Problem A. Beer Problem

Input file:	`beer.in`
Output file:	`beer.out`
Time limit:	1 second
Memory limit:	64 megabytes

Everyone knows that World Finals of ACM ICPC 2004 were held in Prague. Besides its greatest architecture and culture, Prague is world famous for its beer. Though drinking too much is probably not good for contestants, many teams took advantage of tasting greatest beer for really low prices.

A new beer producing company *Drink Anywhere* is planning to distribute its product in several of the n European cities. The brewery is located near Prague, that we would certainly call city number 1. For delivering beer to other cities, the company is planning to use logistics company *Drive Anywhere* that provides m routes for carrying goods. Each route is described by the cities it connects (products can be transported in either direction), its capacity — the number of barrels of beer that can be transported along this route each day, and the cost of transporting one barrel of beer along it. To deliver beer to some city it may be necessary (or advantageous) to use several routes consequently, and maybe even deliver beer using several different paths.

Each city is in turn characterized by the price that local pubs and restaurants are ready to pay for one barrel of beer. You may assume that demand for beer is essentially unlimited in each city, since this is the product that will always find its consumer.

Drink Anywhere is not planning to distribute its beer in Prague for a while, because of the high competition there, so it is just planning to provide beer to other cities for now. Help it to find out, what is the maximal income per day it can get.

Input

The first line of the input file contains n and m — the number of cities in Europe we consider and the number of delivery routes respectively ($2 \le n \le 100$, $1 \le m \le 2000$). The next line contains $n-1$ integer numbers — prices of a barrel of beer in European cities $2, 3 \ldots, n$ respectively (prices are positive integers and do not exceded 1000).

The following m lines contain four integer numbers each and describe delivery routes. Each route is specified by the numbers of cities it connects, its capacity, and the price of transporting one barrel of beer along it (the capacity and the price are positive integers, they do not exceed 1000).

Output

Output the maximal income the company can get each day.

Example

beer.in	beer.out

4 4
80 50 130
1 2 80 50
2 4 40 90
3 1 40 60
3 4 30 50
	3000

The company should deliver 80 barrels of beer to the second city (using the first route it costs 50 per barrel to deliver beer, income is 30 per barrel, 2400 total), and 30 barrels to the fourth city (the best path uses routes 3 and 4, it costs 110 to deliver a barrel, income is 20 per barrel, 600 total). It is not profitable to deliver beer to the third city, so the company should not do it.

Problem B. Another Brick in the Wall

Input file:	`brick.in`
Output file:	`brick.out`
Time limit:	1 second
Memory limit:	64 megabytes

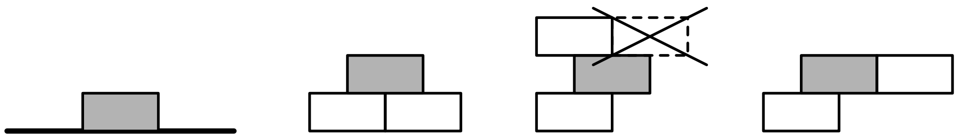
The company *Wall Construction Inc* (WC Inc) is preparing its new software project. The goal of this project is to automatically plan garden walls. You, the company's leading developer, are asked to quickly write the prototype of the system.

The wall is constructed of several bricks. Bricks are organized in layers, the bottom layer contains m bricks lying next to each other. The total number of layers is n. The bricks in each next layer are shifed by one half of the brick horizontaly compared to the bricks of the previous layer.

The number of bricks in different levels may be different, however the wall must be stable. The wall is considered stable if all of its bricks are stable. The brick is stable if one of the following conditions is satisfied:

1. the brick belongs to the first level;
2. the brick is lying on two bricks (none of the two bricks directly under it is missing);
3. the brick is lying on one brick and:
   - there is a brick directly upon it on the side that is lying on a brick;
   - there is no brick directly upon it on the side that is not lying on a brick;
4. the brick is lying on one brick and there is a brick in the same layer, directly next to it at the side that is not lying on a brick.

The pictures below show possible cases of stable bricks.



Note, that only bricks that affect the stability of the shaded brick are shown. Since all these bricks must also be stable, actually some other bricks must also exist for the whole wall to be stable.

Your first task is, given m and n find the number of stable walls.

Input

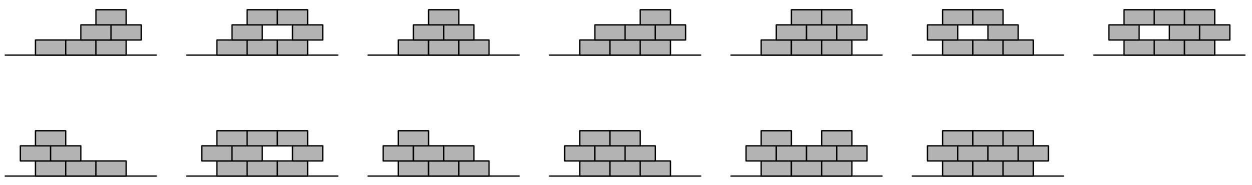
The input file contains m and n ($1 \le m, n \le 8$).

