Problem A. On-Call

- Time Limit: 2 sec

Fox Jiro works for a company that runs a social networking site for programmers. His duty is to recover the service as soon as possible when it becomes unavailable.

His predecessor has developed a monitoring system that detects service unavailability based on the frequency of server error responses such as "502 Bad Gateway" and "503 Service Unavailable". The system records the frequency of server error responses per minute. If the frequency remains high for a while, an alert is triggered and Fox Jiro receives a phone call. After that, if the frequency of server error responses remains low for a while, the alert will be resolved.

In more detail, alerts are triggered and resolved as follows.

- Initially, no alert is triggered.
- An alert is triggered when no alert is triggered and the frequency of server error response for each of the last D consecutive minutes is greater than or equal to U.
- The alert is resolved when the frequency of server error responses for each of the last D consecutive minutes is less than or equal to L.

Fox Jiro can receive compensation from the company according to the number of times he received phone calls due to the alerts. Last month, he received phone calls so many times that he forgot how many times the alerts were triggered. Your task is counting the number of alerts triggered based on the record of server error responses.

Input

The input consists of a single test case of the following format.

$N\ U\ L\ D$
$x_1\ x_2\ \ldots\ x_N$

The first line consists of four integers N, U, L, and D. N represents the length of the record of the frequency of server error responses. U, L, and D are the settings that determine the behavior of the monitoring system as explained above. Here, N and D satisfy $1 \leq N, D \leq 2 \times 10^5$. U and L satisfy $0 \leq L < U \leq 2 \times 10^5$.

The second line consists of N integers. The i-th integer represents the frequency of server error responses that occurred in the i-th minute. Each of the integers in the second line is between 0 and 2×10^5 inclusive.

Output

Print the number of times the alerts were triggered in one line.

Examples

Input	Output
15 1 0 3 1 1 1 1 0 0 0 1 1 0 0 0 1 1 1	2
24 3 1 3 1 1 3 2 3 1 1 1 3 3 3 3 1 3 3 3 1 1 1 2 1 3 3 3	2

B: Warp Points

- Time Limit: 2 sec

You are constructing an interstellar transportation network. There are N stars in your area, and your task is to make it possible to go to any star from any star. Currently, all of these stars are isolated, and we cannot go anywhere from each star.

As an interstellar transportation network designer, you can set warp points. Stars are numbered from 1 to N, and a warp point enables coming and going between consecutively numbered stars. The construction cost for a warp point depends on the "potentials" for stars and the warp point. The potential for i-th star is given as an integer, and you can set the potential for the warp point to any integer. The cost is calculated as the summation of the absolute difference between the warp point's potential and each contained star's potential.

You can construct any number of warp points. Your task is to calculate the minimum total cost to enable coming and going between any pair of stars using some warp points.

For the sample input 1, you should plan to construct one warp point with potential 2.

For the sample input 2, three warp points are needed to minimize the total cost. The first warp point connects from the first to the fourth star. The second warp point connects from the fourth to the sixth stars. The third warp point connects from the sixth to tenth stars.

Input

The input consists of a single test case of the following format.

N
$a_1 \ \ldots \ a_N$

N represents the number of stars ($1 \leq N \leq 3{,}000$). Numbers a_1 through a_N are the potentials of the stars. It is guaranteed that these potentials are integers in the range between $-1{,}000{,}000{,}000$ and $1{,}000{,}000{,}000$.

Output

Output the minimum total cost to connect all stars.

Examples

Input	Output
3 2 5 2	3
10 1 2 3 2 1 10 9 8 9 10	14

C: Rebound Sequences

- Time Limit: 2 sec

An integer sequence a is **rebound sequence** if there are three integers i, j, k ($1 \leq i < j < k \leq N$) satisfying $a_i > a_k > a_j$. You are given an integer sequence s. Your task is to count the number of rebound sequences that can be obtained by permuting the elements of s.

Input

The input consists of a single test case in the format below.

N
s_1 s_2 ... s_N

The first line contains a single integer N ($1 \leq N \leq 200$). The second line contains N integers s_i ($1 \leq s_i \leq N$), which is the i-th element of s.

Output

Output the number of rebound sequences that can be obtained by permuting the elements of s modulo $10^9 + 7$.

