A

Almost Permutation

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Recently Ivan noticed an array a while debugging his code. Now Ivan can't remember this array, but the bug he was trying to fix didn't go away, so Ivan thinks that the data from this array might help him to reproduce the bug.

Ivan clearly remembers that there were n elements in the array, and each element was not less than 1 and not greater than n. Also he remembers q facts about the array. There are two types of facts that Ivan remembers:

- $1\ l_i\ r_i\ v_i$ — for each x such that $l_i \le x \le r_i\ a_x \ge v_i$;
- $2\ l_i\ r_i\ v_i$ — for each x such that $l_i \le x \le r_i\ a_x \le v_i$.

Also Ivan thinks that this array was a permutation, but he is not so sure about it. He wants to restore some array that corresponds to the q facts that he remembers and is very similar to permutation. Formally, Ivan has denoted the $cost$ of array as follows:

$cost = \sum_{i=1}^{n} (cnt(i))^2$, where $cnt(i)$ is the number of occurences of i in the array.

Help Ivan to determine minimum possible $cost$ of the array that corresponds to the facts!

Input

The first line contains two integer numbers n and q ($1 \le n \le 50$, $0 \le q \le 100$).

Then q lines follow, each representing a fact about the array. i-th line contains the numbers t_i, l_i, r_i and v_i for i-th fact ($1 \le t_i \le 2$, $1 \le l_i \le r_i \le n$, $1 \le v_i \le n$, t_i denotes the type of the fact).

Output

If the facts are controversial and there is no array that corresponds to them, print `-1`. Otherwise, print minimum possible $cost$ of the array.

Examples

input
3 0
output
3

input
3 1 1 1 3 2
output
5

input
3 2 1 1 3 2 2 1 3 2
output
9

input

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

output

```
-1
```

B New Year and the Tricolore Re

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Alice and Bob play a game on a grid with n rows and infinitely many columns. In each row, there are three tokens, blue, white and red one. **Before** the game starts and **after every move**, the following two conditions must hold:

- Any two tokens are not in the same cell.
- In each row, the blue token is to the left of the white token, and the red token is to the right of the white token.

First, they pick a positive integer f, whose value is valid for the whole game. Second, the starting player is chosen and makes his or her first turn. Then players take alternating turns. The player who is unable to make a move loses.

During a move, a player first selects an integer k that is either a prime number or a product of two (not necessarily distinct) primes. The smallest possible values of k are thus $2, 3, 4, 5, 6, 7, 9, 10, 11, 13, 14, 15, 17, 19, \ldots$. Furthermore, k must not be equal to the previously picked integer f. Each turn, a move is performed in exactly one of the rows.

If it is Alice's turn, she chooses a single blue token and moves it k cells to the right. Alternatively, she may move both the blue and the white token in the same row by the same amount k to the right.

On the other hand, Bob selects a single red token and moves it k cells to the left. Similarly, he may also move the white and the red token in the corresponding row by k to the left.

Note that Alice may never move a red token, while Bob may never move a blue one. Remember that after a move, the two conditions on relative positions of the tokens must still hold.

Both players play optimally. Given the initial state of the board, determine who wins for two games: if Alice starts and if Bob starts.

Input

The first line contains a two integers n and f ($1 \le n \le 10^5$, $2 \le f \le 2 \cdot 10^5$) — the number of rows and the forbidden move, respectively.

Each of the next n lines contains three integers b_i, w_i, r_i ($-10^5 \le b_i < w_i < r_i \le 10^5$) — the number of column in which the blue, white and red token lies in the i-th row, respectively.

Output

Output two lines.

The first line should contain the name of the winner when Alice starts, and the second line should contain the name of the winner when Bob starts.

Examples

input
1 6 0 3 9
output
Alice Bob

input

```
1 2
0 3 9
```

output

```
Alice
Bob
```

input

```
10 133
-248 -193 -187
97 101 202
-72 67 91
23 89 215
-129 -108 232
-223 -59 236
-99 86 242
-137 -109 -45
-105 173 246
-44 228 243
```

output

```
Bob
Alice
```

Note

The first example is as follows:

When Alice starts, she can win by moving the blue and white token to right by 2 cells, getting into position $2\ 5\ 9$. Regardless of what Bob does, Alice will have one more move and then the game is over. For instance, he can move both the red and white token by 2 cells to the left, reaching state $2\ 3\ 7$. Alice can then move blue and white token by 2 to move into $4\ 5\ 7$, where no more moves are possible.

