In the figure below, which illustrates the first sample test case, any flight route has to stay within the three circles. Thus a plane going from airport 2 to airport 3 has to detour from the direct route via the region around airport 1. Note that the plane would not necessarily have to go to airport 1 itself.



Things are further complicated by the fact that planes have limited fuel supply, and to go longer distances they may need to make a stopover at intermediate airports. Thus, depending on the fuel capacity, a plane going from airport 2 to airport 3 in the figure might have to stop over at airport 1 (or the fuel capacity might be too low even to go to airport 1, in which case the trip would be impossible to make).

We make the following simplifying assumptions:

1. The surface of the earth is a sphere of radius 6370 km.
2. Both time and fuel consumption are directly proportional to distance traveled. In other words we are interested only in total distance traveled.
3. The difference in distance caused by planes flying at different altitudes is negligible. Thus, effectively, we assume them to be flying along the earth's surface.
4. A plane may stop for refueling at as many intermediate airports as needed, each time getting a full tank.

Input

The first line of each test case contains two integers N and R, where $2 \leq N \leq 25$ is the number of airports and $1 \leq R \leq 10\,000$ is the maximum allowed flight distance (in km) from the nearest airport. Each of the next N lines contains two integers ϕ, θ satisfying $0 \leq \phi < 360$ and $-90 \leq \theta \leq 90$, the longitude and latitude (respectively) of an airport, in degrees. The airports are numbered according to their order in the input starting from one. No two airports are at the same position.

Following this is a line containing an integer Q, satisfying $1 \leq Q \leq 100$. Each of the next Q lines contains three integers s, t, c satisfying $1 \leq s, t \leq N$, $s \neq t$, and $1 \leq c \leq 50\,000$, indicating a plane going from airport s to airport t with a fuel capacity yielding a range of c km.

Output

For each test case, display the case number followed by one line for each query containing the length in km of the shortest flight path between airport s and t, subject to the fuel constraint c. Display the length accurate to three decimal places. If there is no permissible path between the two airports, then display the word `impossible` instead.

You may assume the answer is numerically stable for perturbations of up to 0.1 km of R or c.

Sample Input	Output for Sample Input
3 2000 0 0 0 30 30 0 3 2 3 5000 2 3 4000 2 3 3000 2 10000 45 45 225 -45 2 1 2 50000 2 1 50000	Case 1: 4724.686 6670.648 impossible Case 2: impossible impossible

Problem K
Stacking Plates
Problem ID: stacking

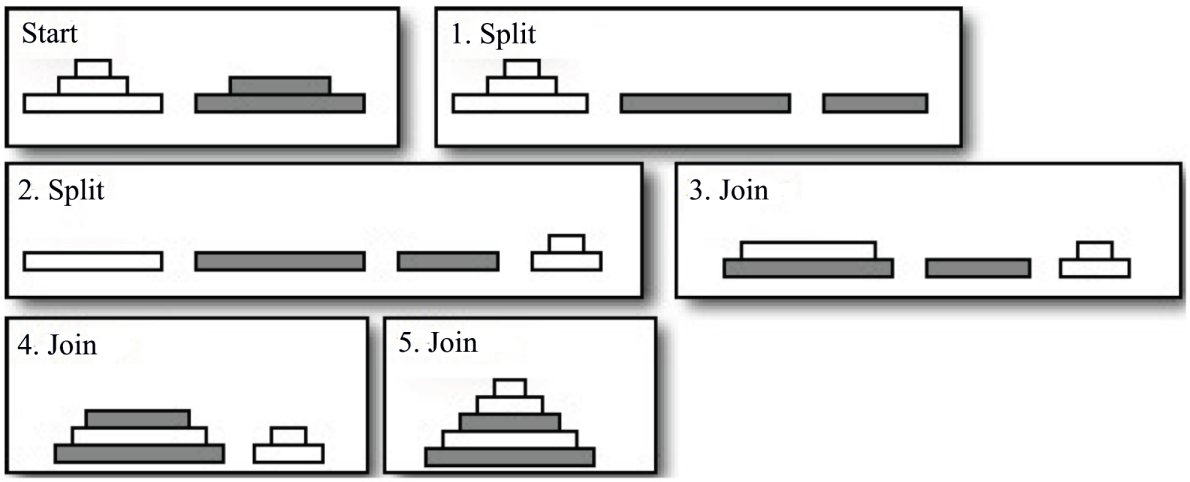
The Plate Shipping Company is an Internet retailer that, as their name suggests, exclusively sells plates. They pride themselves in offering the widest selection of dinner plates in the universe from a large number of manufacturers.

