A. Anti-Tetris

time limit per test: 2 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

Let us consider the game "Sticky Tetris". In this game, there is a field of $n \times m$ squares. Tiles appear on the field and the player can move the tiles.

Each tile is a 4-connected set of at most 7 squares.

Each new tile appears in any position that fits inside the field, does not intersect any other tile, and the top cell of the tile is at the top row of the field. The player can move the tile left, right, and down, and at any moment the tile must still entirely fit inside the field and must not intersect other tiles. The player can stop the tile at any position at any time. After that, it cannot be moved. Since this is "Sticky Tetris," the tile will not fall once stopped.

You are given a final configuration of a "Sticky Tetris" game. You need to restore a sequence of steps that leads to that configuration if it exists.

Input

The first line contains two integers n and m ($1 \le n, m \le 50$) — the size of the playing field.

The next n lines contain a string of m characters each. Each character could be either a '.', or lowercase English letter. Connected components of the same letter correspond to a single tile. Each tile consists of at most 7 squares.

Output

If there is no solution, print -1.

Otherwise, print k — the number of different tiles that are placed on the field.

On the next k lines print the sequence of steps for each of the tiles in the order they are placed.

Each line consists of a number x followed by a string with steps. x ($1 \le x \le m$) is the starting column of the leftmost square in the top row of the tile. The string consists of characters 'L' (for left), 'R' (for right), and 'D' (for down), describing the path of that tile, ending with a single character 'S' (for stop). The final position of the tile determines which tile is being placed. The string with steps can have at most $n \cdot m + 1$ characters.

Examples

input
3 2 aa ab aa
output
2 2 DS 1 S

input
5 6 ....dd .ccccd .cbbdd .aab.a aabbaa
output
5 2 DDDS 4 DDLS 6 DDDS 2 DS 5 S

input
5 3 ... aab abb aab .bb

output
-1

B. Building Forest Trails

time limit per test: 3 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

There are n villages lying equidistant on a circle in the middle of a thick, impassable forest. From ancient times, it was impossible to move from one village to another, but technical progress has changed a lot. Now, there is a technology to build passable trails in the forest.

The building process consists of m events. Each event is either building a trail or querying if two villages are connected. Trails are built as straight lines connecting two villages. After a trail is built, anybody can walk along the trail from one village to another.

Moreover, if two trails cross, anybody can turn at the intersection, and if other trails leave a village you have just reached, they can also be used to walk along. So, for example, if villages are numbered 1 to 6 in the order around the circle, and there are trails 1 to 3, 2 to 4, and 4 to 6, then all villages, except the 5-th, are reachable from the 1-st village.

Given a list of m events, for each query, find if two given villages are reachable from each other at that moment.

Input

The first line contains two integers n ($2 \le n \le 2 \cdot 10^5$) and m ($1 \le m \le 3 \cdot 10^5$) — the number of villages and the number of events respectively.

Next m lines contain events. Each event description consists of three integers e (e is 1 or 2), v ($1 \le v \le n$), and u ($1 \le u \le n$, $u \ne v$). If $e = 1$, then the event is building a trail between villages v and u. If $e = 2$, then the event is a query if the villages v and u are connected. It is guaranteed that each trail is built at most once.

Villages are numbered 1 to n in clockwise order around the circle.

Output

For each query print one character '0' if villages are not reachable, and '1' if villages are reachable from each other. Print answers for all queries as a single string in one line.

Examples

input
6 9 1 1 3 1 4 6 2 3 4 1 2 4 2 1 2 2 1 3 2 1 4 2 6 1 2 5 3
output
011110

input
2 5 2 1 2 2 2 1 1 1 2 2 1 2 2 2 1
output
0011

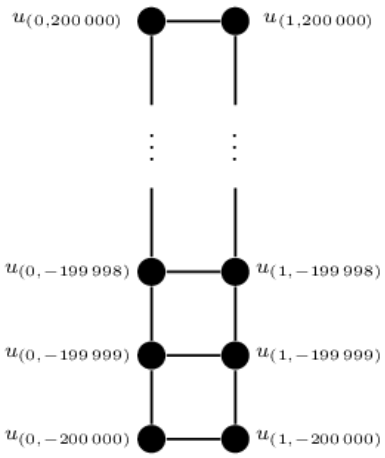
C. Cactus Lady and her Cing

time limit per test: 5 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

Cactus lady loves her cactuses very much. Especially she likes a small cactus named Cing. Cing can be seen as a connected undirected graph in which every vertex lies on at most one simple cycle. Intuitively, a cactus is a generalization of a tree where some cycles are allowed. Multiedges (multiple edges between a pair of vertices) and loops (edges that connect a vertex to itself) are not allowed.

She bought a special grid for her special little cactus Cing. This grid can be represented as a graph consisting of two paths of length $400\,000$, $u_{(0,-200\,000)} - u_{(0,-199\,999)} - \ldots - u_{(0,200\,000)}$ and $u_{(1,-200\,000)} - u_{(1,-199\,999)} - \ldots - u_{(1,200\,000)}$, connected together by

$400\,001$ edges $(u_{(0,i)}, u_{(1,i)})$ for each i. In other words, a grid can be seen as a ladder.