If Bob starts, he gains enough advantage to win. For instance, he may move the red token by 3 to the left, getting into position $0\ 3\ 6$. Alice can, for example, move the blue token by 2, which is countered by Bob by moving the red token by 2. The game ends in position $2\ 3\ 4$.

In the second example, it is forbidden to move by 2, but this doesn't stop Alice from winning! She can move the blue and white token by 4, getting into position $4\ 7\ 9$. Now Bob has no move, since moving by 2 is forbidden.

C A Tide of Riverscape

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Walking along a riverside, Mino silently takes a note of something.
"Time," Mino thinks aloud.

"What?"

"Time and tide wait for no man," explains Mino. "My name, taken from the river, always reminds me of this."

"And what are you recording?"

"You see it, tide. Everything has its own period, and I think I've figured out this one," says Mino with confidence.

Doubtfully, Kanno peeks at Mino's records.

The records are expressed as a string s of characters '`0`', '`1`' and '`.`', where '`0`' denotes a low tide, '`1`' denotes a high tide, and '`.`' denotes an unknown one (either high or low).

You are to help Mino determine whether it's possible that after replacing each '`.`' independently with '`0`' or '`1`', a given integer p is **not** a period of the resulting string. In case the answer is yes, please also show such a replacement to Mino.

In this problem, a positive integer p is considered a period of string s, if for all $1 \leq i \leq |s| - p$, the i-th and $(i + p)$-th characters of s are the same. Here $|s|$ is the length of s.

Input

The first line contains two space-separated integers n and p ($1 \leq p \leq n \leq 2000$) — the length of the given string and the supposed period, respectively.

The second line contains a string s of n characters — Mino's records. s only contains characters '`0`', '`1`' and '`.`', and contains at least one '`.`' character.

Output

Output one line — if it's possible that p is **not** a period of the resulting string, output any one of such strings; otherwise output "`No`" (without quotes, you can print letters in any case (upper or lower)).

Examples

input
10 7 1.0.1.0.1.
output
1000100010

input

```
10 6
1.0.1.1000
```

output

```
1001101000
```

input

```
10 9
1........1
```

output

```
No
```

Note

In the first example, 7 is not a period of the resulting string because the 1-st and 8-th characters of it are different.

In the second example, 6 is not a period of the resulting string because the 4-th and 10-th characters of it are different.

In the third example, 9 is always a period because the only constraint that the first and last characters are the same is already satisfied.

Note that there are multiple acceptable answers for the first two examples, you can print any of them.

D Bad Luck Island

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Bad Luck Island is inhabited by three kinds of species: r rocks, s scissors and p papers. At some moments of time two random individuals meet (all pairs of individuals can meet equiprobably), and if they belong to different species, then one individual kills the other one: a rock kills scissors, scissors kill paper, and paper kills a rock. Your task is to determine for each species what is the probability that this species will be the only one to inhabit this island after a long enough period of time.

Input

The single line contains three integers r, s and p ($1 \le r, s, p \le 100$) — the original number of individuals in the species of rock, scissors and paper, respectively.

Output

Print three space-separated real numbers: the probabilities, at which the rocks, the scissors and the paper will be the only surviving species, respectively. The answer will be considered correct if the relative or absolute error of each number doesn't exceed 10^{-9}.

Examples

input
2 2 2
output
0.333333333333 0.333333333333 0.333333333333

input
2 1 2
output
0.150000000000 0.300000000000 0.550000000000

input
1 1 3
output
0.057142857143 0.657142857143 0.285714285714

E A shade of moonlight

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Gathering darkness shrouds the woods and the world. The moon sheds its light on the boat and the river.
"To curtain off the moonlight should be hardly possible; the shades present its mellow beauty and restful nature." Intonates Mino.

"See? The clouds are coming." Kanno gazes into the distance.

"That can't be better," Mino turns to Kanno.

The sky can be seen as a one-dimensional axis. The moon is at the origin whose coordinate is 0.

