Problem Set

Please check that you have 12 problems that are spanned across 24 pages in total (including this page).

Problem A
Autonomous Vehicle
Time Limit: 1 Second

One autonomous vehicle V is deployed in a mega factory with vertical and horizontal roads in the figure below. The roads are either horizontal or vertical segments. A vertical road and a horizontal road can meet at most once. If they intersect, then they intersect properly, that is, all the intersections are four-way. The end points of segments have integer coordinates.

The vehicle V starts at one end of some road at time $t = 0$ and moves at one unit per second according to a driving algorithm. V goes straight along the road until it meets either an intersection point of two roads or an end of the road. If it reaches the intersection, it turns left at that intersection. If it reaches the end of the road, it bounces back at that end and keeps moving according to the algorithm. The vehicle moves forever even when it comes back at the starting point. The time that V needs to take when it bounces back at the end of the road or turns left at the intersection point is negligible.

Let us explain with an example in Figure A.1(a). The starting point of the vehicle is the end point a. Then it first reaches the intersection b, turns left and moves toward the end point c. It bounces back at c and reaches b again. Then it turns left and moves along the part between b and d. It reaches another intersection d and turns left toward e, and so on. As in Figure A.1 (b), some parts of the segments are not traversed by the vehicle due to the turning rule of the algorithm, and the vehicle continues to move even when it comes back to the starting end point.

Given the roads of the factory, the starting end point of the vehicle V and a nonnegative time t, write a program to compute the coordinate (x, y) of the location of V at time t.

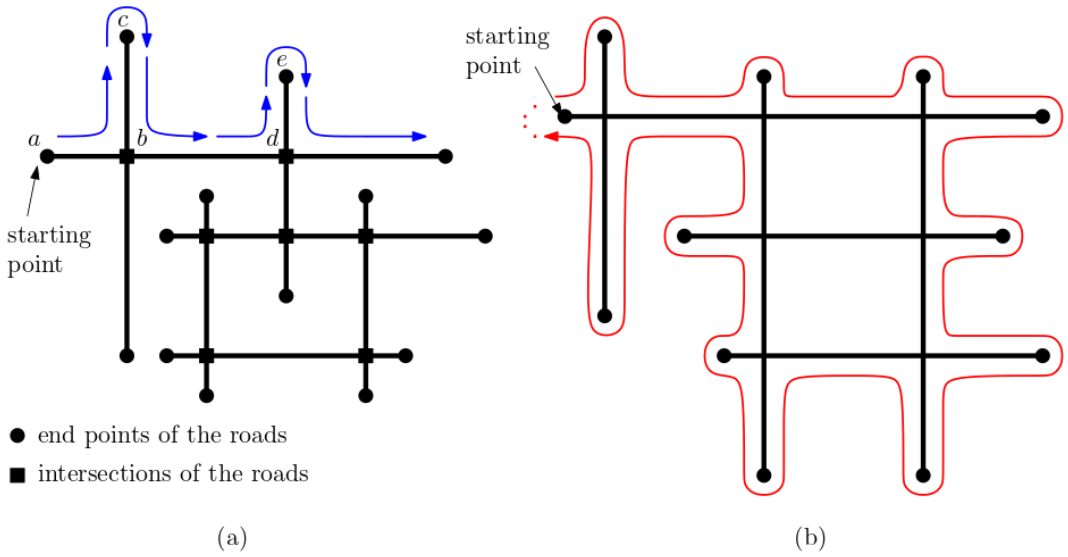


Figure A.1 (a) The roads in the factory. (b) The vehicle comes back to the starting point.

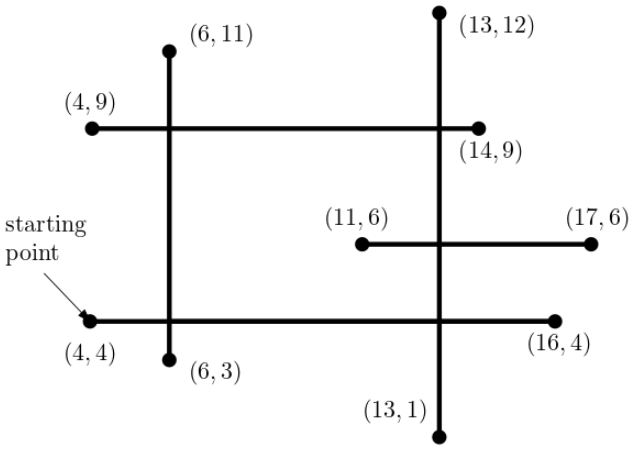


Figure A.2 The roads given in the second sample test case below.

Input

Your program is to read from standard input. The input starts with a line containing two integers, n $(2 \leq n \leq 500)$ and t $(0 \leq t \leq 10^9)$, where n is the number of roads, i.e., the number of line segments and t is the time at which you should output the location of the vehicle. In the following n lines, the i-th line contains four nonnegative integers bx_i, by_i, ex_i, ey_i that represent two end points (bx_i, by_i) and (ex_i, ey_i) of a (horizontal or vertical) segment. Two segments intersect at most one point. The starting end point for the vehicle is designated as (bx_1, by_1). All coordinates of the end points are between 0 and 10^7. No roads share the end points. Note that road segments are not necessarily connected and all the intersections are four-way.

Output

Your program is to write to standard output. Print two integers x and y representing the coordinate (x, y) of the vehicle at time t.

The following shows sample input and output for two test cases. Figure A.2 corresponds to the second sample test case.

