Problem A
Billboards
Time limit: 6 seconds

Billboards in the contest area

Each year, the ICPC (International Collegiate Programming Contest) has many sponsors. Since happy sponsors means happy contest, we plan to add a long billboard on one side of the contest area, so that our lovely sponsors can put their advertisements on it.

There are n wonderful sponsors in total, so to be fair, we would like to give each of them $1/n$ of the total area of the billboard. However, this is complicated by the fact that each marvelous sponsor potentially values sections of the billboard differently. For example, some sponsors want to put their ads near the entrance of the contest area, which may be at the corner where all contestants are guaranteed to see them when they enter. On the other hand, some sponsors may just want to put their ads in the middle, which is more likely to be broadcast in the live stream. (Don't ask us what happens if the entrance is at the middle! It's just an example.)

After talking with all of our delightful sponsors, the ICPC staff finds that the preference values of each sponsor can be modeled as a continuous piecewise linear function, where the domain of each of these value functions is the length of the billboard. Given this information, the ICPC staff wants to split the billboard into n sections, and give them to our n sponsors so that each magnificent sponsor gets at least $1/n$ of the value of the billboard from their point of view. That is, the area of the value function under the section that each sponsor gets must be at least $1/n$ of the total area of their own value function.

Figure A.1 shows a simple example. The billboard length is 10 and there are two sponsors, Orakle Software and Hal++, both of them having a value function with only one piece. Orakle's value function indicates that the company increasingly prefers sections of the billboard as you move left to right, while Hal++'s value function indicates the opposite. The area under each function is the total value of the billboard from each sponsor's perspective. The blue and the green areas shown are both larger than half of each total area, so cutting the billboard in the middle (the dashed line) is a valid solution, resulting in the billboard shown in Figure A.2. Note that there are many other divisions of the billboard that work as well (for example, cuttings at 4 or 6 are also valid solutions, if the sections of billboards are given to the correct sponsor).

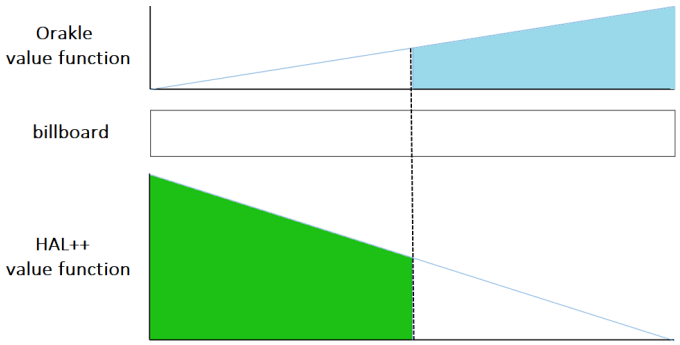


Figure A.1: Sample Input 1

Input

The first line contains two integers: n ($1 \leq n \leq 5\,000$), the number of glorious sponsors that the ICPC has (numbered 1 to n), and l ($1 \leq l \leq 10^6$), the length of the billboard.

Each of the remaining n lines starts with an integer m ($2 \leq m \leq 5\,000$), the number of points specifying this esteemed sponsor's value function. The remainder of the line contains m pairs of integers $(a_1, b_1), \ldots, (a_m, b_m)$ ($0 = a_1 < a_2 < \ldots < a_m = l$, $0 \leq b_i \leq 100$), where each pair represents an endpoint of one section of the piecewise linear function. The sum of all m's is at most $500\,000$ and it is guaranteed that at least one b_i is positive for each sponsor.

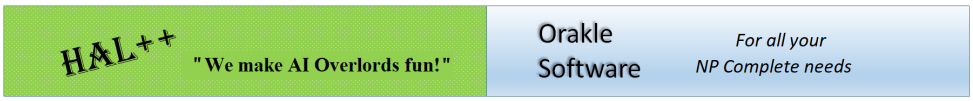


Figure A.2: Final Billboard corresponding to Sample Output 1

Output

Output n pairs of numbers (l_i, f_i) ($1 \leq i \leq n$), which indicate that the billboard section in the range $[l_{i-1}, l_i]$ is given to sponsor f_i (where $l_0 = 0$ is assumed, $l_0 < l_1 < l_2 < \ldots < l_n$, and l_n must be equal to l). The f_i values must be integers in the range $[1, n]$ where each integer appears exactly once, but the l_i can be real numbers. If there are multiple solutions, output any of them. If there is no solution, output `impossible`.