Examples

Input	Output
4 1 2 3 4	10
12 1 2 3 4 5 6 7 8 9 10 11 12	478793588
5 3 1 4 1 5	32
20 1 1 1 2 3 4 4 7 7 8 11 12 13 14 15 16 17 17 19 20	959127228

D: Surround the Castle

- Time Limit: 2 sec

You just started a simulation game of the Age of Civil Wars! In this game, there is an infinite two-dimensional grid. Each cell in this grid is either a *plain cell*, a *castle cell* or a *moat cell*.

Your *territory* is a rectangle of cells that consists of R rows and C columns. Your territory now consists of only one castle cell and the other $RC - 1$ plain cells. You can change each plain cell to a moat cell by paying some costs depending on the cell.

To protect your castle cell from foreign enemies, you decided to *surround* your castle cell with moat cells. More precisely, your castle cell is considered surrounded by moat cells if any cell outside your territory can't be reached from your castle cell by repeatedly moving to a cell that is not your moat cell and is horizontally, vertically, or diagonally adjacent to the current cell.

Figures D-1 to D-3 show examples of your territories whose borders are depicted as bold rectangles. In Figure D-1 the moat cells, which are depicted as blue squares, surround the castle cell. On the other hand, in Figures D-2 and D-3 the moat cells don't surround the castle cells.

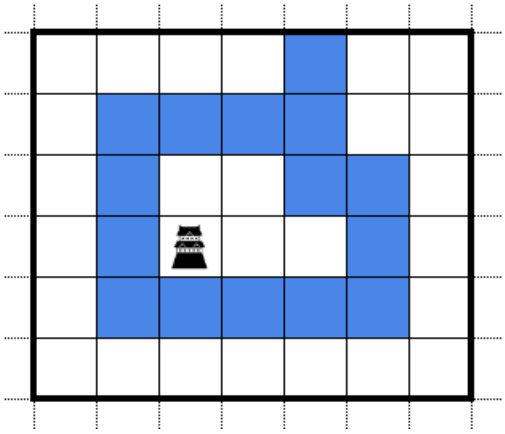

Figure D-1. An example of a castle cell surrounded by moat cells

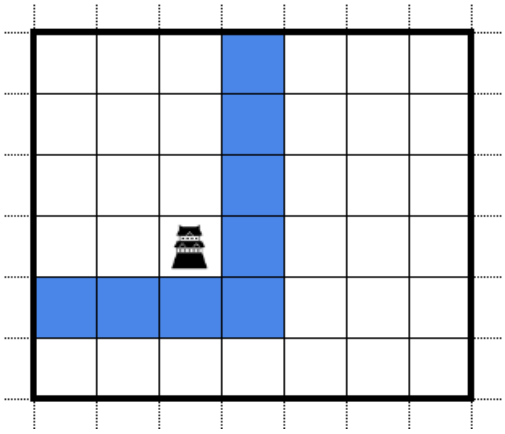

Figure D-2. An example of a castle cell not surrounded by moat cells

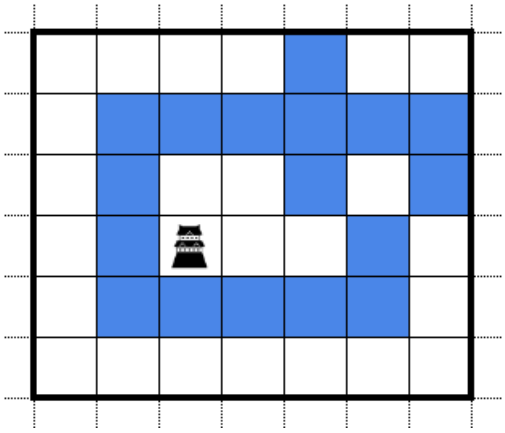

Figure D-3. Another example of a castle cell not surrounded by moat cells

You are given the location of your castle cell and the cost to change each of the other plain cells to a moat cell. Find the minimum total cost to surround your castle cell with moat cells.

Input

The input consists of a single test case of the following format.

R C
a_{11} ... a_{1C}
$\vdots$
a_{R1} ... a_{RC}

R and C represent the number of rows and columns of your territory ($3 \leq R \leq 20$, $3 \leq C \leq 10^4$). Each a_{ij} ($a_{ij} = -1$ or $1 \leq a_{ij} \leq 10^9$) gives the information of the cell of the i-th row and the j-th column of your territory. If it is -1, then the cell is your castle cell. Otherwise, the cell is initially a plain cell and a_{ij} equals the cost to change it to a moat cell.