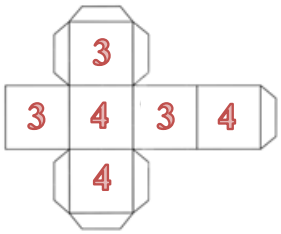
Sample Input 1	**Output for the Sample Input 1**
5 33 4 4 16 4 4 9 14 9 6 11 6 3 11 6 17 6 13 12 13 1	13 6

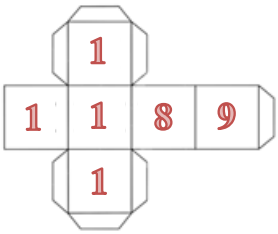
Sample Input 2	**Output for the Sample Input 2**
5 70 4 4 16 4 4 9 14 9 6 11 6 3 11 6 17 6 13 12 13 1	6 6

Problem B
Commemorative Dice
Time Limit: 0.5 Seconds

Since the year 2000, an ICPC regional contest has been held every year in Korea. To commemorate the 21st regional contest this year, it is decided to make a dice. The commemorative dice is a regular cube with a positive number written on each of its sides like an ordinary dice; however, the six numbers are not necessarily to be 1, 2, 3, 4, 5, 6 but just their sum is 21.

The dice can be used in various ways. For example, two people can play a game as follows: Each of the two picks one out of the many dice and then rolls the dice. The winner is the one who tosses a bigger number. It is important which dice to choose in this game because once the dice are set, the probability of one winning against the other is determined. Suppose that KyungYong chooses the dice shown in the figure below left and TaeCheon chooses the dice shown in the figure below right. Then, KyungYong wins when and only when TaeCheon tosses number 1, so the probability that KyungYong wins is 2/3.





Given the dice of the first and second players, write a program to calculate the probability of the first player winning.

Input
Your program is to read from standard input. The input consists of two lines. The first line contains six positive integers that are written on the sides of the dice of the first player. Also, the second line contains six positive integers that are written on the sides of the dice of the second player. The six integers given in a line add up to 21 and are separated by a single space.

Output
Your program is to write to standard output. Print exactly one line that contains an irreducible fraction representing the probability of the first player winning. A fraction should consist of a numerator displayed before a slash and a non-zero denominator displayed after the slash. There are no spaces either before or after the slash. Note that an irreducible fraction refers to a fraction in which the numerator and denominator are integers that have no other common divisors than 1.

The following shows sample input and output for three test cases.

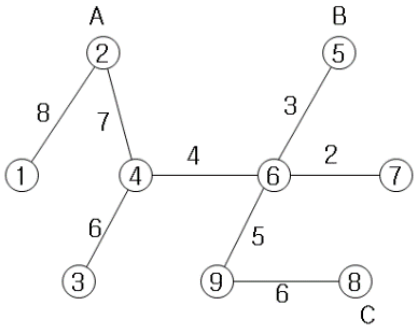
Sample Input 1	Output for the Sample Input 1
3 4 3 4 3 4 1 1 1 1 8 9	2/3

Sample Input 2	Output for the Sample Input 2
1 2 3 4 5 6 3 4 3 4 3 4	5/12

Sample Input 3	Output for the Sample Input 3
1 2 3 4 5 6 8 7 2 2 1 1	1/2

Problem C
Dessert Café
Time Limit: 1 Second

Kim, who wishes to start a business, is trying to open a dessert cafe he has been preparing after graduating from college. The road network in the town where Kim lives forms a tree structure, that is, a connected acyclic graph as shown in the figure below. There are n candidate sites for a dessert café in the town. In the figure below, a circle represents a candidate site for a dessert café, a line segment between two candidate sites represents a road, and the value labeled on a line segment represents the length of a road.



There are k apartment complexes in this town, so he wants his dessert café to be located as close as possible to an apartment complex. In above figure, there are three apartment complexes which are located to the candidate sites labeled by A, B, and C. Considering the competitiveness and profitability, he thinks that a candidate site satisfying the following condition is a good place.

Let $d(x, y)$ be the length of the shortest path on a road network between two candidate sites x and y. A candidate site p is *a good place* if there exists a candidate site z where an apartment complex is located such that $d(p, z) < d(q, z)$ for each candidate site q $(\neq p)$.

In above figure, candidate sites 2, 4, 5, 6, 8, and 9 are good places. More specifically, for example, candidate 6 is a good place because it is closer to apartment complex B than any other candidate sites except for candidate 5, and is closer to apartment complex A than candidate 5. That is, there exists apartment complex B on candidate 5 satisfying $d(6, 5) < d(q, 5)$ for $q \in \{1, 2, 3, 4, 7, 8, 9\}$, and there exists apartment complex A on candidate 2 satisfying $d(6, 2) < d(5, 2)$. Candidate 7 is not a good place because none of apartment complexes are closer than candidate 6.

Given the information on candidate sites and apartment complexes in the town, write a program to output the number of good places.

Input

Your program is to read from standard input. The input starts with a line containing two integers, n and k ($3 \leq n \leq 100{,}000$, $1 \leq k \leq n$), where n is the number of candidate sites and k is the number of apartment complexes. The candidate sites are numbered from 1 to n. In the following $n-1$ lines, each line contains three integers, i, j, and w ($1 \leq i, j \leq n$, $1 \leq w \leq 1{,}000$), where i and j are candidate sites and w is the length of the road between i and j. The last line contains k integers which represent the locations of apartment complexes in the town.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the number of good places.

The following shows sample input and output for two test cases.