Note that you don't need to optimize anything else, as long as each sponsor gets at least $1/n$ of their total area. Each sponsor's allocated area may be below $1/n$ of their total area if the absolute or relative difference is at most 10^{-8}.

Sample Input 1	Sample Output 1
2 10 2 0 0 10 5 2 0 10 10 0	5 2 10 1

Sample Input 2	Sample Output 2
5 100 5 0 0 10 0 20 1 30 0 100 0 5 0 0 10 0 20 1 30 0 100 0 5 0 0 10 0 20 1 30 0 100 0 5 0 0 10 0 20 1 30 0 100 0 5 0 0 10 0 20 1 30 0 100 0	16.3245553203 1 18.9442719100 2 21.0557280900 3 23.6754446797 4 100 5

Problem B
Bingo for the Win!
Time limit: 1 second

Bingo is a game of chance for multiple players. Each player receives a sheet with some numbers, and a game master then calls out these numbers in a random order. Players cross off the numbers that they have heard, and the first player to cross off all their numbers wins the game. This basic version of the game has a reputation for being, well, a bit sedate. No particular action is required of the players except for not falling asleep.

In this problem we will analyze a more dynamic version of Bingo that requires quick thinking. In our version, called Speed Bingo, the game master also calls out the numbers from the sheets in a random order. However, whenever a number is called out, only the first player to signal that he or she has the number is allowed to cross it off their sheet. If a player has the same number multiple times, only one copy may be crossed off at a time. When multiple players have the same number(s) on their sheets, whoever has the fastest reaction time has an advantage in winning Speed Bingo. But how big an advantage? That's what we need your help to find out.

Formally, there are n players, each receiving a (possibly) different sheet of k (not necessarily distinct) numbers. Player 1 is faster to react than player 2, who in turn is faster than player 3, and so on, with player n being the slowest. Consider the following example, corresponding to the first sample input, where three players receive four numbers each:

1	2
3	4

Player 1

1	2
5	6

Player 2

3	4
7	8

Player 3

When number "1" is called for the first time, player 1—being faster—will get to cross it off their sheet. The second time "1" is called, player 2 will get to cross it off. So on average, we would expect player 1 to do better than players 2 and 3, since both of them will need some numbers that player 1 will get to first. However, since players 2 and 3 have no numbers in common, their performances will be independent of each other, even though player 2 is faster than player 3.

Suppose the game is played until all players have crossed off all of their numbers, that is, until all $n \cdot k$ numbers on all of the sheets (including appropriate repetitions) have been read. Assuming the order of the numbers is uniformly random, how likely is it for each player to finish last?

Input

The input describes a single game of Speed Bingo. The first line contains two integers n and k, the number of players and number of numbers on each sheet ($1 \leq n \leq 100$, $1 \leq k \leq 1\,000$). This is followed by n lines containing k integers each, where the i^{th} line gives the numbers on the sheet for the i^{th} player. All those numbers are between 1 and 10^9, inclusive.

Output

Output n lines, one for each player. The i^{th} line should contain the probability that player i finishes last. All values must be accurate to an absolute error of at most 10^{-6}.

Sample Input 1	Sample Output 1
3 4 1 2 3 4 1 2 5 6 3 4 7 8	0.000000000 0.500000000 0.500000000

Sample Input 2	Sample Output 2
4 2 1 2 3 4 10 5 7 8	0.250000000 0.250000000 0.250000000 0.250000000

Problem C
Citizenship
Time limit: 4 seconds

It has been a long time since you moved to a different country and you have decided it is time to become a citizen. Your new country has a strict residency requirement for all applicants. To apply, you must have been physically present in the country for at least d days per year, for the past y consecutive years. These years are counted in 12-month periods backwards from the application date.