There are n clouds floating in the sky. Each cloud has the same length l. The i-th initially covers the range of $(x_i, x_i + l)$ (**endpoints excluded**). Initially, it moves at a velocity of v_i, which equals either 1 or -1.

Furthermore, no pair of clouds intersect initially, that is, for all $1 \le i < j \le n$, $|x_i - x_j| \ge l$.

With a wind velocity of w, the velocity of the i-th cloud becomes $v_i + w$. That is, its coordinate increases by $v_i + w$ during each unit of time. Note that the wind can be strong and clouds can change their direction.

You are to help Mino count the number of pairs (i, j) $(i < j)$, such that with a proper choice of wind velocity w not exceeding $w_{\max}$ in absolute value (possibly negative and/or fractional), the i-th and j-th clouds both cover the moon at the same future moment. This w doesn't need to be the same across different pairs.

Input

The first line contains three space-separated integers n, l, and $w_{\max}$ ($1 \le n \le 10^5$, $1 \le l, w_{\max} \le 10^8$) — the number of clouds, the length of each cloud and the maximum wind speed, respectively.

The i-th of the following n lines contains two space-separated integers x_i and v_i ($-10^8 \le x_i \le 10^8$, $v_i \in \{-1, 1\}$) — the initial position and the velocity of the i-th cloud, respectively.

The input guarantees that for all $1 \le i < j \le n$, $|x_i - x_j| \ge l$.

Output

Output one integer — the number of unordered pairs of clouds such that it's possible that clouds from each pair cover the moon at the same future moment with a proper choice of wind velocity w.

Examples

input
5 1 2 -2 1

```
2 1
3 -1
5 -1
7 -1
```

output

```
4
```

input

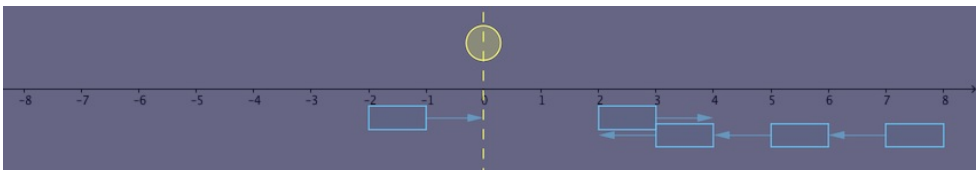
```
4 10 1
-20 1
-10 -1
0 1
10 -1
```

output

```
1
```

Note

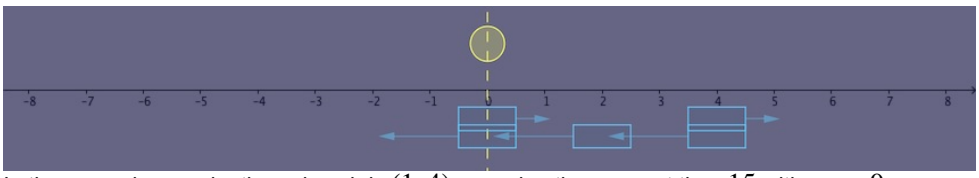
In the first example, the initial positions and velocities of clouds are illustrated below.



The pairs are:

- $(1, 3)$, covering the moon at time 2.5 with $w = -0.4$;
- $(1, 4)$, covering the moon at time 3.5 with $w = -0.6$;
- $(1, 5)$, covering the moon at time 4.5 with $w = -0.7$;
- $(2, 5)$, covering the moon at time 2.5 with $w = -2$.

Below is the positions of clouds at time 2.5 with $w = -0.4$. At this moment, the 1-st and 3-rd clouds both cover the moon.



In the second example, the only pair is $(1, 4)$, covering the moon at time 15 with $w = 0$.

Note that all the times and wind velocities given above are just examples among infinitely many choices.

F Mashmokh and ACM

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Mashmokh's boss, Bimokh, didn't like Mashmokh. So he fired him. Mashmokh decided to go to university and participate in ACM instead of finding a new job. He wants to become a member of Bamokh's team. In order to join he was given some programming tasks and one week to solve them. Mashmokh is not a very experienced programmer. Actually he is not a programmer at all. So he wasn't able to solve them. That's why he asked you to help him with these tasks. One of these tasks is the following.