Cactus lady wants to know whether she can embed Cing into this grid, i.e., map each vertex of the cactus onto a separate vertex of the grid while each edge of the cactus will be mapped onto some edge of the grid.

Input

The first line contains an integer t — the number of test cases.

Each test case begins with a line containing two integers n and m — the number of vertices and the number of edges in a given cactus, respectively ($1 \le n \le 200\,000$; $0 \le m \le 250\,000$).

Each of the following m lines contains two integers v and u, describing the edges of the cactus ($1 \le v, u \le n, u \ne v$).

The total sum of all n in the input doesn't exceed $200\,000$.

Output

Print an answer for each test case in the same order the cases appear in the input.

For each test case print "No" in the first line, if no layout exists.

Otherwise print "Yes" in the first line, and the following n lines describing the layout. The i-th of these n lines should contain two integers x_i and y_i, the location of the i-th vertex ($0 \le x_i \le 1$; $-200\,000 \le y_i \le 200\,000$).

Example

input

```
5
4 3
1 2
2 3
3 4

8 7
1 2
3 2
2 4
4 5
4 6
6 7
6 8

5 4
1 2
1 3
1 4
1 5

8 9
1 2
2 3
3 4
1 4
4 5
5 6
6 7
7 8
5 8

10 10
1 2
2 3
3 4
4 5
5 6
6 1
3 7
4 8
1 9
6 10
```

output

```
Yes
```

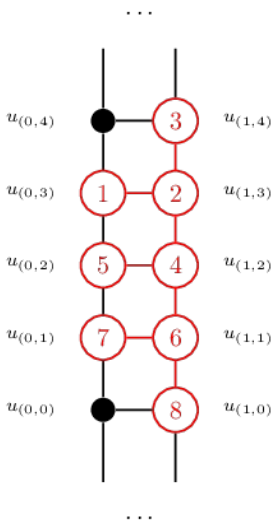
```
0 0
0 1
1 1
1 2
Yes
0 3
1 3
1 4
1 2
0 2
1 1
0 1
1 0
No
Yes
0 0
1 0
1 1
0 1
0 2
0 3
1 3
1 2
Yes
1 1
1 2
1 3
0 3
0 2
0 1
1 4
0 4
1 0
0 0
```

Note

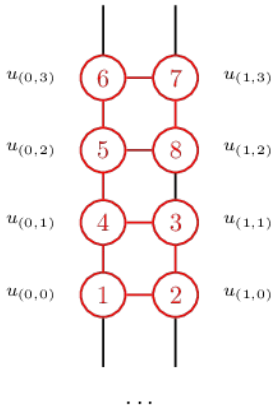
Empty lines between test cases are for clarity. In real test cases there are no empty lines.

In these notes, we consider the embeddings for tests 2 and 4.

We start with the embedding for test 2.



Here goes the embedding for test 4.

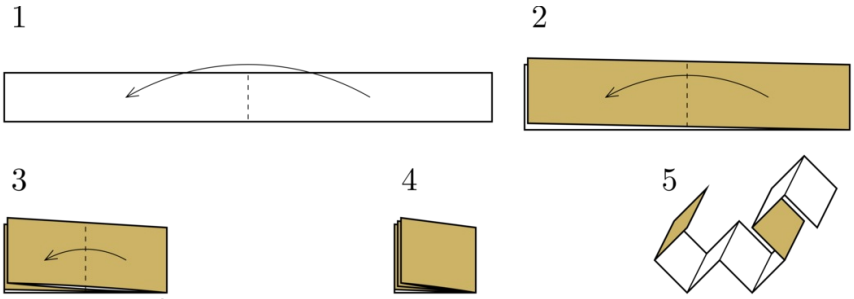


D. Dragon Curve

time limit per test: 5 seconds
memory limit per test: 1024 megabytes
input: standard input

output: standard output

A dragon curve is a self-similar fractal curve. In this problem, it is a curve that consists of straight-line segments of the same length connected at right angles. A simple way to construct a dragon curve is as follows: take a strip of paper, fold it in half n times in the same direction, then partially unfold it such that the segments are joined at right angles. This is illustrated here:



In this example, a dragon curve of order 3 is constructed. In general, a dragon curve of a higher order will have a dragon curve of a lower order as its prefix. This allows us to define a dragon curve of infinite order, which is the limit of dragon curves of a finite order as the order approaches infinity.

Consider four dragon curves of infinite order. Each starts at the origin (the point $(0, 0)$), and the length of each segment is $\sqrt{2}$. The first segments of the curves end at the points $(1, 1)$, $(-1, 1)$, $(-1, -1)$ and $(1, -1)$, respectively. The first turn of each curve is left (that is, the second segment of the first curve ends at the point $(0, 2)$). In this case, every segment is a diagonal of an axis-aligned unit square with integer coordinates, and it can be proven that there is exactly one segment passing through every such square.

Given a point (x, y), your task is to find on which of the four curves lies the segment passing through the square with the opposite corners at (x, y) and $(x + 1, y + 1)$, as well as the position of that segment on that curve. The curves are numbered 1 through 4. Curve 1 goes through $(1, 1)$, 2 through $(-1, 1)$, 3 through $(-1, -1)$, and 4 through $(1, -1)$. The segments are numbered starting with 1.