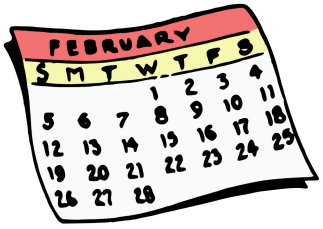


Image via Rawpixel, CC0

For this problem, assume that a calendar year has 12 months of 365 days, and each month has exactly the number of days below:

month	01	02	03	04	05	06	07	08	09	10	11	12
days	31	28	31	30	31	30	31	31	30	31	30	31

For example, if you were to apply on 2024–09–19 you must have been in the country for at least d days during the 12-month periods from 2023–09–19 to 2024–09–18, 2022–09–19 to 2023–09–18, and so on for y such periods.

You have lived in the country for at least y years, but having traveled a lot, you are not sure if you meet the residency requirement. Write a program that finds the earliest date you can submit your citizenship application given your travel history.

Input

The first line contains three integers n, y and d ($1 \leq n \leq 500$, $1 \leq y \leq 1\,000$, $1 \leq d \leq 365$). You have been out of the country n times and y and d specify the country's residency requirement as described above.

Each of the following n lines contains two dates in the form `YYYY-MM-DD` ($0000 \leq$ `YYYY` $\leq 5\,000$, $01 \leq$ `MM` ≤ 12, $01 \leq$ `DD` ≤ 31). You have been out of the country between the two dates, inclusive. All dates in the input are sorted in increasing order. The only dates which may be equal are dates on the same line. All given dates are valid.

Output

Output the earliest date on which you meet the residency requirement. The date must be after the last date of the input.

Sample Input 1	Sample Output 1
3 5 240 2022-02-28 2022-10-01 2022-11-11 2022-11-11 2023-12-30 2024-01-01	2024-05-31

Sample Input 2	Sample Output 2
3 5 240 2011-11-11 2012-12-12 2022-02-28 2022-10-01 2025-01-01 2025-06-30	2028-02-26

Problem D
Doubles Horseback Wrestling
Time limit: 4 seconds

The Nomadic Games Exploratory Committee (NGEC) is floating the idea of a doubles horseback wrestling tournament with pairs of riders astride single horses. They have advertised a pilot tournament, and n eager riders have signed up to compete! So now the NGEC needs to pair the riders in order to make the tournament both fair and exciting.

The Central Asian Audaryspak League (CAAL) maintains a list of all horseback wrestlers and their ratings. From their previous experience with ordinary horseback wrestling, the NGEC has decided that the pairs are best balanced if the ratings of their two riders add up to a particular integer, s.

For obscure licensing reasons, the CAAL refuses to release the exact rating of each rider. But the NGEC has some good estimates, knowing that any rider i's true rating r_i lies in an interval $[l_i, u_i]$. So the NGEC would consider pairing two riders i and j if there are ratings $r_i \in [l_i, u_i]$ and $r_j \in [l_j, u_j]$ such that $r_i + r_j = s$.

The NGEC wants to form as many non-intersecting pairs of riders as possible. You need to help them.

Horseback wrestling in Kyrgyzstan by Theklan
Wikimedia Commons, CC BY-SA 4.0

Input

The first line contains two integers n and s ($2 \le n \le 2 \cdot 10^5$, $1 \le s \le 10^9$), denoting the number of riders and the desired sum of ratings of riders in a pair. Riders are numbered 1 to n. This is followed by n lines, where the i^{th} line contains two integers l_i and u_i ($1 \le l_i \le u_i \le 10^9$), denoting the rating range of the i^{th} rider.

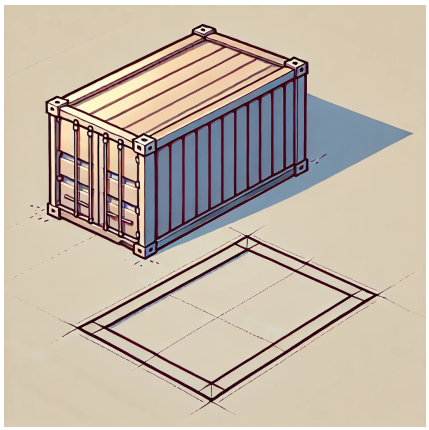
Output

Output k, the maximum number of riding pairs that can be formed, followed by k pairs of integers, denoting the numbers of the riders forming each pair. If there are multiple ways to pair off the riders, output any one.