A sequence of l integers $b_1, b_2, ..., b_l (1 \le b_1 \le b_2 \le ... \le b_l \le n)$ is called *good* if each number divides (without a remainder) by the next number in the sequence. More formally $b_i \mid b_{i+1}$ for all $i (1 \le i \le l - 1)$.

Given n and k find the number of good sequences of length k. As the answer can be rather large print it modulo $1000000007 (10^9 + 7)$.

Input

The first line of input contains two space-separated integers $n, k (1 \le n, k \le 2000)$.

Output

Output a single integer — the number of good sequences of length k modulo $1000000007 (10^9 + 7)$.

Examples

input
3 2
output
5

input
6 4
output
39

input
2 1
output
2

Note

In the first sample the good sequences are: $[1, 1], [2, 2], [3, 3], [1, 2], [1, 3]$.

Supported by

G Nearest Leaf

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Let's define the Eulerian traversal of a tree (a connected undirected graph without cycles) as follows: consider a depth-first search algorithm which traverses vertices of the tree and enumerates them in the order of visiting (only the first visit of each vertex counts). This function starts from the vertex number 1 and then recursively runs from all vertices which are connected with an edge with the current vertex and are not yet visited in increasing numbers order. Formally, you can describe this function using the following pseudocode:

```
next_id = 1
id = array of length n filled with -1
visited = array of length n filled with false

function dfs(v):
    visited[v] = true
    id[v] = next_id
    next_id += 1
    for to in neighbors of v in increasing order:
        if not visited[to]:
            dfs(to)
```

You are given a weighted tree, the vertices of which were enumerated with integers from 1 to n using the algorithm described above.

A *leaf* is a vertex of the tree which is connected with only one other vertex. In the tree given to you, the vertex 1 is not a leaf. The distance between two vertices in the tree is the sum of weights of the edges on the simple path between them.

You have to answer q queries of the following type: given integers v, l and r, find the shortest distance from vertex v to one of the leaves with indices from l to r inclusive.

Input

The first line contains two integers n and q ($3 \le n \le 500\,000, 1 \le q \le 500\,000$) — the number of vertices in the tree and the number of queries, respectively.

The $(i - 1)$-th of the following $n - 1$ lines contains two integers p_i and w_i ($1 \le p_i < i, 1 \le w_i \le 10^9$), denoting an edge between vertices p_i and i with the weight w_i.

It's guaranteed that the given edges form a tree and the vertices are enumerated in the Eulerian traversal order and that the vertex with index 1 is not a leaf.

The next q lines describe the queries. Each of them contains three integers v_i, l_i, r_i ($1 \le v_i \le n, 1 \le l_i \le r_i \le n$), describing the parameters of the query. It is guaranteed that there is at least one leaf with index x such that $l_i \le x \le r_i$.

Output

Output q integers — the answers for the queries in the order they are given in the input.

Examples

input
5 3 1 10 1 1

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

output

```
3
0
13
```

input

```
5 3
1 1000000000
2 1000000000
1 1000000000
1 1000000000
3 4 5
2 1 5
2 4 5
```

output

```
3000000000
1000000000
2000000000
```

input

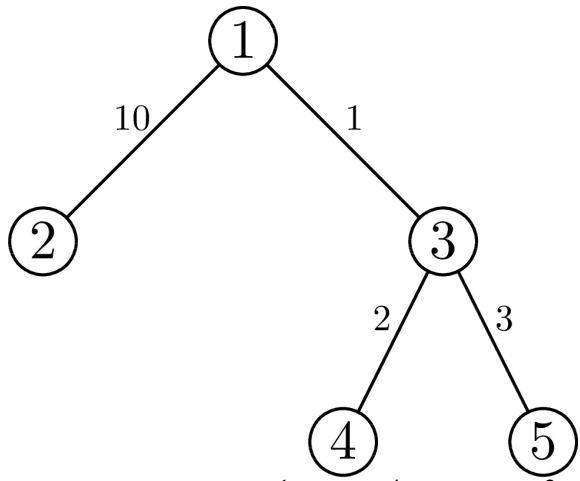
```
11 8
1 7
2 1
1 20
1 2
5 6
6 2
6 3
5 1
9 10
9 11
5 1 11
1 1 4
9 4 8
6 1 4
9 7 11
9 10 11
8 1 11
11 4 5
```

output

```
8
8
9
16
9
10
0
34
```

Note

In the first example, the tree looks like this:



In the first query, the nearest leaf for the vertex 1 is vertex 4 with distance 3. In the second query, the nearest leaf for vertex 5 is vertex 5 with distance 0. In the third query the nearest leaf for vertex 4 is vertex 4; however, it is not inside interval $[1, 2]$ of the query. The only leaf in interval $[1, 2]$ is vertex 2 with distance 13 from vertex 4.

H

Weak Memory

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Zart PMP is qualified for ICPC World Finals in Harbin, China. After team excursion to Sun Island Park for snow sculpture art exposition, PMP should get back to buses before they leave. But the park is really big and he does not know how to find them.

The park has n intersections numbered 1 through n. There are m bidirectional roads that connect some pairs of these intersections. At k intersections, ICPC volunteers are helping the teams and showing them the way to their destinations. Locations of volunteers are fixed and distinct.

When PMP asks a volunteer the way to bus station, he/she can tell him the whole path. But the park is fully covered with ice and snow and everywhere looks almost the same. So PMP can only memorize at most q intersections after each question (excluding the intersection they are currently standing). He always tells volunteers about his weak memory and if there is no direct path of length (in number of roads) at most q that leads to bus station, the volunteer will guide PMP to another volunteer (who is at most q intersections away, of course). ICPC volunteers know the area very well and always tell PMP the best way. So if there exists a way to bus stations, PMP will definitely find it.

PMP's initial location is intersection s and the buses are at intersection t. There will always be a volunteer at intersection s. Your job is to find out the minimum q which guarantees that PMP can find the buses.

Input

The first line contains three space-separated integers n, m, k ($2 \le n \le 10^5, 0 \le m \le 2 \cdot 10^5, 1 \le k \le n$) — the number of intersections, roads and volunteers, respectively. Next line contains k distinct space-separated integers between 1 and n inclusive — the numbers of cities where volunteers are located.

Next m lines describe the roads. The i-th of these lines contains two space-separated integers u_i, v_i ($1 \le u_i, v_i \le n, u_i \ne v_i$) — two intersections that i-th road connects. There will be at most one road between any two intersections.

Last line of input contains two space-separated integers s, t ($1 \le s, t \le n, s \ne t$) — the initial location of PMP and the location of the buses. It might not always be possible to reach t from s.

It is guaranteed that there is always a volunteer at intersection s.

Output

Print on the only line the answer to the problem — the minimum value of q which guarantees that PMP can find the buses. If PMP cannot reach the buses at all, output -1 instead.

Examples

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

```
3
```

input

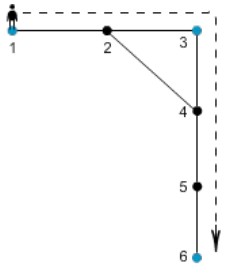
```
6 5 3
1 5 6
1 2
2 3
3 4
4 5
6 3
1 5
```

output

```
3
```

Note

The first sample is illustrated below. Blue intersections are where volunteers are located. If PMP goes in the path of dashed line, it can reach the buses with $q = 3$:



In the second sample, PMP uses intersection 6 as an intermediate intersection, thus the answer is 3.

I Copying Data

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

We often have to copy large volumes of information. Such operation can take up many computer resources. Therefore, in this problem you are advised to come up with a way to copy some part of a number array into another one, quickly.

More formally, you've got two arrays of integers $a_1, a_2, ..., a_n$ and $b_1, b_2, ..., b_n$ of length n. Also, you've got m queries of two types:

1. Copy the subsegment of array a of length k, starting from position x, into array b, starting from position y, that is, execute $b_{y+q} = a_{x+q}$ for all integer q $(0 \le q < k)$. The given operation is correct — both subsegments do not touch unexistent elements.
2. Determine the value in position x of array b, that is, find value b_x.

For each query of the second type print the result — the value of the corresponding element of array b.