Input

The first line contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the number of test cases.

Each of the following n lines contains two integers x and y ($-10^9 \le x, y \le 10^9$) — the coordinates.

Output

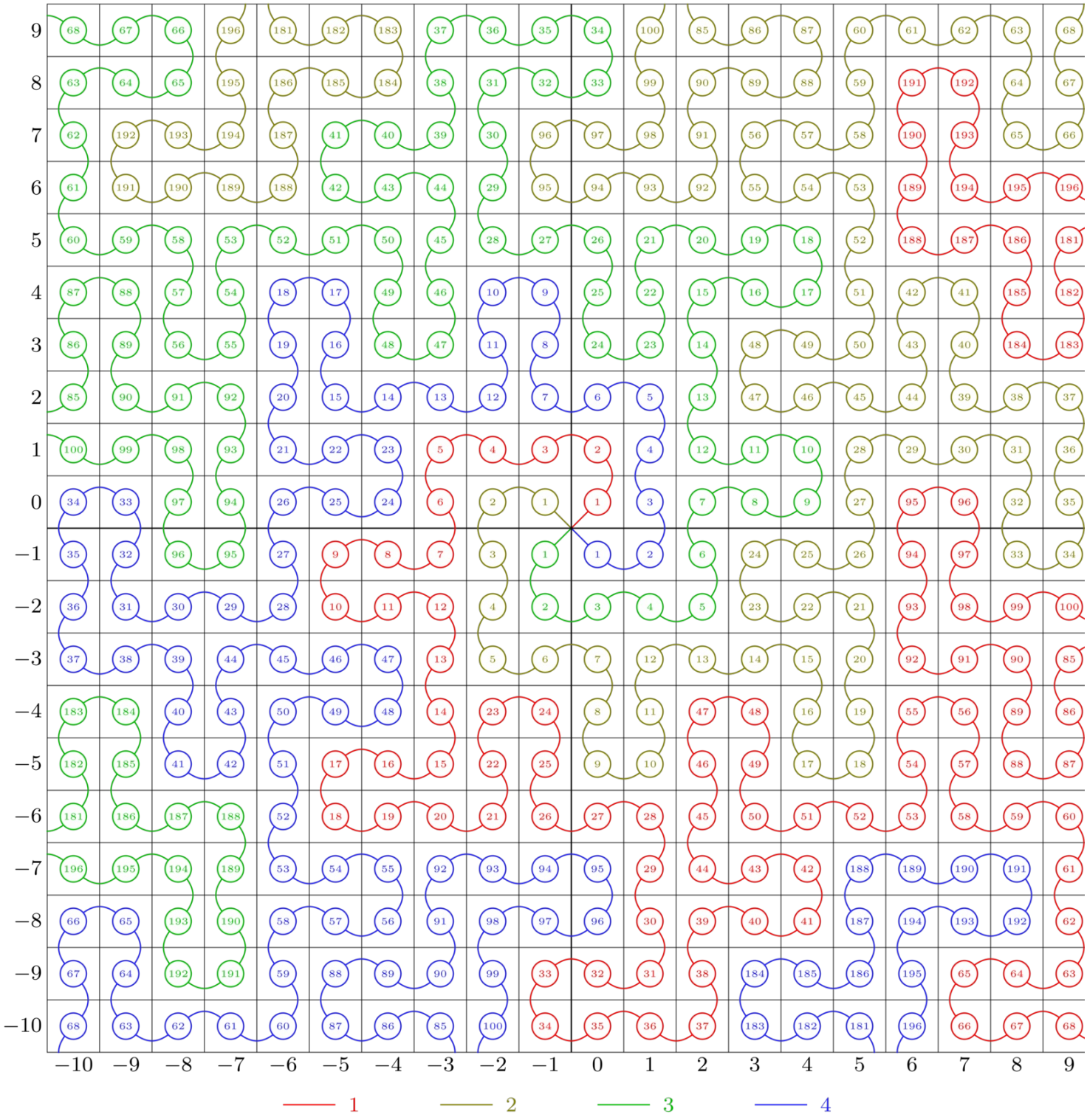
For each test case, print a line containing two integers — the first is the index of the curve (an integer between 1 and 4, inclusive), and the second is the position on the curve (the first segment has the position 1).

Example

input
5 0 0 -2 0 -7 -7 5 -9 9 9
output
1 1 2 2 3 189 4 186 2 68

Note

You can use this illustration to debug your solution:



E. Easy Scheduling

time limit per test: 2 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

Eonathan Eostar decided to learn the magic of multiprocessor systems. He has a full binary tree of tasks with height h. In the beginning, there is only one ready task in the tree — the task in the root. At each moment of time, p processes choose at most p ready tasks and perform them. After that, tasks whose parents were performed become ready for the next moment of time. Once the task becomes ready, it stays ready until it is performed.

You shall calculate the smallest number of time moments the system needs to perform all the tasks.

Input

The first line of the input contains the number of tests t ($1 \le t \le 5 \cdot 10^5$). Each of the next t lines contains the description of a test. A test is described by two integers h ($1 \le h \le 50$) and p ($1 \le p \le 10^4$) — the height of the full binary tree and the number of processes. It is guaranteed that all the tests are different.