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

Sample Input 2	Output for the Sample Input 2
4 4 1 2 1 1 3 1 1 4 1 2 4 1 3	4

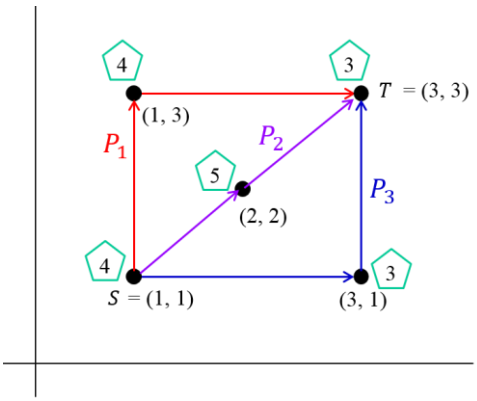
Problem D
Electric Vehicle
Time Limit: 2 Seconds

An electric vehicle(EV) is an alternative fuel automobile which uses one or more electric motors for propulsion, in place of the internal combustion engine such as diesel engine or gasoline one. The EV stores electricity in an energy storage device, such as a battery, and the electricity powers the vehicle's wheels via an electric motors. Due to its limited energy storage capacity, the EV must be regularly recharged by plugging into an electrical source.

There are n villages, referred to points on a plane. For arbitrary two villages, there is only one road between them. Thus, there are exactly $n \times (n-1)/2$ roads connecting villages each other. Let two villages A and B have coordinates (a, b) and (c, d), respectively. Then the length of the road between A and B is defined to be $d(A, B) = |a - c| + |b - d|$, which is called the distance between A and B. For convenience, it is assumed that the distance between A and B is equivalent to the amount of electricity needed for an EV to move between A and B. That is, while an EV consumes a unit electricity, say electricity 1, it can move a unit distance, say distance 1.

Each village has a charging station for an EV and the cost of charging may vary from village to village. Let $c(A)$ be the cost per electricity 1 when an EV is charged in village A. By the above assumption, an EV with zero charge at village A must be charged at the cost of at least $c(A)d(A, B)$ in A to arrive at village B.

Minsu should go from the village S to the village T by his EV. The EV is discharged, that is, with zero charge, at S before departure. Also the EV has a battery of maximum charge capacity W. That is, the EV cannot store the amount of electricity more than W. Thus if the EV is fully charged, then it can move as much as W. During the way from S to T, Minsu is allowed to make at most Δ stops for recharging. Also the charging at the starting village S is considered as one of the stops for recharging. Then Minsu should find the cheapest way possible to move from S to T.



For example, the above figure shows five villages in a plane with the costs of charging represented by the numbers in pentagons. Also, let $W = 3$ and $\Delta = 2$. Three ways from S to T, P_1, P_2, and P_3, are shown as red, purple, and blue, respectively. In P_1, the EV is charged by electricity 2 at S, and it is recharged by electricity 2 at $(1, 3)$. So, totally the cost of $2 \times 4 + 2 \times 4 = 16$ is incurred. For the case of P_2, the EV is fully charged at S, and so the electricity 1 remains in the battery of EV when arriving at $(2, 2)$. Then the EV is recharged at $(2, 2)$ only by electricity 1 to reach T. So, totally the cost of $3 \times 4 + 1 \times 5 = 17$ is incurred. Finally, in P_3,

the EV is charged by electricity 2 at S to move to (3, 1) and it is recharged by electricity 2 at (3, 1). So, totally the cost of $2 \times 4 + 2 \times 3 = 14$ is incurred. The cost 14 is minimum and P_3 is the cheapest way to move from S to T.

Given the coordinates of n villages including S and T, the charging cost at each village, the maximum charge capacity W of EV's battery, and the arbitrary positive integer Δ, write a program to find the cheapest way to move from S to T with at most Δ stops for recharging.

Input

Your program is to read from standard input. The input starts with an integer n ($2 \leq n \leq 1{,}000$), representing the number of villages. Each of the following n lines contains three integers, a, b, and c ($0 \leq a, b \leq 10^6$ and $1 \leq c \leq 10^4$), where (a, b) is the coordinate of a village and c is the charging cost at the village. Here, in the n lines, the first line contains the coordinate of the starting village S and the second line contains the coordinate of the destination village T. Note that the coordinates of villages are all distinct. The next line contains an integer W ($1 \leq W \leq 10^5$) that represents the maximum charge capacity of EV's battery. The last line contains a positive integer Δ ($1 \leq \Delta \leq 10$), which indicates the maximum number of times for recharging while the EV travels from S to T. The charging at the starting village S is considered as one of the stops for recharging.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the cost of the cheapest way for the EV to move from S to T with at most Δ stops for recharging. If such a way does not exist, then the line should contain `-1`.

The following shows sample input and output for three test cases.

Sample Input 1

```
4
0 0 1
3 0 3
1 0 3
2 0 3
4
2
```

Output for the Sample Input 1

```
3
```

Sample Input 2

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

Output for the Sample Input 2

```
14
```

Sample Input 3

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

Output for the Sample Input 3

```
-1
```

Problem E
Imprecise Computer
Time Limit: 1 Second

The Imprecise Computer (IC) is a computer with some structural issue that it can compare two integers correctly only when their difference is at least two. For example, IC can always correctly answer '4 is larger than 2', but it can answer either '2 is larger than 3' or '3 is larger than 2' (in this case, IC arbitrarily chooses one of them). For two integers x and y , we say 'x defeats y' when IC answers 'x is larger than y'.

Given a positive integer n, let $P_n = \{1, 2, \ldots, n\}$ be the set of positive integers from 1 to n. Then we run a double round-robin tournament on P_n using IC. The double-round-robin tournament is defined as follows:

1. The tournament is composed of two rounds (the 1st round and the 2nd round).
2. For each round, each element in P_n is compared to every other element in P_n.