Input

The first line contains two space-separated integers n and m $(1 \le n, m \le 10^5)$ — the number of elements in the arrays and the number of queries, correspondingly. The second line contains an array of integers $a_1, a_2, ..., a_n$ $(|a_i| \le 10^9)$. The third line contains an array of integers $b_1, b_2, ..., b_n$ $(|b_i| \le 10^9)$.

Next m lines contain the descriptions of the queries. The i-th line first contains integer t_i — the type of the i-th query $(1 \le t_i \le 2)$. If $t_i = 1$, then the i-th query means the copying operation. If $t_i = 2$, then the i-th query means taking the value in array b. If $t_i = 1$, then the query type is followed by three integers x_i, y_i, k_i $(1 \le x_i, y_i, k_i \le n)$ — the parameters of the copying query. If $t_i = 2$, then the query type is followed by integer x_i $(1 \le x_i \le n)$ — the position in array b.

All numbers in the lines are separated with single spaces. It is guaranteed that all the queries are correct, that is, the copying borders fit into the borders of arrays a and b.

Output

For each second type query print the result on a single line.

Examples

input
5 10 1 2 0 -1 3 3 1 5 -2 0 2 5 1 3 3 3 2 5 2 4 2 1 1 2 1 4 2 1 2 4 1 4 2 1 2 2

output
0 3 -1

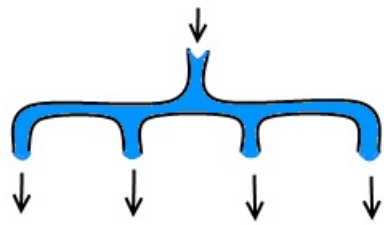
```
3
2
3
-1
```

J Pipeline

time limit per test: 0.4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Vova, the Ultimate Thule new shaman, wants to build a pipeline. As there are exactly n houses in Ultimate Thule, Vova wants the city to have exactly n pipes, each such pipe should be connected to the water supply. A pipe can be connected to the water supply if there's water flowing out of it. Initially Vova has only one pipe with flowing water. Besides, Vova has several splitters.

A splitter is a construction that consists of one input (it can be connected to a water pipe) and x output pipes. When a splitter is connected to a water pipe, water flows from each output pipe. You can assume that the output pipes are ordinary pipes. For example, you can connect water supply to such pipe if there's water flowing out from it. At most one splitter can be connected to any water pipe.



The figure shows a 4-output splitter

Vova has one splitter of each kind: with $2, 3, 4, ..., k$ outputs. Help Vova use the minimum number of splitters to build the required pipeline or otherwise state that it's impossible.

Vova needs the pipeline to have exactly n pipes with flowing out water. Note that some of those pipes can be the output pipes of the splitters.

Input

The first line contains two space-separated integers n and k ($1 \le n \le 10^{18}$, $2 \le k \le 10^{9}$).

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Output

Print a single integer — the minimum number of splitters needed to build the pipeline. If it is impossible to build a pipeline with the given splitters, print -1.

Examples

input
4 3
output
2

input
5 5
output
1

input

8 4
output
-1

K

Crazy Town

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Crazy Town is a plane on which there are n infinite line roads. Each road is defined by the equation $a_i x + b_i y + c_i = 0$, where a_i and b_i are not both equal to the zero. The roads divide the plane into connected regions, possibly of infinite space. Let's call each such region a block. We define an intersection as the point where at least two different roads intersect.

Your home is located in one of the blocks. Today you need to get to the University, also located in some block. In one step you can move from one block to another, if the length of their common border is nonzero (in particular, this means that if the blocks are adjacent to one intersection, but have no shared nonzero boundary segment, then it are not allowed to move from one to another one in one step).

Determine what is the minimum number of steps you have to perform to get to the block containing the university. It is guaranteed that neither your home nor the university is located on the road.

Input

The first line contains two space-separated integers x_1, y_1 ($-10^6 \le x_1, y_1 \le 10^6$) — the coordinates of your home.

The second line contains two integers separated by a space x_2, y_2 ($-10^6 \le x_2, y_2 \le 10^6$) — the coordinates of the university you are studying at.