Output

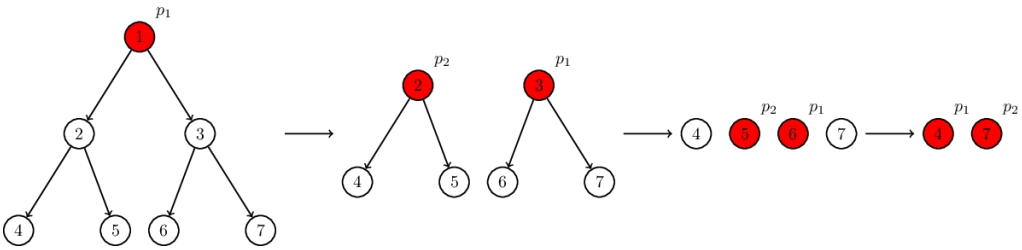
For each test output one integer on a separate line — the smallest number of time moments the system needs to perform all the tasks.

Example

input
3 3 1 3 2 10 6
output
7 4 173

Note

Let us consider the second test from the sample input. There is a full binary tree of height 3 and there are two processes. At the first moment of time, there is only one ready task, 1, and p_1 performs it. At the second moment of time, there are two ready tasks, 2 and 3, and the processes perform them. At the third moment of time, there are four ready tasks, 4, 5, 6, and 7, and p_1 performs 6 and p_2 performs 5. At the fourth moment of time, there are two ready tasks, 4 and 7, and the processes perform them. Thus, the system spends 4 moments of time to perform all the tasks.



F. Framing Pictures

time limit per test: 2 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

Life has been discovered on Venus! What is more, the life forms appear to be convex polygons. An international consortium is designing a probe to send to Venus to take pictures, but they need to estimate the bandwidth needed to send back pictures.

When the probe takes a picture of a life form and wishes to send it back to Earth, the bandwidth required is proportional to the area of the bounding box (in other words, the smallest axis-aligned rectangle that contains the life-form). The shape and size of the life forms are known, but the orientation relative to the camera is random. You must thus determine the expected (average) area of the bounding box across all orientations.

Input

The input describes the shape of a life form as a convex polygon in two dimensions.

The first line of input contains an integer n ($3 \leq n \leq 200\,000$) — the number of vertices. The remaining n lines each contain two integers x and y ($-10^9 \leq x, y \leq 10^9$) — the coordinates of a vertex. The vertices are given in counterclockwise order, and no three vertices lie on a straight line.

Output

Output a single line containing the expected area of the bounding box of the polygon. Your answer should have an absolute or relative error of at most 10^{-6}.

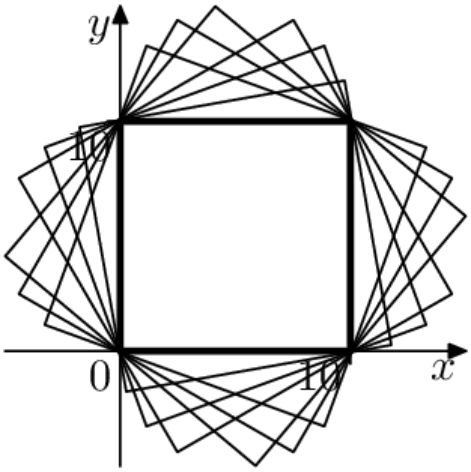
Examples

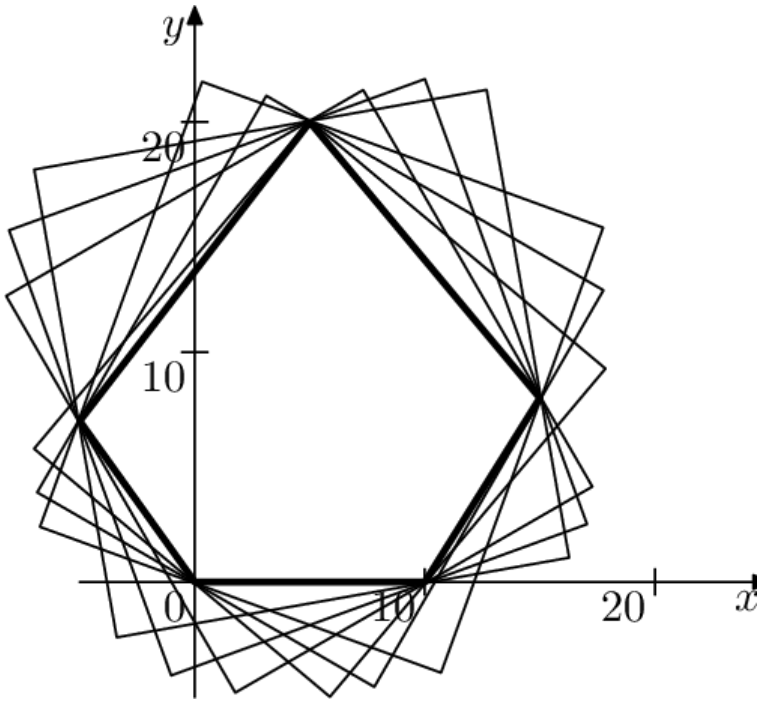
input
4 0 0 10 0 10 10 0 10
output
163.661977237

input
5 0 0 10 0 15 8 5 20 -5 7
output
365.666028588

Note

The pictures show example life forms and various camera orientations.