Now for each element k in P_n, let $r_i(k)$ be the number of wins of k in the i-th round of the tournament. We also define the 'difference sequence' $D = d_1 d_2 \ \ldots \ d_n$ where for each $1 \le k \le n$, $d_k = |r_1(k) - r_2(k)|$.

The following shows an example when $n = 5$.

1st round	2nd round
2 defeats 1	3 defeats 1
3 defeats 1	4 defeats 1
4 defeats 1	5 defeats 1
5 defeats 1	1 defeats 2
3 defeats 2	4 defeats 2
4 defeats 2	5 defeats 2
5 defeats 2	2 defeats 3
5 defeats 3	4 defeats 3
3 defeats 4	5 defeats 3
4 defeats 5	5 defeats 4

In the example above, $r_1(1) = 0$, $r_1(2) = 1$, $r_1(3) = 3$, $r_1(4) = 3$, $r_1(5) = 3$, and $r_2(1) = 1$, $r_2(2) = 1$, $r_2(3) = 1$, $r_2(4) = 3$, $r_2(5) = 4$. Therefore, the difference sequence is $D = 1\ 0\ 2\ 0\ 1$ in this example.

Given a sequence of n nonnegative integers, write a program to decide whether the input sequence can be a difference sequence of P_n.

Input
Your program is to read from standard input. The input starts with a line containing an integer n, ($3 \le n \le 1{,}000{,}000$), where n is the size of P_n. In the following line, a sequence of n integers between 0 and n is given, where each element in the sequence is separated by a single space.

Output

Your program is to write to standard output. Print exactly one line. Print `YES` if the sequence can be the difference sequence of P_n, and print `NO` otherwise.

The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
5 1 0 2 0 1	YES

Sample Input 2	Output for the Sample Input 2
5 1 1 2 1 0	NO

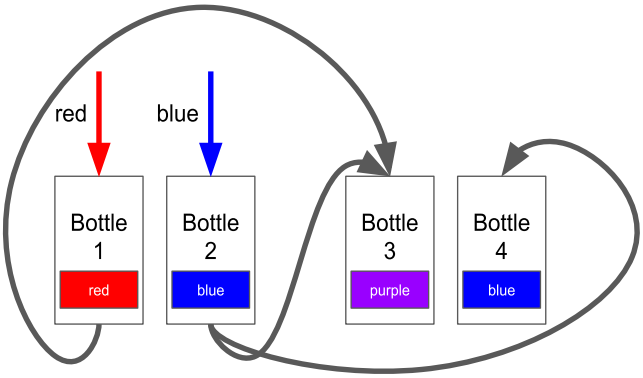
Problem F
Ink Mix
Time Limit: 2 Seconds

There are n ink bottles, numbered from 1 to n. The bottles are initially empty. The first m bottles are fed with inks of m different colors, and the n ink bottles may be fed from each other with hoses. The ink colors of the bottles may change as inks are continuously fed and mixed, but at the end of the day, the colors reach an equilibrium and do not change any longer. At an equilibrium, the following properties hold:

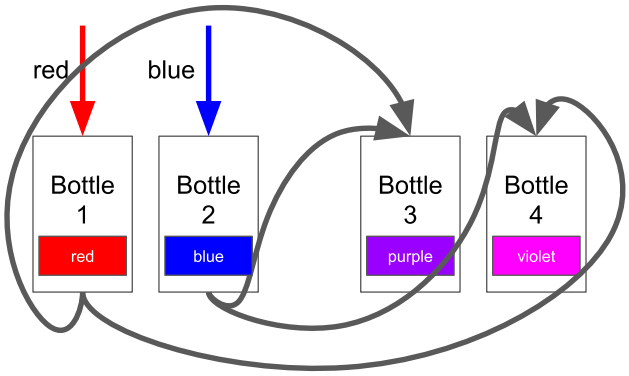
- A bottle, say A, contains ink if and only if the inks fed to the first m bottles may flow to A via possibly multiple hoses. Otherwise, A is empty.
- Each non-empty bottle contains ink of a single color.
- If a bottle, say B, is fed from multiple ink sources (either inks of m different colors fed to the first m bottles or inks of non-empty bottles fed to hoses), and not all the ink colors of the sources are the same, then B is called "mixer".
- A non-empty non-mixer bottle contains ink of the same color with those contained in its sources.
- A mixer bottle contains ink of a whole new color that is different from those of the inks fed to the first m bottles. Furthermore, different mixer bottles contain inks of different colors.

You want to know the minimum possible number of different ink colors in the bottles at an equilibrium.

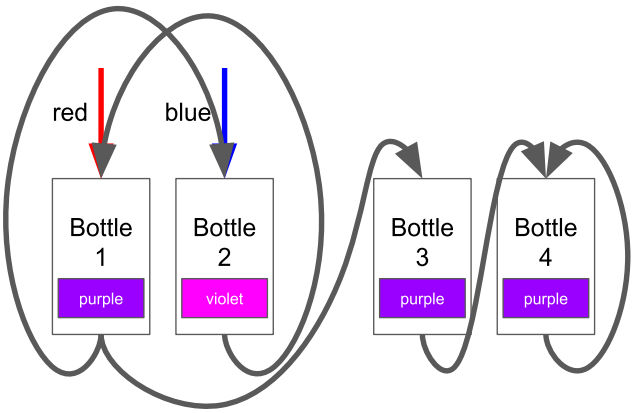
For example, the figure right shows four ink bottles at an equilibrium. The first two bottles are fed with red and blue ink, respectively. The first bottle's ink is fed to the third bottle, and the second bottle's ink is fed to the third and fourth bottles. The first and second bottles contain red and blue ink, respectively. The third bottle is a mixer because it is fed from two source bottles of inks with different colors, red and blue, so it contains the ink with a new color, say purple, which is the mixture of red and blue inks. The fourth bottle contains the blue ink. Overall, there are three different ink colors: red, blue, and purple, which is the minimum.