The third line contains an integer n ($1 \le n \le 300$) — the number of roads in the city. The following n lines contain 3 space-separated integers ($-10^6 \le a_i, b_i, c_i \le 10^6$; $|a_i| + |b_i| > 0$) — the coefficients of the line $a_i x + b_i y + c_i = 0$, defining the i-th road. It is guaranteed that no two roads are the same. In addition, neither your home nor the university lie on the road (i.e. they do not belong to any one of the lines).

Output

Output the answer to the problem.

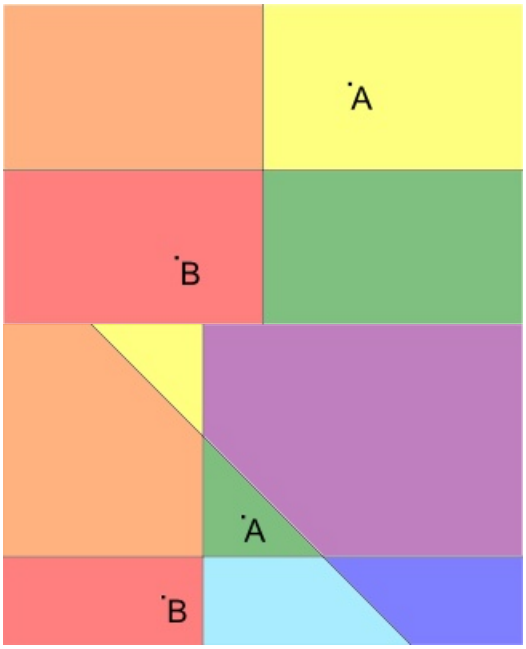
Examples

input
1 1 -1 -1 2 0 1 0 1 0 0
output
2

input
1 1 -1 -1 3 1 0 0 0 1 0 1 1 -3
output
2

Note

Pictures to the samples are presented below (A is the point representing the house; B is the point representing the university, different blocks are filled with different colors):



Vasya and Chess

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Vasya decided to learn to play chess. Classic chess doesn't seem interesting to him, so he plays his own sort of chess.

The queen is the piece that captures all squares on its vertical, horizontal and diagonal lines. If the cell is located on the same vertical, horizontal or diagonal line with queen, and the cell contains a piece of the enemy color, the queen is able to move to this square. After that the enemy's piece is removed from the board. The queen cannot move to a cell containing an enemy piece if there is some other piece between it and the queen.

There is an $n \times n$ chessboard. We'll denote a cell on the intersection of the r-th row and c-th column as (r, c). The square $(1, 1)$ contains the white queen and the square $(1, n)$ contains the black queen. All other squares contain green pawns that don't belong to anyone.

The players move in turns. The player that moves first plays for the white queen, his opponent plays for the black queen.

On each move the player has to capture some piece with his queen (that is, move to a square that contains either a green pawn or the enemy queen). The player loses if either he cannot capture any piece during his move or the opponent took his queen during the previous move.

Help Vasya determine who wins if both players play with an optimal strategy on the board $n \times n$.

Input

The input contains a single number n $(2 \le n \le 10^9)$ — the size of the board.

Output

On the first line print the answer to problem — string "`white`" or string "`black`", depending on who wins if the both players play optimally.

If the answer is "`white`", then you should also print two integers r and c representing the cell (r, c), where the first player should make his first move to win. If there are multiple such cells, print the one with the minimum r. If there are still multiple squares, print the one with the minimum c.

Examples

input
2

output
white 1 2

input
3

output
black

Note

In the first sample test the white queen can capture the black queen at the first move, so the white player wins.

In the second test from the statement if the white queen captures the green pawn located on the central vertical line, then it will be captured by the black queen during the next move. So the only move for the white player is to capture the green pawn located at $(2, 1)$.

Similarly, the black queen doesn't have any other options but to capture the green pawn located at $(2, 3)$, otherwise if it goes to the middle vertical line, it will be captured by the white queen.

During the next move the same thing happens — neither the white, nor the black queen has other options rather than to capture green pawns situated above them. Thus, the white queen ends up on square $(3, 1)$, and the black queen ends up on square $(3, 3)$.

In this situation the white queen has to capture any of the green pawns located on the middle vertical line, after that it will be captured by the black queen. Thus, the player who plays for the black queen wins.