In a recent cost analysis the company has discovered that they spend a large amount of money on packing the plates for shipment. Part of the reason is that plates have to be stacked before being put into shipping containers. And apparently, this is taking more time than expected. Maybe you can help.

A shipment of plates consists of plates from several manufacturers. The plates from each manufacturer come stacked, that is, each arranged in a single stack with plates ordered by size (the smallest at the top, the largest at the bottom). We will call such a stack *properly ordered*. To ship all these plates, you must combine them into a single stack, again properly ordered. To join the manufacturers' stacks into a single stack, two kinds of operations are allowed:

- Split: a single stack can be split into two stacks by lifting any top portion of the stack and putting it aside to form a new stack.
- Join: two stacks can be joined by putting one on top of the other. This is allowed only if the bottom plate of the top stack is no larger than the top plate of the bottom stack, that is, the joined stack has to be properly ordered.

Note that a portion of any stack may never be put directly on top of another stack. It must first be split and then the split portion must be joined with the other stack. Given a collection of stacks, you have to find the minimum number of operations that transforms them into a single stack. The following example corresponds to the sample input, and shows how two stacks can be transformed to a single stack in five operations:



Input

Each test case starts with a line containing a single integer n ($1 \leq n \leq 50$), the number of stacks that have to be combined for a shipment. This is followed by n lines, each describing a stack. These lines start with an integer h ($1 \leq h \leq 50$), the height of the stack. This number is followed by h positive integers that give the diameters of the plates, from top to bottom. All diameters are at most 10 000. These numbers will be in non-decreasing order.

Output

For each test case, display the case number and the minimum number of operations (splits and joins) that have to be performed to combine the given stacks into a single stack.

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

Problem L
Takeover Wars
Problem ID: takeover

You are studying a takeover war between two large corporations, Takeover Incorporated and Buyout Limited. Each of these corporations controls a number of subsidiaries. The aim in this war is simply to drive the competition out of the market. There are N subsidiaries of Takeover Incorporated and M subsidiaries of Buyout Limited, and you know the market value of each subsidiary.

Each company can designate one of its subsidiaries to perform a takeover. The takeover can either be friendly or hostile. A friendly takeover means a subsidiary of a corporation merges with a different subsidiary of the same corporation. The market value of the merged subsidiary is the sum of the market values of the constituent subsidiaries. There is no constraint on the relative sizes of the subsidiaries participating in a friendly takeover.

A hostile takeover means a subsidiary A of a corporation attempts to take over a subsidiary B of the other corporation. For this to succeed, the market value of A has to be greater than the market value of B. After this move, B disappears from the market. The market value of A does not change (the gain of incorporating B's assets is offset by the monetary cost of the takeover). For simplicity we assume that no sequence of moves leads to two subsidiaries of different corporations having the same market value.

The companies take turns making moves in this takeover war, with Takeover Incorporated going first. A company will do nothing on its turn only if it cannot make a takeover. A company loses the takeover war if all its subsidiaries are taken over.

Your aim is to learn which company can guarantee a victory from this war. In the first case of the sample data, Takeover Incorporated can simply take over one of the companies of Buyout Limited in its first move with the 7-value subsidiary. Then it will lose one of its small (1-value) subsidiaries to a hostile takeover, and then it will take over the second subsidiary of Buyout Limited. In the second case, Takeover has to make a friendly takeover in its first move. Buyout Limited will join its two subsidiaries into a single company with market value 10. Takeover will have to make a friendly takeover again (as again it will not have a large enough subsidiary to take over Buyout's giant). Now Takeover will have two subsidiaries, valued either 9 and 3 or 6 and 6. In either case, Buyout takes over one of these subsidiaries, Takeover has to pass, and Buyout takes over the other one.

Input

Each test case is described by three lines of input. The first line contains two numbers $1 \leq N \leq 10^5$ and $1 \leq M \leq 10^5$ denoting respectively the number of subsidiaries of Takeover Incorporated and Buyout Limited. The next line lists the N sizes a_i of the subsidiaries of Takeover Incorporated ($1 \leq a_i \leq 10^{12}$), and the third line lists the M sizes b_j of the subsidiaries of Buyout Limited ($1 \leq b_j \leq 10^{12}$).

Output

For each test case, display the case number and either the phrase `Takeover Incorporated` or the phrase `Buyout Limited` depending on who wins the takeover war if both corporations act optimally.

Sample Input	Output for Sample Input
3 2 7 1 1 5 5 4 2 3 3 3 3 5 5	Case 1: Takeover Incorporated Case 2: Buyout Limited