In the figure right, the first two bottles are again fed with red and blue ink, respectively. Their inks are fed to the third and fourth bottles. Blue and red inks into bottle 3 are mixed as a new color, say purple, as in the previous example. Even though the inks of the same blue and red are fed to bottle 4 like the case of bottle 3, the resulting color, say violet, is different from the color of bottle 3 because bottles 3 and 4 are different mixers. Overall, there are four different ink colors: red, blue, purple, and violet, which is the minimum.



The first m bottles may also feed their inks to each other. In the figure right, the first two bottles are fed with red and blue ink, respectively, but they are fed to each other. The first bottle's ink is fed to the third bottle, the third bottle's ink is fed to the fourth bottle, and the fourth bottle's ink is fed to itself. The first and the second bottles are mixers that create the inks with different colors: purple and violet. The ink color of the third bottle is the same with the ink color of the first bottle because only the first bottle supplies the ink to the third. The fourth bottle is fed with two inks but with the same color, so it is not a mixer and the resulting color is the same. Overall, there are two different ink colors: purple and violet, which is the minimum.



Given the configuration of bottles, write a program that prints the minimum possible number of different ink colors in the bottles at an equilibrium. Empty bottles may exist, but they are not counted.

Input

Your program is to read from standard input. The input starts with a line containing three integers, n, m, and k ($1 \le m \le n \le 100{,}000$, $1 \le k \le 500{,}000$), where n is the number of ink bottles, m is the number of bottles that are fed with unique ink, and k is the number of hoses. The ink bottles are numbered from 1 to n. In the following k lines, the i-th line contains two nonnegative integers, f_i and t_i, that means the f_i-th bottle's ink is fed to the t_i-th bottle by the i-th hose. Multiple hoses may connect the same pairs of bottles.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the minimum possible number of different ink colors in the bottles at an equilibrium.

The following shows sample input and output for four test cases.

Sample Input 1

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

Output for the Sample Input 1

```
3
```

Sample Input 2

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

Output for the Sample Input 2

```
4
```

Sample Input 3

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

Output for the Sample Input 3

```
2
```

Sample Input 4

```
4 2 1
3 4
```

Output for the Sample Input 4

```
2
```

Problem G
Mobile Robot
Time Limit: 1 Second

Mobile robots are nowadays commonly used in various industrial and research sites. You are in charge of controlling n mobile robots that explore a very long, narrow and straight cave, which can be seen just as a line. The mobile robots collect data from the environment nearby and have effective mobility with their caterpillar tracks. You can control the n mobile robots on your control desk by a wireless control system. The n mobile robots you are controlling are labelled with numbers 1 to n, and are identified by robot 1, robot 2, …, robot $n-1$, and robot n.

The mobile robots can also share their collected data to each other by a simple infrared communication protocol, while this robot-to-robot communication only works when the following very strict arrangement is completed for all the n mobile robots: the distance between robot i and robot $i+1$ should be *exactly* d for all $i = 1, 2, \ldots, n-1$, where d is a prescribed positive real number, and no two robots should be at the same location in the cave. The location of each mobile robot in the cave is represented by a real number x since the cave is very long, very narrow, and very straight, so can be considered a line which stretches limitlessly in both directions. The distance between two mobile robots is thus calculated by the difference of their locations.

From the current locations of the mobile robots, they now need to share data to each other, and you are going to move them for the robot-to-robot communication. Since the robots are slow and simultaneously move at the same speed along the cave, you want to minimize the maximum distance each robot should travel to waste as little time as possible. During travelling, any two robots are assumed to safely pass by each other at the moment when both are at a common location in the cave. Note hence that currently two or more robots may be at a common location in the cave.

Given the current locations of the n mobile robots, write a program that computes their new locations for the robot-to-robot communication that minimizes the maximum distance each of the n robots travels and outputs the minimized maximum distance the robots should travel.

Input
Your program is to read from standard input. The input consists of exactly two lines. The first line consists of two integers, n and d ($2 \le n \le 1{,}000{,}000$ and $1 \le d \le 10^{10}$), where n denotes the number of mobile robots you are controlling and d is the distance that the robots should keep for the robot-to-robot communication. Each mobile robot is identified by a label from 1 to n. The second line consists of n integers, each of which ranges from -10^{16} and 10^{16}, representing the current locations of robot 1, robot 2, …, and robot n in this order.

Output
Your program is to write to standard output. Print exactly one line consisting of a real number, rounded to the first decimal place, that represents the minimum possible value of the maximum distance the mobile robots should travel for the robot-to-robot communication from the given current locations.

The following shows sample input and output for four test cases.