It is guaranteed that there is exactly one castle cell in the input. In addition, your castle cell never locates at the first row, the R-th row, the first column nor the C-th column. In other words, it is guaranteed that you can surround your castle cell with moat cells.

Output

Output the minimum total cost to surround your castle cell with moat cells.

Examples

Input	Output
5 5 1 1 1 1 1 1 9 9 9 1 1 9 -1 9 1 1 9 9 9 1 1 1 1 1 1	16
6 8 13 15 25 28 88 90 74 87 20 85 20 18 14 26 18 56 40 54 11 90 91 13 41 66 30 82 51 -1 38 22 86 62 27 55 13 51 42 64 77 36 12 90 43 25 11 60 60 72	319

E: Buttons

- Time Limit: 2 sec

There is an $H \times W$ grid, with one button in each cell. Initially, all buttons are off. You will push them and turn them on.

Your task is to find a "good" timing of pressing the buttons. Let t_{ij} be the timing to push the button of row i and column j. The timing is said to be "good" if and only if the following conditions are satisfied.

- t_{ij} is an integer between 0 and 10^9 for all i and j.
- $t_{kl} + a_{ij} \leq t_{ij} \leq t_{kl} + b_{ij}$ for every cell kl which is a horizontal or vertical neighbor of the cell ij, i.e., $|i - k| + |j - l| = 1$.

Write a program to output a "good" timing for the given a and b. If there are several possible timings, you can output any of them. If there is no "good" timing, you should output -1.

Input

The input consists of a single test case of the following format.

$$
\begin{aligned}
&H\ W \\
&a_{11}\ \ldots\ a_{1W} \\
&\vdots \\
&a_{H1}\ \ldots\ a_{HW} \\
&b_{11}\ \ldots\ b_{1W} \\
&\vdots \\
&b_{H1}\ \ldots\ b_{HW}
\end{aligned}
$$

H and W represent the height and width of the given grid ($2 \leq H, W \leq 50$). a_{ij} and b_{ij} represent the range of time differences for the button of row i and column j ($-100{,}000 \leq a_{ij} \leq b_{ij} \leq 100{,}000$).

Output

If there is a "good" timing, output it in the following format.

$$
\begin{aligned}
&T_{11}\ \ldots\ T_{1W} \\
&\vdots \\
&T_{H1}\ \ldots\ T_{HW}
\end{aligned}
$$

T_{ij} is an integer representing the timing to push the button of row i and column j. The timings should satisfy the conditions defined in the problem statement. If there are multiple correct answers, you can print any of them.

If there is no "good" timing, you should output -1 instead.

Examples

Input	Output
3 3 -2 1 -2 1 -2 1 -2 1 -2 -1 2 -1 2 -1 2 -1 2 -1	0 1 0 1 0 2 0 2 0
2 2 1 1 1 1 1 1 1 1	-1

F: Mountain View

- Time Limit: 2 sec

Mr. Mt. is eager to take photos of mountains all over the world. Today, Mr. Mt. visits one of the most interesting mountain ranges, the Alps of Curved Mountains. The shape of all the mountains of the Alps of Curved Mountains is, surprisingly, hemisphere.

Mr. Mt. just decided the place from where he will take photos of the mountains. The area covering the mountains is flat, horizontal ground, and the place he chose is far enough from the mountains. Hence the mountains look like two-dimensional semicircles in a row, where all the straight segments of the semicircles are aligned with a straight line. Some of them may be overlapped or completely covered by other semicircles.

Mr. Mt. is waiting for the sunset. After sunset, the mountains should be in shadow and cannot be distinguished. In that situation, Mr. Mt. is interested in how the photos will look like. You, his camera assistant, are also a great programmer. Given the coordinates and radiuses of all the semicircles, he asks you to write a program to calculate the heights of the 2D shadow at the positions he wants to know.

Figure F-1 illustrates the mountains of Sample 2. The first, second, third mountains are overlapped, and the fifth mountain is covered by the fourth mountain. Figure F-2 shows the mountains in shadow. For example, at position 82, there are the fourth and fifth mountains, and the shadow's height is the highest of them, the height of the fourth mountain.