G. Game of Chance

time limit per test: 6 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

The King wants to marry off his daughter, and he wants her husband to have the greatest innate luckiness possible. To find such a person he decided to hold a heads-or-tails tournament.

If person A with luckiness x and person B with luckiness y play heads-or-tails against each other, person A wins with probability $x/(x+y)$.

The tournament has several rounds. Each round some participants are split into pairs. Each pair plays against each other, and the loser leaves the tournament.

The participants are numbered from 1 to n. During the first round, a number k ($1 \le k \le n$) is selected such that $n - k/2$ is a power of 2 (such k always exists and is unique). Only participants numbered from 1 to k take part in the first round. It ensures that in all other rounds the number of participants is the power of 2.

During other rounds, all the participants who still have not left the tournament take part. If during some round, participants numbered $p_1 < \ldots < p_{2m}$ take part, then they are split into pairs in the following manner: participant p_{2i-1} plays against participant p_{2i} for each i from 1 to m.

The rounds are held until only one participant is left. He is declared the winner of the tournament and he will marry the King's daughter. The princess can't wait to find out who is her future husband. She asked every participant to tell her his luckiness. Assuming they did not lie, she wants to know the probability of each participant winning the tournament. As you are the best friend of the princess, she asks you to help her.

Input

The first line of the input contains the number of participants, n ($2 \le n \le 3 \cdot 10^5$). The second line of the input contains n integer numbers, $a_1, \ldots, a_n$ ($1 \le a_i \le 10^9$). The luckiness of the i-th participant equals to a_i.

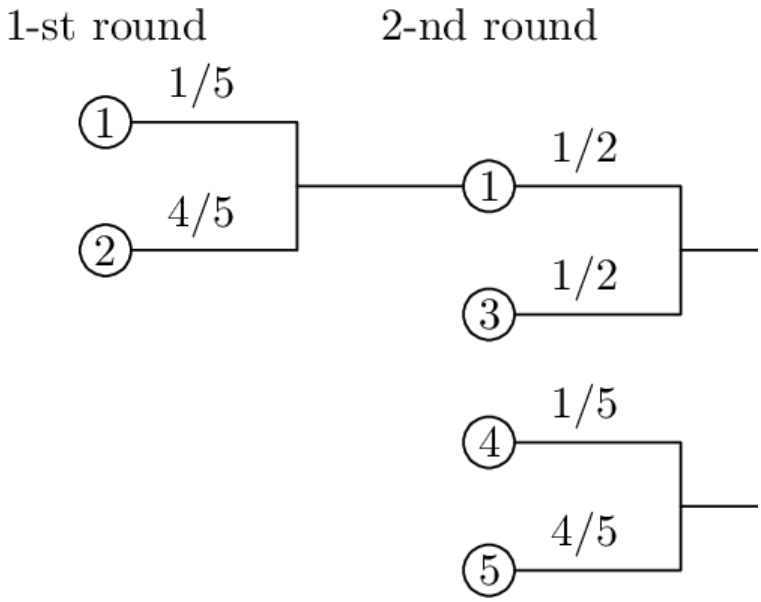
Output

Print n numbers p_i. The i-th number should be the probability of the i-th participant winning the tournament. The absolute error of your answer must not exceed 10^{-9}.

Example

input
5 1 4 1 1 4
output
0.026 0.3584 0.0676 0.0616 0.4864

Note
Here is an example of a tournament bracket, showing the winning probability in each pair.



H. Higher Order Functions

time limit per test: 2 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

Helen studies functional programming and she is fascinated with a concept of higher order functions — functions that are taking other functions as parameters. She decides to generalize the concept of the function order and to test it on some examples.

For her study, she defines a simple grammar of types. In her grammar, a type non-terminal T is defined as one of the following grammar productions, together with $\text{order}(T)$, defining an order of the corresponding type:

- "()" is a unit type, $\text{order}(\text{"()"}) = 0$.
- "(" T ")" is a parenthesized type, $\text{order}(\text{"("}\ T\ \text{")"}) = \text{order}(T)$.
- T_1 "->" T_2 is a functional type, $\text{order}(T_1\ \text{"->"}\ T_2) = max(\text{order}(T_1) + 1, \text{order}(T_2))$. The function constructor T_1 "->" T_2 is right-to-left associative, so the type "()->()->()" is the same as the type "()->(()->())" of a function returning a function, and it has an order of 1. While "(()->())->()" is a function that has an order-1 type "(()->())" as a parameter, and it has an order of 2.

Helen asks for your help in writing a program that computes an order of the given type.

Input
The single line of the input contains a string consisting of characters '(', ')', '-', and '>' that describes a type that is valid according to the grammar from the problem statement. The length of the line is at most 10^4 characters.

Output
Print a single integer — the order of the given type.

Examples

input
()
output
0

input
()->()
output

1

input
()->()->()
output
1

input
(()->())->()
output
2

input
()->((((()->())->()->())->())
output
3

I. Interactive Rays

time limit per test: 2 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

This is an interactive problem.

Your goal is to find a circle on a plane by shooting rays and getting the distance to the circle as a result.