Sample Input 1	**Output for the Sample Input 1**
5 1 1 3 5 7 9	2.0

Sample Input 2	**Output for the Sample Input 2**
5 1 -10 -1 0 1 2	4.0

Sample Input 3	**Output for the Sample Input 3**
5 1 1 3 5 9 7	2.5

Sample Input 4	**Output for the Sample Input 4**
5 1 1 1 1 1 1	2.0

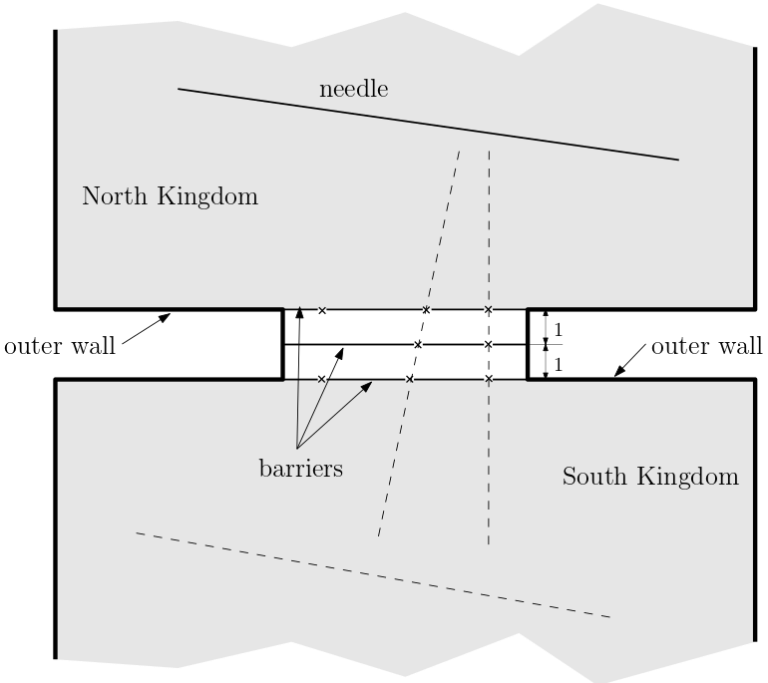
Problem H
Needle
Time Limit: 1 Second

The “needle” is a legendary assassin who lives in the North Kingdom. As you know, the needle is very thin and long. More than anything, it is deadly sharp. The king of the North Kingdom is obsessed with the idea that the needle might kill him by stabbing countless times. The king issued an emergency order to arrest the needle. So, the needle decided to escape to the South Kingdom.

As shown in the figure below, the border between two kingdoms consists of three horizontal barriers (line segments), each of which has one or more infinitesimally small holes inside. (The holes are marked as x in the figure.) Three barriers have the same length and are aligned vertically as in the figure. The upper barrier is one unit above the middle barrier, which is one unit above the lower barrier. Two kingdoms are surrounded by impenetrable outer wall. Each kingdom also has a very large territory so that the needle can move (translate or rotate) freely inside the kingdom. The needle is at least twice as long as the barriers. The needle is rigid, i.e., not bendable, and has zero-thickness, so it can pass the holes freely, but cannot drill any other part of the barriers than the holes.

The only way from the Northern Kingdom to the Southern Kingdom is through three holes, one from each of the three barriers, at the same time. In other words, the needle can pass the border only through three holes, exactly one from each barrier, which are aligned on a line. The border in the figure has two possible escape passages from the north to the south.

For this pity assassin, write a program to tell how many possible escape passages from the North Kingdom to the South Kingdom are available.



Input

Your program is to read from standard input. The input consists of six lines. The first line contains a positive integer n_u representing the number of holes of the upper barrier. The second line contains n_u integers separated by a space that represent the x-coordinates of the holes. The third and fourth lines are for the middle barrier, each containing n_m, the number of holes of the middle barrier, and n_m x-coodinates of the holes. The fifth and sixth lines are for the lower barrier, each containing n_l, the number of holes of the lower barrier, and n_l x-coodinates of the holes. $1 \leq n_u, n_m, n_l \leq 50{,}000$ and all x-coordinates of the holes are integers between $-30{,}000$ and $30{,}000$. Holes of each barrier have all distinct x-coordinates.

Output

Your program is to write to standard output. Print exactly one line. The line should contain a nonnegative integer representing the number of all possible passages from the north to the south.

The following shows sample input and output for three test cases. The second sample corresponds to the figure above.

Sample Input 1

```
1
1
1
2
1
1
```

Output for the Sample Input 1

```
0
```

Sample Input 2

```
3
4 -3 2
2
4 1
3
-3 4 0
```

Output for the Sample Input 2

```
2
```

Sample Input 3

```
3
-1 1 0
3
0 1 -1
3
0 -1 1
```

Output for the Sample Input 3

```
5
```

Problem I
Stock Analysis

Time Limit: 2 Seconds

SY Company wants to analyze a stock. The fluctuation value, which is the difference in stock prices for two consecutive days, is the most frequently used data for the time-series analysis of the stock. It is important to utilize the largest sum of the contiguous fluctuation values. However, using the largest contiguous sum as a key indicator could be risky. As an alternative, the company utilizes the largest contiguous sum that is not greater than a predetermined value U in a specified period $[S, E]$ from S to E. The company wants to process such queries as fast as possible, where a query is defined as a predetermined value U and a period $[S, E]$.

Given a collection of n recent fluctuation values for some stock and m queries $\{(S_1, E_1, U_1), \ldots, (S_m, E_m, U_m)\}$, write a program to find the largest sum of contiguous fluctuation values that is less than or equal to U_i in the period $[S_i, E_i]$for each query (S_i, E_i, U_i).

Input

Your program is to read from standard input. The input starts with a line containing two integers, n and m, representing the number of fluctuation values and the number of queries respectively, where $1 \le n \le 2{,}000$ and $1 \le m \le 200{,}000$. The next line contains n integers representing n fluctuation values, which are numbered in chronological order from 1 to n. Each of the following m lines represents a query that consists of three integers, S_i, E_i, and U_i, where $[S_i, E_i]$ is the period from S_i to E_i over which the fluctuation values should be considered and U_i is the value that the contiguous sum should not exceed. Note that all fluctuation values are between -10^9 and 10^9, $1 \le S_i \le E_i \le n$ and $-10^{14} \le U_i \le 10^{14}$ for $i = 1, \ldots, m$.