Figure F-1. Mountains of Sample 2

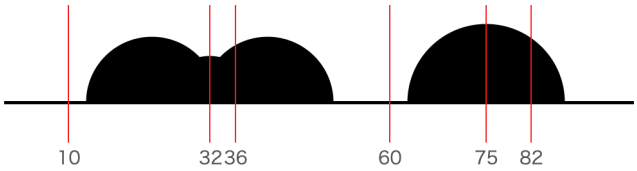

Figure F-2. Mountains in shadow and positions of Sample 2

Input

The input consists of a single test case in the format below.

$$
\begin{aligned}
&N \\
&c_1\ r_1 \\
&\vdots \\
&c_N\ r_N \\
&Q \\
&x_1 \\
&\vdots \\
&x_Q
\end{aligned}
$$

The first line contains a single integer N ($1 \leq N \leq 10^5$), where N is the number of mountains of the Alps of Curved Mountains. The i-th of the following N lines contains two integers c_i ($1 \leq c_i \leq 10^8$) and r_i ($1 \leq r_i \leq 10^8$). c_i is the position of the center of the i-th semicircle in a straight line, where "center" means the center of the circle made by reflecting the semicircle on the straight segment of the semicircle. r_i is the radius of the i-th semicircle. The following line contains a single integer Q ($1 \leq Q \leq 10^5$), where Q is the number of the positions on a straight line Mr. Mt. is interested in. The j-th of the following Q lines contains a single integer x_j ($1 \leq x_j \leq 10^8$), where x_j is the j-th position on a straight line Mr. Mt. is interested in.

Output

Output Q lines, the j-th of which is the height of the shadow of the mountains at the j-th position Mr. Mt. is interested in. Absolute or relative errors less than 10^{-7} are permissible.

Examples

Input	Output
3 1 10 4 10 10 5 6 4 2 8 12 14 18	10.0000000000 9.9498743711 9.1651513899 6.0000000000 3.0000000000 0.0000000000
5 23 10 32 7 41 10 75 12 79 6 6 10 32 36 60 75 82	0.0000000000 7.0000000000 8.6602540378 0.0000000000 12.0000000000 9.7467943448
9 3 8 21 10 12 9 15 3 1 11 19 4 9 6 17 1 22 7 7 19 3 16 8 9 31 1	9.7979589711 10.8166538264 8.6602540378 8.4852813742 8.4852813742 0.0000000000 11.0000000000

G: Yet Another Expression Mining

- Time Limit: 2 sec

You are given an integer A and a string S which consists of the following 10 characters: an addition operator `'+'`, a digit `'1'`, `'2'`, ... , `'9'`.

Count the number of subsequences of S which satisfy the following conditions.

- The subsequence is not empty.
- The first letter of the subsequence is not `'+'`.
- The last letter of the subsequence is not `'+'`.
- `'+'` does not appear consecutively in the subsequence.
- Reading the subsequence as a mathematical expression, its evaluation result is A.

Note that a subsequence is a sequence that can be obtained from the given sequence by removing zero or more elements without changing the order of the remaining elements.

Two subsequences are considered different if the set of removed elements' indices are different.

Input

The input consists of a single test case in the format below.

S
A

The first line contains a single string S ($1 \leq |S| \leq 36$). Each character of S is either `'+'`, or a digit between `'1'` and `'9'`. The second line contains a single integer A ($1 \leq A \leq 10^{18}$).

Output

Output the number of subsequences which satisfy the given conditions in a single line.

Examples

Input	Output
1+2+3+4+5+6 18	9
+391+49++21+9934 43	47
1111111111111111 1111111111	8008

H: Aggressive Traveller

- Time Limit: 2 sec

You are very eager to travel to many countries. There are N countries in the world. M distinct pairs of the countries have a travel route between the two countries. All the M routes are one-way: if there is a route from A to B, you can move from country A to country B, but not from B to A unless there is another route from B to A. Now you are in country S and set the goal of your travel to country T. From S to T, you want to visit the countries as many times as possible.

However, there are some restrictions. There are K countries which have strict security checks before you enter these countries. The flow of the security check of the i-th restricted country c_i is as follows:

1. You show your passport to an officer.
2. The officer checks your passport. If at least one of the following two conditions is satisfied, your entrance is rejected:
    1. Your passport has two or more stamps of the same country.
    2. Your passport has more than r_i stamps.
3. Otherwise your entrance is accepted. The officer stamps your passport with a stamp of country c_i.
4. You enter the country c_i.