Sample Input 1	Sample Output 1
6 10 6 7 1 4 2 2 3 8 5 7 9 9	2 6 2 3 4

This page is intentionally left blank.

Problem E
Flipping Container
Time limit: 2 seconds



Generated by ChatGPT 4o

A large, cuboid-shaped shipping container has been delivered to your shipping yard. Before you can open it up, you first need to move the container to a different location in your yard.

Normally, you would simply lift up the container using a crane and drop it in its target location. Unfortunately, your crane is broken, and the only way for you to move the container is by carefully flipping the container over one of its bottom four edges, resulting in a 90-degree rotation around the axis running along that edge. Your hope is that you will eventually be able to reach the target location by repeating this action enough times.

Just how many times will you need to perform this flip operation to get the container to the right spot? Note that the orientation of the container changes as you move it across the yard, but in the end it needs to be in the same orientation as at the start. Two orientations are considered the same when they have the same length along each of the three axes of the coordinate system. The three side lengths of the container are distinct.

Input

The first line contains three distinct integers a, b and c ($1 \leq a, b, c \leq 1\,000$), the dimensions of the container in meters. In the initial orientation of the container, the sides of length a run in east-west direction, the sides of length b run in north-south direction, and the sides of length c run in up-down direction.

The second line contains two integers x and y ($-10^{18} \leq x, y \leq 10^{18}$) giving the target location of the container. The container needs to be moved by x meters in the east-west direction and by y meters in the north-south direction (positive numbers indicate moving to the east/north, negative numbers indicate west/south).

Output

Output the least number of times the container needs to be flipped to reach its target location. If it is not possible to reach the target location, output `impossible`.

Sample Input 1	Sample Output 1
3 4 5 8 0	2

Sample Input 2	Sample Output 2
3 4 5 −8 9	4

Sample Input 3	Sample Output 3
3 4 5 123 45	40

Sample Input 4	Sample Output 4
20 10 30 13 37	impossible

Problem F
Friendly Rivalry
Time limit: 2 seconds

The leaders of the International Coalition for Planetary Change (ICPC), a non-profit fighting for environmental awareness, are worried that their regional chapters are not doing enough to make a real impact on climate change. Inspired by the latest studies that competition is one of the best motivators, they have decided to start a competition between their chapters.

At the same time, the ICPC does not want to slow the spread of ideas. To encourage chapters to share effective climate change methods, the ICPC has decided to assign their $2n$ chapters into two teams, the green team and the blue team. For balance, each team should consist of exactly n chapters.

To ensure the teams are not getting in each other's way, the ICPC wants the two teams to be as far apart as possible. Specifically, the Euclidean distance between the closest pair of chapters belonging to different teams should be as large as possible.

Help the ICPC set up the teams according to these rules.

Input

The first line contains an integer n ($1 \leq n \leq 500$), the size of each team. Chapters are numbered from 1 to $2n$. Each of the remaining $2n$ lines contains two integers x_i and y_i ($-10^9 \leq x_i, y_i \leq 10^9$), the Cartesian coordinates of the location of the i^{th} chapter. All chapters are at distinct locations.

Output

Output $n + 1$ numbers. The first number is the distance between the two closest chapters belonging to different teams. The next n numbers are the chapters belonging to the blue team. If there are multiple ways to divide the teams with the same minimal distance, output any of them. The distance should have an absolute or relative error of at most 10^{-6}.

Sample Input 1	Sample Output 1
2 0 1 1 0 1 1 0 0	1.000000 1 2

Sample Input 2	Sample Output 2
2 0 1 -1 -1 1 0 2 2	2.236068 4 2

Sample Input 3	Sample Output 3
3 0 0 1 1 2 2 3 3 4 4 5 5	1.414214 1 2 3

Problem G
Kindergarten
Time limit: 2 seconds

Generated by ChatGPT 4o

Taking a group of kindergarten kids to the planetarium isn't easy. You really wanted to do this, to allow every kid a chance to get into the room with the giant telescope and take a look at Jupiter. And now that you're going, you remember the stories from other caretakers that kids can misbehave and leave some nasty surprises in the telescope room for the kids after them. You really want to avoid that.