Output

Your program is to write to standard output. Print exactly m lines. The i-th line should contain the largest sum of contiguous fluctuation values that does not exceed U_i and in the period $[S_i, E_i]$ for the i-th query. Note that every contiguous sum is the sum of one or more consecutive fluctuation values. When it is impossible to find such the sum, the program should print `NONE`.

The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
5 3 1 -2 -3 5 4 1 3 -2 1 5 8 1 5 3	-2 6 2

Sample Input 2	Output for the Sample Input 2
6 4 3 8 -3 2 5 2 1 6 17 1 6 16 2 5 4 2 5 -4	17 15 4 NONE

Problem J
Switches
Time Limit: 1 Second

Alice, who loves to travel, got to stay at a very interesting hotel. Every guest who wishes to stay at the hotel should solve an interesting quiz about the lights installed in the room where they will be staying. The front desk of the hotel gives each guest an information on the lights installed in the room where they are staying and the switches connected to the lights. According to the information, one or more lights are connected to one switch, and each light is connected to one or more switches.

There are N lights and N switches in the room where Alice will be staying. What is interesting is that when Alice makes a switch on, not only one light is turned on, but several lights are turned on at the same time. Also, making another switch on while one or more switches are already on may turn off some lights that are already turned on. Fortunately, when all the switches are off, all the lights are off as well.

Figure J.1 is an example of the information Alice received from the front desk of the hotel, where $N = 5$.

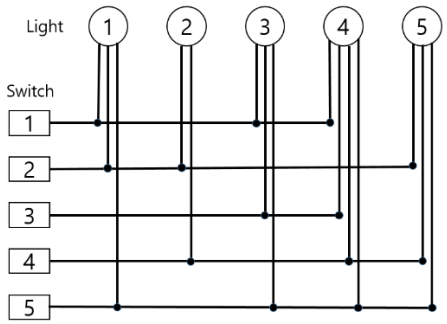


Figure J.1

To figure out how each light is affected by the switches, Alice experiments as follows. First, each light is numbered and the switches are also numbered so that they could be identified. She makes all the switches off initially. Then, she checks which lights are turned on by making only the switch #1 on. After that she makes switch #1 off and switch #2 on to check which lights are turned on. Again, she makes switch #2 off and switch #3 on, and so on. She repeats this to check which lights are turned on by each switch.

Then, she makes two or more switches on to find what rules exist. As a result, it is found that each light is toggled by the switches that are connected to it. This rule can be stated as follows:

- A light is turned on (off) when the number of switches, which are connected to it and are on state, is odd (even).

For example, consider the connection information shown in Figure J.1 and focus on how light #1 operates. The light #1 turns on by making each of the switches #1, #2 and #5 on. If she makes switch #1 on while all the other switches (i.e., switch #2 and #5) are off, light #1 turns on. And, if she additionally makes switch #2 on, light #1 turns off. If she additionally makes switch #5 on (that is, all the three switches are on), light #1 turns on again. While operating such switches, the state of other lights may also change.

Alice wonders if for each light, it is possible to turn it on and the rest of the lights off by operating some switches.

Given the connection information between switches and lights, write a program to help Alice. In other words, your program should tell whether for each light, it is possible to turn it on and the rest of the lights off by operating some switches.

Input

Your program is to read from standard input. An integer N ($3 \leq N \leq 500$) is given in the first line. Each of the following N line contains N integers of `0`'s and `1`'s separated by a single space. The numbers in the i-th ($1 \leq i \leq N$) line represent which lights are connected to the i-th switch. If the k-th ($1 \leq k \leq N$) value in the i-th line is `1`, it means the k-th light is connected to the i-th switch; `0` means it is not connected.

Output

Your program is to write to standard output. If for every light, it is possible to turn it on and the rest of the lights off by operating some switches, print the switch numbers in increasing order that should be on. Otherwise, print `-1` as shown in the following samples. If your output is not `-1`, the switch numbers in the k-th line should be those that make the k-th light on. If there are more than one correct answers print any of them.

The following shows sample input and output for three test cases.

Sample Input 1

```
4
1 0 1 0
0 1 0 1
0 1 1 1
1 1 0 0
```

Output for the Sample Input 1

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

Sample Input 2

```
4
1 1 1 0
0 1 0 1
1 0 0 1
0 1 1 1
```

Output for the Sample Input 2

```
-1
```

Sample Input 3

```
5
1 0 1 1 0
1 1 0 0 1
0 0 1 1 0
0 1 0 1 1
1 0 1 1 1
```

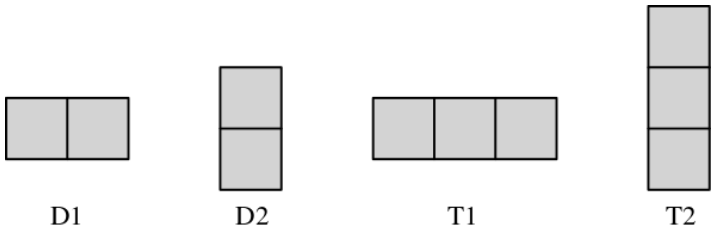
Output for the Sample Input 3

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

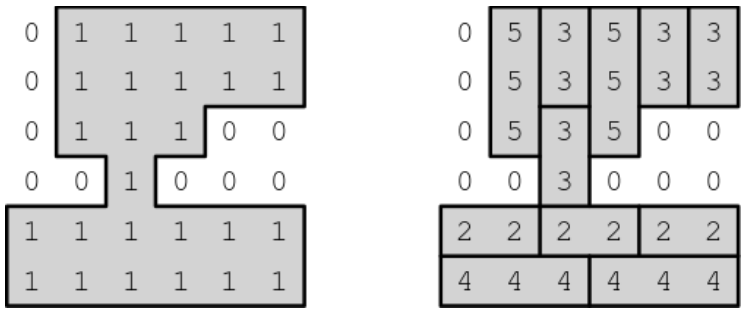
Problem K
Tiling Polyomino
Time Limit: 1.5 Seconds