Note that in $N - K$ countries other than K restricted countries, passport checking is skipped so you can freely enter these countries but you must get a stamp of a country you enter. Also notice that you may be able to enter c_i at most twice, because you get a stamp **after** passport checking. Initially your passport has only a single stamp of country S.

As an aggressive traveller, you want to maximize the number of stamps on your passport. You are going to start your travel from country S and eventually finish the travel at country T. Write a program that outputs the maximum number of stamps you can get on your passport when you reach T. This number includes the stamps of countries S and T. If you cannot reach T from S, output `'UNREACHABLE'` instead. Also, if you can indefinitely repeat to visit countries and then eventually reach T, output `'INFINITY'` instead. Note that you don't have to stop your travel when reaching T and can visit T multiple times.

Input

The input consists of a single test case in the format below.

$N\ M\ K\ S\ T$
$u_1\ v_1$
$\vdots$
$u_M\ v_M$
$c_1\ r_1$
$\vdots$
$c_K\ r_K$

The first line contains five integers N ($3 \leq N \leq 1{,}000$), M ($2 \leq M \leq 10{,}000$), K ($1 \leq K \leq N$), S ($1 \leq S \leq N$), T ($1 \leq T \leq N$). N is the number of countries. M is the number of routes between pairs of countries. K is the number of restricted countries. S and T are the countries you start your travel from and want to reach, respectively. The i-th of following M lines is the information of the i-th route: you can move from countries u_i to country v_i ($1 \leq u_i, v_i \leq N$). The j-th of further following K lines is the information of the j-th restricted country: country c_j ($1 \leq c_j \leq N$) has the limit r_j ($1 \leq r_j \leq 5$) of the number of stamps on your passport.

You can assume:

- $S \neq T$,
- no self-loop, i.e. $u_i \neq v_i$ for $1 \leq i \leq M$,
- no duplicate routes, i.e. $u_i \neq u_j$ and/or $v_i \neq v_j$ for $1 \leq i < j \leq M$,
- no duplicate restricted countries, i.e. $c_i \neq c_j$ for $1 \leq i < j \leq K$, and
- countries S and T are not restricted, i.e. $c_j \neq S$ and $c_j \neq T$ for $1 \leq j \leq K$.

Output

Output the maximum number of stamps you can get on your passport during your travel from S to T. If you cannot reach T from S, output `'UNREACHABLE'`. If you can get an infinite number of stamps on your passport, output `'INFINITY'`.

Examples

Input	Output
4 5 1 1 4 1 2 1 3 2 1 2 3 3 4 3 2	4

3 3 1 1 2 1 2 1 3 3 1 3 5	4
4 2 2 1 4 1 2 3 4 2 5 3 5	UNREACHABLE
4 4 1 1 4 1 2 2 3 3 2 3 4 2 3	6
4 4 1 2 3 2 1 1 3 3 4 4 3 1 1	INFINITY
4 4 1 1 3 1 2 2 3 3 4 4 3 4 5	5
6 7 1 1 6 1 2 2 3 3 1 3 4 4 5 5 6 6 4 3 5	INFINITY
6 7 1 1 6 1 2 2 3 3 1 3 4 4 5 5 6 6 4 4 5	6
7 6 5 1 7 1 2 2 3 3 4 4 5 5 6 6 7 2 1 3 2 4 3 5 4 6 4	UNREACHABLE
4 4 1 1 4 1 2 2 3 3 1 2 4 3 2	6

I: Irreversible Reactions

- Time Limit: 2 sec

Life is ephemeral. What is done sometimes cannot be irreversible. In terms of this, chemical reaction networks are similar to life.

For brevity, we model a chemical reaction network as follows. There are N states in a chemical reaction network. The reaction starts with the state S. M distinct pairs (u_i, v_i) of states have a relation such that u_i can be transited to v_i. For each unit of time, a state is transited to other states if there is at least one state that can be transited from the state. If there are no such states, the state does not change. If there are two or more states to which the current state can be transited, the next state is selected at uniformly random from these states.

In an experiment of the chemical reaction network, we stop the experiment when the current state does not change. In addition, after transitions, it might reach a state from which it's impossible to return back to S by repeating transitions. In such cases we also stop the experiment. Your task as a bioinformatics researcher is to write a program to compute the expected time until we stop the experiment. If the expected time is infinite, output -1 instead.

A more precise description of the experiment termination criteria is the following. If there are no states that can be transited from the current state, we stop the experiment immediately. Otherwise, the experiment does not stop, if it is possible to return back to S from the current state by repeating zero or more transitions. In other words, the state S itself is always considered as a state that can return to S in this case.