You know the kids in your group very well. Each kid is jealous of one other kid, who is cooler than them, and they might misbehave in the telescope room if they know the kid they're jealous of will be there at some point after them—not necessarily immediately after them, just at some later point. You thought this would be easy—the coolest kid in the class doesn't have anyone to be jealous of, so she can go first, and then all the other kids in order of coolness. However, you just learned that the coolest kid is an exception—instead of being jealous of someone cooler than herself, she's jealous of some other random kid in the group. This sounds like a disaster!

Fortunately, you also know that each kid has some other kid they really, really like. So whenever a kid in the telescope room is thinking about setting up a surprise, if they know the kid they like is going to be in the room after them and before the kid they're jealous of, they will refrain from misbehaving. To make this formal, if a kid A is jealous of kid B, and really likes kid C, then there's a risk A will misbehave and set up a surprise in the telescope room if B will be in the telescope room after A, and C will be there either before A or after B.

Can you figure out an order in which the kids can go to the telescope room so no surprises occur?

Input

The first line contains an integer n ($3 \leq n \leq 200\,000$), the number of kids in your group. The kids are indexed from 1 to n in decreasing order of coolness. Each of the next n lines describes one of the kids. The i^{th} of these lines contains two integers: j_i, the index of the kid that the i^{th} kid is jealous of, and l_i, the index of the kid that the i^{th} kid really, really likes ($1 \leq j_i, l_i \leq n$, $j_i \neq l_i$, $j_i \neq i$, $l_i \neq i$, and $j_i < i$ for all i except 1).

Output

Output a line containing n integers, the order in which the kids should enter the telescope room. If there are multiple ways to order the children, output any of them. If no order exists output `impossible`.

Sample Input 1	Sample Output 1
4 4 2 1 4 2 4 2 1	1 2 3 4

Sample Input 2	Sample Output 2
4 2 3 1 4 2 1 1 2	impossible

Problem H
Maxwell's Demon
Time limit: 6 seconds

Relax: No knowledge of thermodynamics is needed to solve this problem.

Maxwell's demon sits in a container of height h and width $2w$. The container is divided into two adjacent chambers, each of height h and width w. An impenetrable wall separates the two chambers, and the demon sits at a fixed position on this center wall.

The chambers contain particles, each particle having a position, a velocity, and a color. When a particle strikes the wall of a chamber, it reflects off the wall with perfect elasticity. Figure H.1 shows a container with two particles, a blue one in the left chamber and a red one in the right chamber.

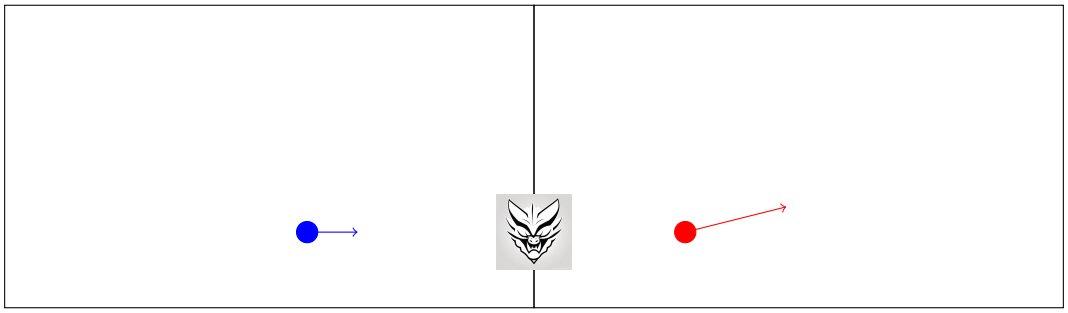


Figure H.1: The first sample input. The demon is shown enlarged in gray, while in reality it is infinitesimally small (and does not have a face).

Maxwell's demon seeks to reduce entropy by sorting the particles. It wants all red particles to be in the left chamber and all blue particles to be in the right chamber. To achieve this, it has a special power: it can allow particles to pass through the center wall when they hit the demon's position.