Output

Output the number of stable walls with m bricks in the first layer and n layers.

Example

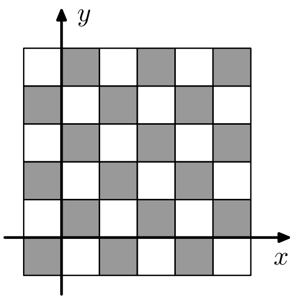
`brick.in`	`brick.out`
`3 3`	`13`

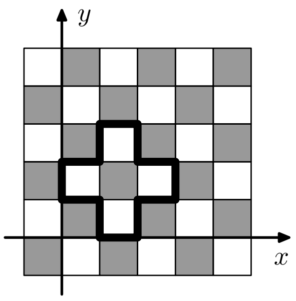


Problem C. Black and White

Input file:	`bw.in`
Output file:	`bw.out`
Time limit:	1 second
Memory limit:	64 megabytes

Consider an infinite chessboard. Introduce a coordinate system on it in such a way that chessboard cells are unit squares with integer corner coordates. Let the cells be colored black and white like on the standard chessboard, let the cell with bottom left corner at $(0, 0)$ be colored black.





Somebody has drawn a closed polyline on the board. The vertices of the polyline are in the corners of the cells and its sides are parallel to the coordinate axes. It's interesting, what is the number of black and white cells inside the polyline. Find that out.

Input

The first line of the input file contains n — the number of vertices of the polyline ($1 \le n \le 50\,000$). The following n lines contain the coordinates of the vertices in counter-clockwise order. Coordinates are integer and do not exceed 10^9 by their absolute values. Polyline has no self-intersections and no self-touchings.

Output

Output two numbers: b and w — the number of black and white cells inside the polyline repectively.

Example

`bw.in`	`bw.out`

`12`
`1 0`
`2 0`
`2 1`
`3 1`
`3 2`
`2 2`
`2 3`
`1 3`
`1 2`
`0 2`
`0 1`
`1 1`
	`1 4`

Problem D. Integer Numbers

Input file:	`integer.in`
Output file:	`integer.out`
Time limit:	1 second
Memory limit:	64 megabytes

The boy likes numbers. He has a sheet of paper. He have written a sequence of consecutive integer numbers on the sheet. The boy likes them.

But then the girl came. The girl is cruel. She changed some of the numbers.

The boy is disappointed. He cries. He does not like all these random numbers. He likes consecutive numbers. He really likes them. But his numbers are not consecutive any more. The boy is disappointed. He cries.

Help the boy. He can change some numbers. He would not like to change many of them. He would like to change as few as possible. He cannot change their order. He would like the numbers to be consecutive again. Help the boy.

Input

The first line of the input file contains n — the number of numbers in the sequence ($1 \le n \le 50\,000$). The next line contains the sequence itself — integer numbers not exceeding 10^9 by their absolute values.

Output

Output the minimal number of numbers that the boy must change. After that output the sequence after the change.

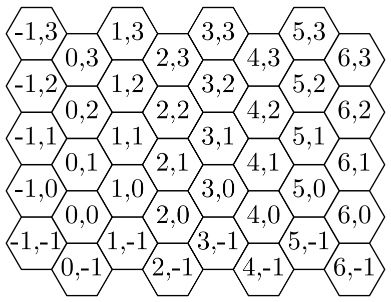
Example

integer.in	integer.out
6 5 4 5 2 1 8	3 3 4 5 6 7 8

Problem E. Islands

Input file:	`islands.in`
Output file:	`islands.out`
Time limit:	1 second
Memory limit:	64 megabytes

You are working in a team developing the new strategic game *Island Warriors*. The game proceeds on a hexagonal grid, the coordinate system on the map is introduced as shown on the picture below.



A hexagon can be either land, or sea. An *island* is a maximal connected set of land hexagons, an area of the island is the number of hexagons in it.

Your current task is writing random map generator. The map generated must satisfy some conditions, in particular no island area must exceed s.

The design of your module is the following. Initially all hexagons are sea. Special random generator provides you with the sequence of hexagons. Getting the next hexagon, you check whether it is already the land one. If it is, you just skip this hexagon. If it is sea, you check whether convering it to land results in an island with area exceeding s. If it does, you also skip this hexagon. In the other case you convert it to land.

So, the design step is completed, now its time to implement the module. Fortunately, your teammate has already implemented random generator, so you just have to implement the rest of the module. To check yourself you decided first to output only the number of islands in the resulting map and their areas.