Input

The input consists of a single test case in the format below.

$$
\begin{array}{l}
N\ M\ S \\
u_1\ v_1 \\
\vdots \\
u_M\ v_M
\end{array}
$$

The first line contains three integers N ($2 \leq N \leq 200$), M ($1 \leq M \leq \min(N(N-1), 1000)$), S ($1 \leq S \leq N$). N is the number of states in a chemical reaction network. M is the number of transitions between states. S is the initial state. The i-th of the following M lines represents the i-th transition, which means a state u_i can be transited to v_i ($1 \leq u_i, v_i \leq N$, $u_i \neq v_i$). Note that it does not necessarily imply v_i can be transitioned to u_i. It is guaranteed that the pairs u_i and v_i are distinct. That is, for $1 \leq i < j \leq M$, either $u_i \neq u_j$ or $v_i \neq v_j$ holds.

Output

Output the expected time to reach one of the states from which it's impossible to return back to the initial state S. The expected time can be huge, thus outputs the expected time modulo $10^9 + 7$. More precisely, if it is finite, the expected time can be represented as an irreducible fraction b/a, and you should output the minimum non-negative integer x which satisfies $ax \equiv b \bmod 10^9 + 7$. You may assume that such x exists for the given input.

If the expected time is infinity, output -1 instead.

Examples

Input	Output
3 3 1 1 2 2 1 2 3	4
5 7 1 1 2 2 1 2 3 3 1 3 4 4 1 4 5	22
10 17 5 4 6 8 10 5 8 7 3 8 3 3 5 9 6 4 8 3 2 6 1 3 7 2 10 2 8 10 2 2 9 8 6 5 4	800000011
3 2 3 1 2 2 3	0
3 3 3 1 2 2 3 3 1	-1

J: X-percent Blooming

- Time Limit: 2 sec

Spring is coming. Cherry blossom forecast is one of the things we feel spring is just around the corner. International Cherry-blossom Prediction Committee (ICPC) is an organization that announces the state of cherry blossoms every spring. Your task as a staff of the organization is to observe a specific cherry tree and evaluate the level of the blossom of the cherry tree.

The tree that you observe can be considered as a graph tree. There is a root node with no parent node, and the other nodes have only one parent node. Let $d(v)$ be the minimum distance from node v to the leaf nodes of the subtree of v. Here, a leaf node is a node that has no child nodes. Then a node v blooms if $d(v) \leq O$ at the time you observe. On the other hand, at full blossom, a node v blooms if $d(v) \leq F$. The level of cherry blossom is evaluated as the ratio of the numbers of blooming nodes at the time you observe and full blossom. More formally, let $b(x)$ be the number of nodes v satisfying $d(v) \leq x$. The level of cherry blossom is defined as $b(O)/b(F)$.

Note that the tree is also growing. When the tree grows, a new node is added to a node as a child node. So the number of nodes might be increased between your observations.

Initially, the tree had only one node, which is the root node. You recorded the growth and observation time but forgot to record the level of cherry blossom on each observation. Now, your new task is to restore the data of the levels of cherry blossom from the given records.

For Sample 1, the tree at the first observation looks like the figure below, where green border means a node is a leaf and pink circle means a node that blooms. The left tree represents a tree without blossom. The center tree represents blossom at the time you observe and the right tree represents full blossom.

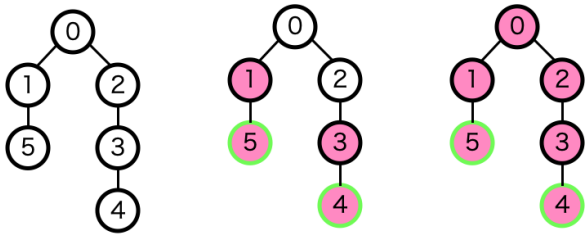

Figure J-1. First observation of Sample 1

Similarly, the following figure illustrates the second observation of Sample 1.