The chamber bottoms lie on the x-axis, and their dividing center wall runs along the positive y-axis. At any time of the demon's choosing, all particles at the demon's position $(0, d)$ will not reflect off the center wall but will instead pass through to the other chamber, maintaining their velocity. The demon can do this whenever and as often as it wants and can also choose not to allow a particle through even though it hits the demon's position. However, if multiple particles are at position $(0, d)$ simultaneously, either all such particles pass through the center wall or all particles are reflected.

Help the demon sort the particles and reduce entropy! What is the earliest time when it is possible for all red particles to be in the left chamber and all blue particles to be in the right chamber?

Input

The first line consists of five integers w, h, d, r, b, where w and h ($2 \leq w, h \leq 200$) are the width and the height of each chamber, d ($0 \leq d \leq h$) is the y-coordinate of the demon's position on the container's center wall, and r and b ($0 \leq r, b$ and $1 \leq r + b \leq 200$) are the number of red and blue particles, respectively.

This is followed by $r + b$ lines, each describing a single particle using four integers p_x, p_y, v_x, v_y, where (p_x, p_y) ($0 < |p_x| < w$, $0 < p_y < h$) is the initial position of the particle, and (v_x, v_y) ($|v_x| < w$, $|v_y| < h$, $(v_x, v_y) \neq (0, 0)$) is the initial velocity of the particle. The first r particles described are red while the remaining are blue.

Output

Output the least amount of time needed for all red particles to be in the left chamber and all blue particles to be in the right chamber. Your answer should have an absolute or relative error of at most 10^{-6}. If it is impossible for all red particles to be in the left chamber and all blue particles to be in the right chamber within a finite amount of time, output `impossible`.

Sample Input 1	Sample Output 1
7 4 1 1 1 2 1 4 1 -3 1 2 0	24.0

Sample Input 2	Sample Output 2
4 4 1 2 2 3 1 2 2 -2 3 -2 -1 3 2 1 -2 -2 2 2 2	impossible

Problem I
Steppe on It
Time limit: 3 seconds

The gas pedal on the floor. Squealing tires. Wailing sirens. Emergency vehicles do whatever is necessary to reach their target locations as quickly as possible. Time is critical because lives often depend on it.


Kazakh Steppe by Carole a via Wikimedia Commons, CC BY-SA 3.0

Providing emergency services is always challenging, especially for sparsely populated areas such as the Kazakh Steppe. The cost of building infrastructure is high compared to the number of people served. It is therefore important to minimize both the number of roads and the number of vehicles. On the other hand, it is also vital to minimize the response time of emergency services.

This problem considers a road network that already minimizes the number of roads, which means that any two villages are connected by exactly one path. Thanks to a government grant, the Kazakh Steppe Fire Department recently acquired some shiny new fire engines. The department wants to establish fire stations in some of the villages and allocate the fire engines to them in a way that optimizes the guaranteed response time.

Your task is to find an optimal placement of fire engines that minimizes the time needed for any village to be reached by a fire engine. You can neglect the time needed to assemble the fire crew and start the engine as well as the time to travel across any villages. The response time is determined solely by traveling along the roads.

Input

The first line contains two integers: the number of villages n ($1 \leq n \leq 100\,000$) and the number of fire engines f ($1 \leq f \leq n$).

This is followed by $n-1$ lines numbered from 2 to n. Line number i contains two integers v_i ($1 \leq v_i < i$) and t_i ($1 \leq t_i \leq 10\,000$) meaning that there is a two-way road between villages i and v_i that can be traveled in time t_i.

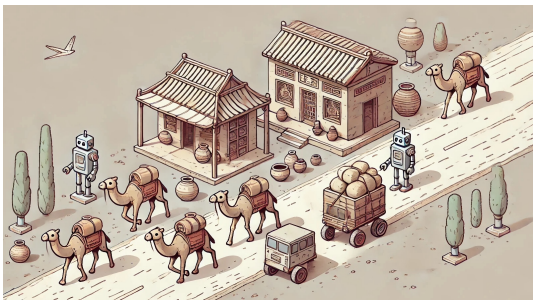
Output

Output the minimum response time that can be guaranteed by placing fire engines into f villages.