A polyomino is a plane geometric figure formed by joining one or more unit squares edge to edge. The figure below shows two examples of polyominos. Each of the squares contained in a polyomino is called a cell, and two cells sharing an edge are called neighbors of each other. Note that the number of neighbors of a cell can be 0, 1, 2, 3 and 4. A polyomino P is called connected if every pair of cells (a, b) in P has a path connecting neighboring cells from a to b, and P is called simply connected if P is connected and does not contain any "hole". In the figure below, the left one is simply connected but the right one is not. We will deal with a simply connected polyomino P such that every cell contained in P has two or more neighbors.

A tiling of a polyomino P is a tessellation (covering using geometric shapes with no overlaps and no gaps) of P by translated copies of D1, D2, T1, and T2, where D1 (resp. D2) is a polyomino formed by joining two unit squares horizontally (resp. vertically), and T1 (resp. T2) is a polyomino formed by joining three unit squares horizontally (resp. vertically). The figure below shows D1, D2, T1, and T2. According to the shape of P, a tiling of P may or may not exist.



To represent a polyomino P, we assume that P is contained in an $n \times n$ unit square grid. We label each unit square s in the grid as `1` if s is a cell of P, or `0` otherwise. Then, the unit square grid containing P can be represented by an $n \times n$ matrix of `0`'s and `1`'s. A tiling of P can also be represented by an $n \times n$ matrix of integers as follows. If a cell of P is covered by a copy of D1 or D2, then we label the cell as `2` or `3`, respectively. If a cell of P is covered by T1 or T2, then we label the cell as `4` or `5`, respectively. The figure below shows an example of tiling and its representation.



Given a simply connected polyomino P such that every cell contained in P has two or more neighbors represented by an $n \times n$ matrix of `0`'s and `1`'s, write a program that outputs a tiling of P, if it exists.

Input

Your program is to read from standard input. The input starts with a line containing an integer $2 \leq n \leq 1{,}000$, where n is the number of rows and columns of the unit square grid containing a polyomino P. Each of the following n lines contains n many `0`'s and `1`'s, and `1` denotes that the square is a cell of P. The polyomino P is simply connected and every cell contained in P has two or more neighbors.

Output

Your program is to write to standard output. If there is a tiling of P, print the tiling of P using n lines. Each of the n lines contains n integers from `0` to `5`. A `0` represents that the square is not a cell of P. A `2` represents that the square is a cell of P, and is covered by D1. Similarly, a `3`, `4`, or `5` represents that the square is a cell of P, and is covered by D2, T1, or T2, respectively. If there is no possible tiling of P, then print `-1`.

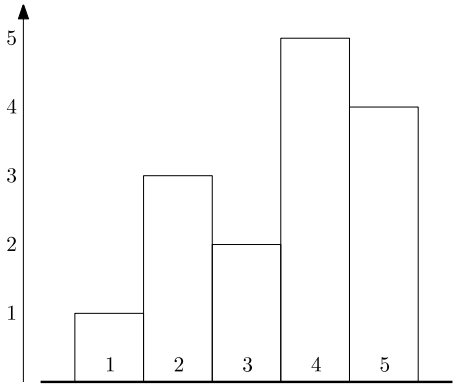
The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
3 011 111 111	022 322 322

Sample Input 2	Output for the Sample Input 2
6 011111 011111 011100 001000 111111 111111	053533 053533 053500 003000 222222 444444

Problem L
Two Buildings
Time Limit: 1 Second

There are n buildings along a horizontal street. The buildings are next to each other along the street, and the i-th building from left to right has width 1 and height h_i. Among the n buildings, we are to find two buildings, say the i-th building and j-th building with $i < j$, such that $(h_i + h_j) * (j - i)$ is maximized.



For example, the right figure shows 5 buildings, with heights 1, 3, 2, 5, 4, from left to right. If we choose the first 2 buildings, then we get $(1 + 3) * (2 - 1) = 4$. If we choose the first and fifth buildings, then we $(1 + 4) * (5 - 1) = 20$. The maximum value is achieved by the second and fifth buildings with heights 3 and 4, respectively: $(3 + 4) * (5 - 2) = 21$.

Write a program that, given a sequence of building heights, prints $\max_{1 \le i < j \le n} (h_i + h_j) * (j - i)$.

Input
Your program is to read from standard input. The input starts with a line containing an integer n ($2 \le n \le 1{,}000{,}000$), where n is the number of buildings. The buildings are numbered 1 to n from left to right. The second line contains the heights of n buildings separated by a space such that the i-th number is the height h_i of the i-th building ($1 \le h_i \le 1{,}000{,}000$).

Output
Your program is to write to standard output. Print exactly one line. The line should contain $\max_{1 \le i < j \le n} (h_i + h_j) * (j - i)$.

The following shows sample input and output for two test cases.

Sample Input 1	Output for the Sample Input 1
5 1 3 2 5 4	21

Sample Input 2	Output for the Sample Input 2
5 8 3 6 3 1	36