Interactor has three hidden integer parameters that are determined in advance for each test, but which you don't know — x_c, y_c, and r_c. (x_c, y_c) are coordinates of the circle's center and r_c is its radius. The absolute values of x_c, y_c, and r_c do not exceed 10^5, and $1 \le r_c \le \sqrt{x_c^2 + y_c^2} - 1$.

You can shoot rays that extend from the origin $(0, 0)$ and go via a point (x_q, y_q) with the integer coordinates you specify. For each ray, you get a distance from the ray to the circle or 0 if the ray intersects the circle.

Interaction

The interaction starts with your program printing a query to the standard output and finishes when your program finds and prints the answer to the problem.

Each query has a form of "? x_q y_q", where x_q and y_q are integers ($|x_q|, |y_q| \le 10^6$; $x_q \neq 0$ or $y_q \neq 0$).

The interactor outputs a line with a single floating-point number — the distance between a query ray and a circle that is precise to 10^{-10} by an absolute value.

Your program can make the next query, read the output, and so on. You are allowed to do at most 60 queries. At the end of the interaction, print the answer line "! x_c y_c r_c", flush the output and exit.

Note, that the output of the interactor is actually rounded to the 10-th digit after a decimal point, so if you are stress-testing your solution locally, make sure that you also perform the corresponding rounding.

Example

input
? 0 -10 ? 10 -10 ? 10 0 ? 10 10 ? 10 20 ? 10 30 ! 20 10 10
output
12.360679775 11.2132034356 0.0 0.0 3.416407865

5.8113883008

Note

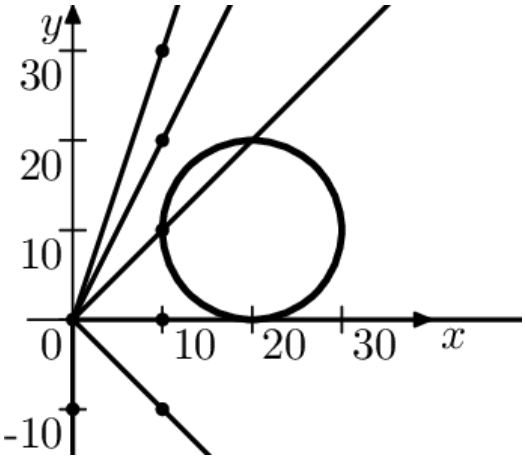


Illustration of the queries from the example interaction.

J. Just Kingdom

time limit per test: 5 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

The Just Kingdom is ruled by a king and his n lords, numbered 1 to n. Each of the lords is a vassal of some overlord, who might be the king himself, or a different lord closer to the king. The king, and all his lords, are just and kind.

Each lord has certain needs, which can be expressed as a certain amount of money they need. However, if a lord, or the king, receives any money, they will first split it equally between all their vassals who still have unmet needs. Only if all the needs of all their vassals are met, they will take the money to fulfill their own needs. If there is any money left over, they will return the excess to their overlord (who follows the standard procedure for distributing money).

At the beginning of the year, the king receives a certain sum of tax money and proceeds to split it according to the rules above. If the amount of tax money is greater than the total needs of all the lords, the procedure guarantees everybody's needs will be fulfilled, and the excess money will be left with the king. However, if there is not enough money, some lords will not have their needs met.

For each lord, determine the minimum amount of tax money the king has to receive so that this lord's needs are met.

Input

The first line of the input contains the number of lords n ($0 \le n \le 3 \cdot 10^5$). Each of the next n lines describes one of the lords. The i-th line contains two integers: o_i ($0 \le o_i < i$) — the index of the overlord of the i-th lord (with zero meaning the king is the overlord), and m_i ($1 \le m_i \le 10^6$) — the amount of money the i-th lord needs.