Sample Input 1	Sample Output 1
6 2 1 8 2 7 2 7 3 6 3 5	8

Sample Input 2	Sample Output 2
3 3 1 1000 2 1000	0

Problem J
The Silk Road . . . with Robots!
Time limit: 5 seconds



Generated by ChatGPT 4o

Parts of the ancient silk road passed through southern Kazakhstan. You've been fantasizing about a modern silk road, which has its own special features. Along your fantasy road are robots as well as stores holding stashes of tenges (the national currency of Kazakhstan). If a robot moves to a location with a store, the robot collects all of that store's tenges for you.

The cost of moving a robot is 1 tenge for every meter moved. So the amount of profit from moving a robot to a store is the number of tenges held by the store minus the number of meters the robot has moved to reach the store.

Consider this scenario, which stretches over several days. Initially, the road is empty, with no robots or stores. Every day, either a new robot or a new store is placed on an unoccupied location along the road. Immediately before that, each existing store on the road is resupplied with tenges so that its total amount is the same as it was when it was first placed on the road, and each robot is returned to its original starting location.

For each day, you need to determine the maximum amount of profit that could be gained by moving robots to collect tenges from the stores. Note that no two robots start in the same location, but they may occupy the same location as they move. Each store can be emptied of its tenges only once during a single day.

Input

The first line contains an integer n ($1 \le n \le 2 \cdot 10^5$), the number of days. This is followed by n lines, where the i^{th} line starts with an integer t_i, which is equal to 1 if a new robot is added on day i, or is equal to 2 if a new store is added that day. It is followed by an integer x_i ($0 \le x_i \le 10^8$), denoting the location of the new robot or the new store. If $t_i = 2$, the line contains another integer c_i ($0 \le c_i \le 10^8$), denoting the number of tenges at the store. All the given locations are distinct.

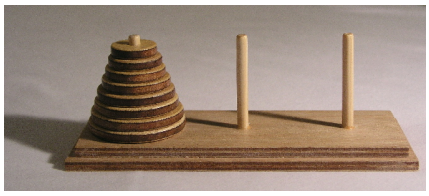
Output

Output n integers, the maximum profit you can make after each day.

Sample Input 1	Sample Output 1
6 1 20 2 15 15 2 40 50 1 50 2 80 20 2 70 30	0 10 35 50 50 60

This page is intentionally left blank.

Problem K
Tower of noiHa
Time limit: 1 second



Tower of Hanoi via Wikimedia Commons, CC BY-SA 3.0

Lucas believes that at six years old, his son is ready to learn some basic algorithms. To start, he chose one of the most beautiful techniques: recursion, and to illustrate it, he picked the well-known recursion game: the Tower of Hanoi.

The Tower of Hanoi is a mathematical game consisting of three rods and a number of disks of various diameters, which can slide onto any rod. The puzzle begins with the disks stacked on the first rod in order of size, the smallest at the top, thus approximating a conical shape. The objective of the puzzle is to transfer the entire stack to the last rod, obeying the following rules:

- Only one disk may be moved at a time.
- Each move consists of taking the top disk from one of the stacks and placing it on top of another stack or on an empty rod.
- No disk may be placed on top of a disk smaller than itself.

Lucas knows that the minimal number of moves required to solve a Tower of Hanoi puzzle is $2^n - 1$, where n is the number of disks. What's more, the optimal moves are unique, which means that n and the number of moves that have been done uniquely represent the current state of the game, given that the disks are always moved optimally.

Lucas was showing his son how to solve the game step by step. He has already done the first k optimal moves. Since it will still take a while to finish, he took a short break to grab some snacks. Unfortunately, when he came back, he found that his naughty little son has done a big "move": knowing that the goal is to transfer all disks from the first rod to the last rod, his son literally transferred "all disks from the first rod to the last rod" in one move (without changing their respective order), see figure K.1.