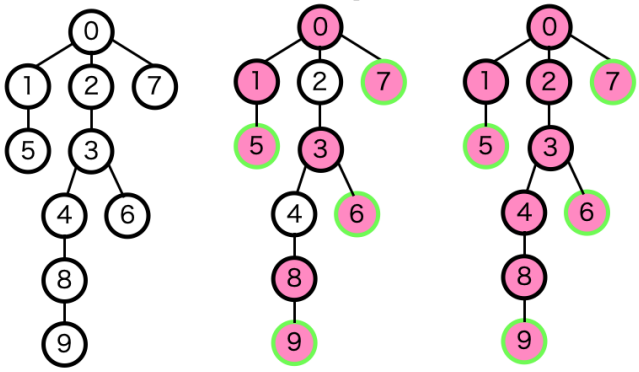

Figure J-2. Second observation of Sample 1

Input

The input consists of a single test case in the format below.

$Q\ O\ F$
$query_1$
$\vdots$
$query_Q$

The first line contains three integers Q ($1 \leq Q \leq 10^5$), O ($0 \leq O \leq 10^5$), F ($O \leq F \leq 10^5$). Q is the number of records you stored. O is the threshold of distances to determine blooming nodes at the time you observe. F is the threshold of distances to determine blooming nodes at full blossom. The i-th of the following Q lines represents the i-th recorded event. An event is either of two types of events:

- Growth event: a new node is added as a child node of the k_i-th added node, where the root node is the 0-th added node. For this type of event, $query_i$ is '1 k_i'.
- Observation event: evaluate the level of cherry blossom on the state of the tree at that time. For this type of event, $query_i$ is '2'. Note that whether each node bloomed in the past does not matter to the current observation. The node v does not bloom even if v bloomed in the past when the current $d(v) > x$.

Output

For each observation event, output the level of cherry blossom per line in the order of the events. Absolute or relative errors less than 10^{-7} are permissible for each level.

Examples

Input	Output
11 1 2 1 0 1 0 1 2 1 3 1 1 2 1 3 1 0 1 4 1 8 2	0.6666666667 0.8
10 2 3 1 0 1 1 2 1 2 1 3 2 1 4 1 5 1 0 2	1 0.75 0.83333333333333
12 2 2 1 0 2 1 1 1 2 2 2 1 0 1 0 1 0 1 5 1 6 2	1 1 1 1

K: Feed candies

- Time Limit: 2 sec

You are playing a game to bring a slime up. The slime has two integral parameters called *softness* and *transparency*. In this game, there are 10^{100} types of candies numbered from 1 to 10^{100}, and if you feed the i-th type of candy to the slime, its softness and transparency increase by s_i and t_i, respectively. Here, you know that s_i and t_i are calculated by the following formulae where A and B are integers.

- $(s_1, t_1) = (1, 0)$
- $(s_i, t_i) = (As_{i-1} - Bt_{i-1}, Bs_{i-1} + At_{i-1})$ for each $2 \leq i \leq 10^{100}$

In addition, the slime likes eating new types of candies. Therefore, you can feed each type of candy at most once.

Initially, the slime's softness and transparency are both zero. Your objective is to feed zero or more types of candies to the slime so that the slime's softness and transparency become X and Y, respectively. Determine whether this is possible, and if it is possible, find such a way.

In the first sample input, the characteristics of the first four types of candies are as follows.

- $(s_1, t_1) = (1, 0)$
- $(s_2, t_2) = (2, -1)$
- $(s_3, t_3) = (3, -4)$
- $(s_4, t_4) = (2, -11)$

If you feed the first, second and fourth types of candies to the slime, the slime's softness and transparency become $1 + 2 + 2 = 5$ and $0 + (-1) + (-11) = -12$, respectively.

Input

The input consists of multiple datasets. Each dataset is represented in the following format.

A B X Y

Each dataset consists of a single line which contains four integers A, B, X and Y. You may assume that $-100 \leq A \leq 100$, $-100 \leq B \leq 100$, $-10^{16} \leq X \leq 10^{16}$, $-10^{16} \leq Y \leq 10^{16}$ and $|A| + |B| \geq 2$.

The end of the input is represented by a line consisting of four zeros. The number of datasets should not exceed 200.

Output

For each dataset, if your objective is unachievable, print -1 in a single line. Otherwise, let m be the number of types of candies you feed to the slime. Print m on the first line. Then, for each $1 \leq k \leq m$, on the $(k+1)$-st line print the k-th smallest type of candy you feed to the slime.

If there are multiple correct answers, print any of them.

Examples

Input	Output
2 -1 5 -12 2 0 33 0 -10 0 123 0 -4 7 143800796 -5765753 0 0 0 0	3 1 2 4 2 1 6 -1 1 10