Output

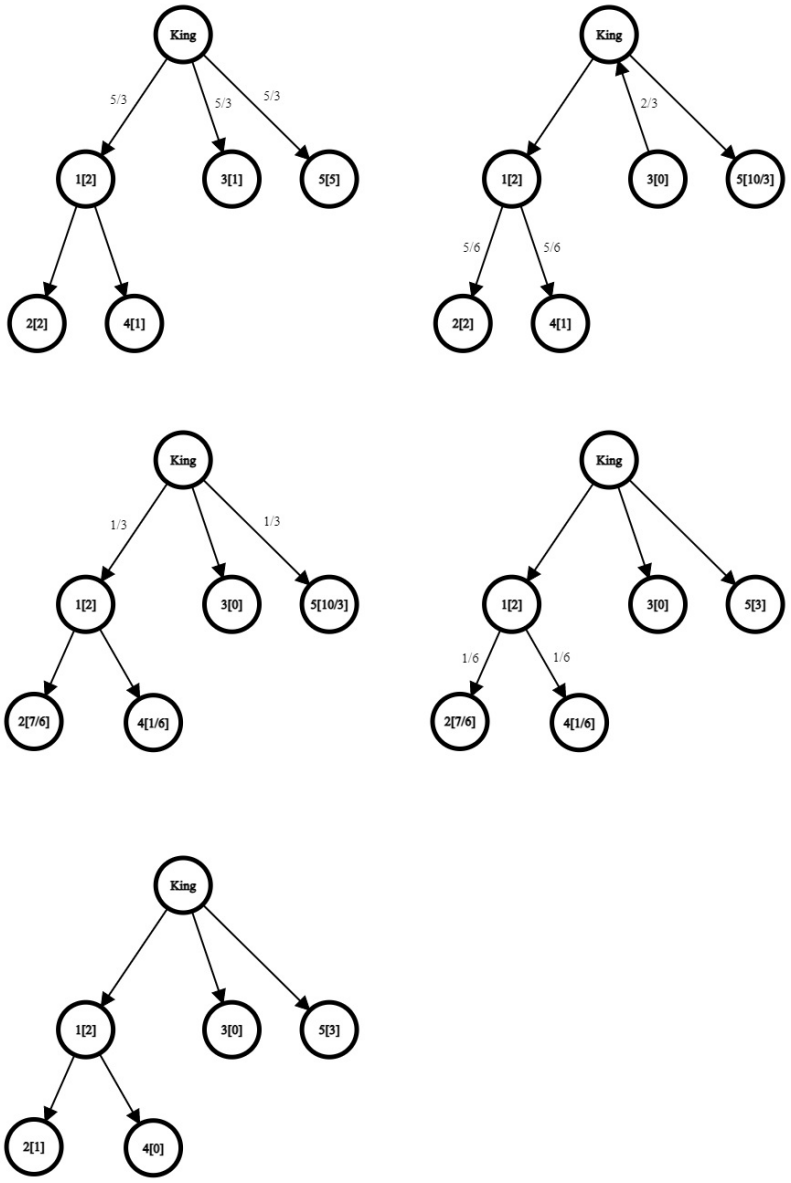
Print n integer numbers t_i. The i-th number should be the minimum integer amount of tax money the king has to receive for which the needs of the i-th lord will be met.

Example

input
5 0 2 1 2 0 1 1 1 0 5
output
11 7 3 5 11

Note

In the sample input, if the king receives 5 units of tax money, he will split it equally between his vassals — the lords 1, 3, and 5, with each receiving $\frac{5}{3}$ of money.



Lord 1 will split the money equally between his vassals — 2 and 4, with each receiving $\frac{5}{6}$. Lord 5 will keep the money (having no vassals). Lord 3 will keep 1 unit of money, and give the remaining $\frac{2}{3}$ to the king. The king will then split the $\frac{2}{3}$ between the vassals with unmet needs — 1 and 5, passing $\frac{1}{3}$ to each. Lord 5 will keep the extra cash (now having a total of 2, still not enough to meet his needs). Lord 1 will split it equally between his vassals, and the extra $\frac{1}{6}$ will be enough to meet the needs of lord 4.

K. Kingdom of Islands

time limit per test: 2 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

The Kingdom of Islands consists of p islands. As the king, you rule over the whole kingdom, while each island is ruled over by one or several jarls under your rule. In total, there are n jarls under your jurisdiction.

Each island of the kingdom has its own strong traditions, so jarls that rule over the same island support each other and never have conflicts. The downsides of such strength are cultural conflicts between people inhabiting different islands. Thus, two jarls that rule over different islands are in conflict.

However, recent years brought a few changes to traditional relations between the jarls. To your knowledge, there are exactly k pairs of jarls such that relationships between two jarls in the pair are different from the traditional. That is, if two jarls of the pair you know rule over the same island, these jarls are in conflict. If they rule over different islands, then they overcome cultural disagreement and there is no conflict between them anymore.

As a true responsible king, you are worried about whether the kingdom is close to a major conflict. In order to estimate the current situation, you would like to find the largest possible group of jarls such that every two jarls in the group are in conflict.

Input

The first line of the input consists of two integers p and n ($1 \le p \le n \le 10^5$; $1 \le p \le 10^4$).

The second line consists of n integers $s_1, s_2, \ldots, s_n$ ($1 \le s_i \le p$). The integer s_i denotes that the i-th jarl rules over the island

number s_i. It is guaranteed that each island is ruled by at least one jarl.

The third line consists of a single integer k ($0 \leq k \leq 20$).

Then k lines follow. The j-th of these lines consists of two distinct integers a_j and b_j ($1 \leq a_j < b_j \leq n$), denoting that the relation between the a_j-th jarl and the b_j-th jarl differs from traditional. It is guaranteed that no pair of jarls appears twice in this list.

Output

In the first line print a single integer q between 1 and n — the largest possible number of jarls in a pairwise conflicting group. In the second line print q distinct integers between 1 and n — the numbers of jarls in the group. The numbers of jarls can be printed in any order.