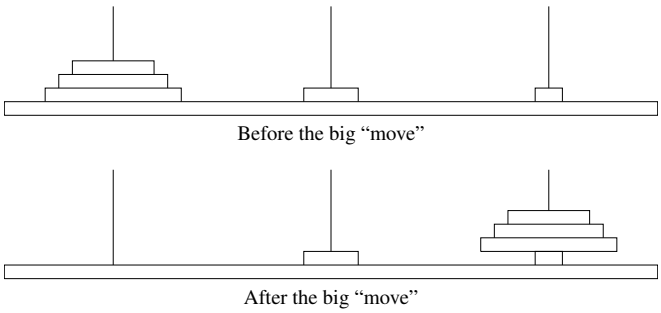


Figure K.1: Layout of the game before and after the son's big "move".

Lucas believes that he can still use this as a teaching opportunity. He decides to solve the game still using only "valid" moves. However, he wonders what is the current minimum number of moves required to solve the game. Since he is also busy dealing with his son, he needs your help!

Note that a "valid" move is still well-defined even if the given state is invalid. That is, you can only move one top disk at a time, and you cannot place it on top of another disk that is smaller than it. In particular, it is valid to put a disk of size a on top of a rod that contains a disk of size b ($b < a$) if the top disk on this rod has size c ($a < c$).

Input

The first line contains an integer n ($1 \leq n \leq 200\,000$), the number of disks in the game.

The second line contains an integer k ($0 \leq k \leq 2^n - 1$), the number of optimal moves Lucas did prior to the big move. Note that k is given in binary.

Output

Output one integer in binary, the minimum number of moves required to finish the game.

Sample Input 1	Sample Output 1
3 0	0

Sample Input 2	Sample Output 2
3 10	110

Sample Input 3	Sample Output 3
5 11011	11

Problem L
Where Am I Now?
Time limit: 5 seconds

Who am I? What am I? Why am I? These are all difficult questions that have kept philosophers reliably busy over the past millennia. But when it comes to "Where am I?", then, well, modern smartphones and GPS satellites have pretty much taken the excitement out of that question.

But what if you have no GPS at hand? In one of the World Finals 2021 problems, the Instant Cartographic Positioning Company (ICPC) demonstrated a way to determine your current location using spiral movements and observing your surroundings. Unfortunately, their method can only be used in open areas where you can move freely without any obstacles. What if you need to locate your exact position in a closed space? How can you do it? Well, now is the time to find out.

You are given a map of an area consisting of unit squares, where each square is either open or occupied by a wall. At the beginning, you are placed in one of the open unit squares, but you do not know which square it is or what direction you face. Any two individual open spaces are indistinguishable, and likewise for walls. You may walk around the area, at each step observing the distance to the next wall in the direction you face. The goal is to determine your exact position on the map.

Interaction

The first line of input contains two integers r and c ($1 \leq r, c \leq 100$) specifying the size of the map. This is followed by r lines, each containing c characters. Each of these characters is either a dot (`.`) denoting an open square, or a number sign (`#`) denoting a square occupied by a wall.

At least one of the squares is open. You know you start in one of the open squares on the map, facing one of the four cardinal directions, but your position and direction is not given in the input. All squares outside the map area are considered walls.

Interaction then proceeds in rounds. In each round, one line becomes available, containing a single integer d ($0 \leq d \leq 99$) indicating that you see a wall in front of you at distance d. This means there are exactly d open squares between your square and the closest wall in the current direction. You should then output a line containing one of the following:

- `left` to turn 90 degrees to the left,
- `right` to turn 90 degrees to the right,
- `step` to move one square forward in your current direction,
- `yes` i j to claim that your current position is row i, column j ($1 \leq i \leq r$, $1 \leq j \leq c$),
- `no` to claim that no matter what you do, it will not be possible to reliably determine your position.

If you output `yes` or `no`, interaction stops and your program should terminate. Otherwise, a new interaction round begins. In order to be accepted, your solution must never `step` into a wall, and can run for at most 100 000 interaction rounds (the final round where you only report `yes` or `no` counts towards this limit).

Read	Sample Interaction 1	Write
3 3 ##. #.. ... 1		
	right	
1		
	step	
0		
	left	
0		
	right	
0		
	right	
1		
	yes 2 2	

Read	Sample Interaction 2	Write
3 5 ##.## ###.# .#.## 0		
	left	
0		
	no	

Read	Sample Interaction 3	Write
2 1 # . 0		
	yes 2 1	