Examples

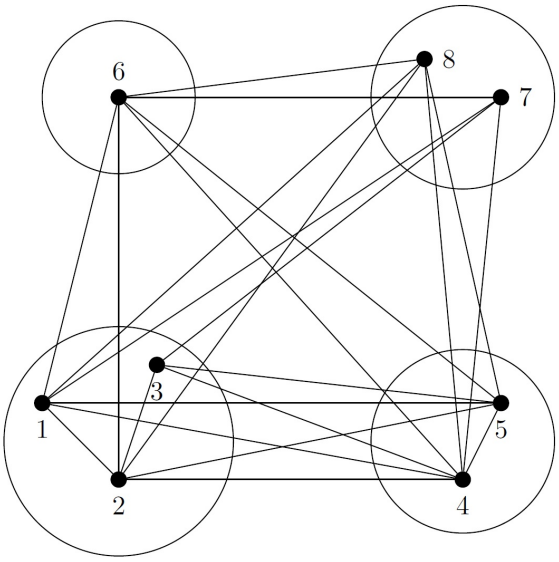
input
4 4 1 2 3 4 1 2 3
output
3 1 4 2

input
2 4 1 1 2 2 1 3 4
output
3 2 4 3

input
4 8 1 1 1 2 2 3 4 4 7 1 2 2 3 3 6 4 5 5 7 2 7 3 8
output
6 8 6 5 4 2 1

Note

The conflict graph for the last sample testcase is given below. Each circle represents an island.



L. Labyrinth

time limit per test: 2 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

In a dream, Lucy found herself in a labyrinth. This labyrinth consists of n rooms, connected by m passages (i-th passage is w_i cm wide). Each passage can be traversed in both directions. It is guaranteed that it is possible to get from any room to any other room. But this is not an ordinary labyrinth — each room in this labyrinth contains a magic candy. When Lucy eats this magic candy, she is getting wider. Specifically, if she eats candy from room i she becomes wider by c_i cm. Note that she is not obliged to eat candy the first time she visits a particular room, but she can eat each candy only once.

Unfortunately, passages in this labyrinth are pretty narrow, so after eating some candy, Lucy can get too wide and will not be able to traverse them — her width should not be greater than the width of the corresponding passage.

Lucy starts her journey in a room number 1. She wants to eat all the candies. After that, she will just wake up, so she does not have to be able to return to the room 1. She realizes that with her current width, she may not be able to do so, so she plans a workout before embarking on her journey. Lucy wants to know if it is possible to start with some positive width and still eat all the candies. If yes, then what is the maximal starting width with which it is possible.

Input

The first line contains two integers, n and m ($2 \le n \le 10^5; n - 1 \le m \le 10^5$) — the number of rooms and the number of passages.

The second line contains n integers — c_i ($1 \le c_i \le 10^9$).

Next m lines contain three integers each — a_i, b_i and w_i ($1 \le a_i, b_i \le n; a_i \ne b_i; 1 \le w_i \le 10^9$) describing passage that connects rooms a_i and b_i and is w_i cm wide. It is guaranteed that the resulting labyrinth is connected and there is at most one passage between any pair of rooms.

Output

If it is possible to eat all the candies, output the maximal possible starting width, otherwise output -1.

Examples

input
3 3 1 2 3 1 2 4 1 3 4 2 3 6
output
3

input
2 1 1 1 1 2 1
output
-1

M. The Mind

time limit per test: 2 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

This is an interactive problem.

In this problem, you need to come up with a strategy for a cooperative game. This game is played by two players. Each player receives 5 cards. Each card has a random integer between 1 and 100 on it. It is guaranteed that all numbers on cards are distinct.

The goal of the game is to play a card with a minimal number on it out of all 10 cards dealt to the players before any other card. The problem is that each player can only see their own cards and cannot communicate with another player in any way.

The game consists of 5 turns. During each turn, players simultaneously make a move. Each player can either play their smallest card or do nothing. If on some turn the smallest card is played, and no other card is played on or before that turn, players win. If two cards are played at the same turn or if after all 5 turns, no card is still played, players lose.

Players cannot communicate, so a strategy for the game should only be based on 5 cards that the player has. You can describe a strategy as five numbers $0.0 \le p_i \le 1.0, \sum_{i=1}^{5} p_i \le 1$, where p_i — the probability of playing the player's smallest card in their hand on i-th turn. If you know the cards dealt to the players, and the strategies that players choose, you can compute the probability of winning by a simple formula.

You will be given $n = 1000$ randomly generated hands of 5 cards. You need to generate a strategy for each of the hands to maximize the probability of winning. After the judge program receives all n strategies, it generates all possible valid pairs of those hands (pairs which have the same numbers are discarded), and computes a probability of winning based on two strategies provided by your program.

To ensure that answers for different hands are independent, you must output a strategy for a hand and flush the standard output before reading information about the next hand.

If the average probability of winning a game is more than 85% over all valid pairs of hands, the test is considered passed. This

problem contains the sample test and 20 randomly generated tests with $n = 1000$.

Input

The first line contains one integer n — the number of hands. It is guaranteed that $n = 1000$ for all cases except the first sample case.

Each of the next n lines contains 5 numbers a_i ($1 \le a_i \le 100, a_i < a_{i+1}$) — the cards in the hand. It is guaranteed that each possible set of 5 cards has an equal probability of being chosen.

Output

For each of the n hands you need to output 5 numbers $0.0 \le p_i \le 1.0, \sum_{i=1}^{5} p_i \le 1$, where p_i — probability of playing the smallest card on i-th turn.

Example

input
2 2 12 27 71 100 22 29 39 68 90
output
  0.8 0.2 0.0 0.0 0.0 0.0 0.2 0.2 0.2 0.3

Note

In the example test there is only one valid pair of hands. The winning probability for the example output is equal to $0.8 + 0.2 \cdot (1 - 0.2) = 0.96$. Also note that the second player will not play a card at all with probability 0.1.