Input

The first line of the input file contains n — the number of hexagons provided to you by random generator ($1 \le n \le 50\,000$), and s ($1 \le s \le n$).

The following n lines contain two integer numbers each — coordinates of the hexagons. Coordinates do not exceed 500 by their absolute values.

Output

Output the number of islands in the resulting map and their areas. List areas in the ascending order.

Example

`islands.in`	`islands.out`

`7 4`
`0 1`
`2 1`
`3 0`
`1 -1`
`2 1`
`1 0`
`0 0`
	`2`
	`2 3`

Problem F. Counterfeit Money

Input file:	`money.in`
Output file:	`money.out`
Time limit:	1 second
Memory limit:	64 megabytes

With the development of modern colorful copying and printing hardware as well as image processing software the problem of counterfeit money has become critical. To prevent spreading of the counterfeit money, the financial organizations force image processing software developers to insert special functions into their software that would prevent users from operating with images of banknotes.

You are working on a prototype of such system and are asked to implement the routine that would check whether the loaded image is the modified image of a banknote. To make a first test you decided to find a largest common rectangle on the loaded image and the image of the banknote and if it is large enough, report it to the other routine, that would make a detailed check.

Given two rectangular images find their common rectangle with the largest area.

Input

The first line of the input file contains m_1 and n_1 — the height and the width of the first image. The next m_1 lines contain n_1 letters each — different letters correspond to different colors of the image. Only capital letters of the English alphabet are used.

The rest of the file describes the second image in the same format. Both width and height of each image do not exceed 40.

Output

On the first line of the output file print the height and the width of the largest common rectangle of the two images. On the second line print the coorindates of its upper-left corner in the first rectangle (first coordinate is the row, second is the column, rows are numbered from top to bottom, columns — from left to right). On the third line print the coordinates of its upper-left corner in the second rectangle.

If no common rectangle exists, print two zeroes on the first and the only line of the output file.

Example

money.in	money.out

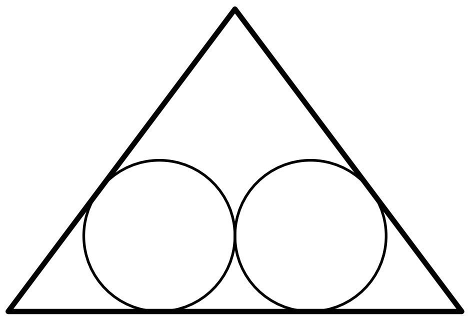
3 4
ABAB
CBDC
BACD
4 3
BAB
BDB
ACD
ABA
	3 2
	1 2
	1 1

1 1 A 1 1 B	0 0

Problem G. Chinese New Year

Input file:	`newyear.in`
Output file:	`newyear.out`
Time limit:	1 second
Memory limit:	64 megabytes

Jerry decided to congratulate his friend Jessica with Chinese New Year upcoming. He bought her a present — a nice collection of n colored candles. Now he wants to pack them to present to Jessica. He has made a nice triangular box and started to put candles inside, when it turned out that the box is too small. So now he wants to know how many candles he can put into the box. Help him.



The box is a triangle with integer side lengths in inches. Each candle is a cylinder with radius of one inch. All candles must be put into the box vertically, so we can consider each candle as a circle when looking from above. Candle may touch each other and the box sides, but of course they must not be deformed.

Input

The first line of the input file contains n ($1 \le n \le 10$). The next line contains three integer numbers — the lengths of the sides of the box. The lengths are positive and do not exceded 10.

Output

Output the maximal number of candles Jerry can pack into his box.

Example

newyear.in	newyear.out
3 6 5 5	2
1 10 10 10	1
7 1 1 1	0

Problem H. Saving Princess

Input file:	`princess.in`
Output file:	`princess.out`
Time limit:	1 second
Memory limit:	64 megabytes

Saving princesses is always a hard work. Ivan D'Ourack is planning to save the princess locked in the tower. However, n dangerous monsters are guarding the road from the city where Ivan lives to the tower where the princess is locked.

Fortunately Ivan is a warrior and a magician. Thus he can defeat monsters in a fight, and enchant them to pass unnoticed.

Initially Ivan has h health points, strength s, spell power p and m mana points. To defeat i-th monster in a fight, he must have strength at least s_i, and he loses $\max(2s_i - s, 0)$ health points in a fight. If the number of health points becomes 0 or less, Ivan dies. After defeating a monster Ivan's strength increases by 1.

To enchant i-th monster Ivan must have spell power at least p_i and he spends m_i mana points to do it. If Ivan does not have m_i mana points, he cannot enchant the monster. After enchanting the monster Ivan's spell power increases by 1.

Find out, whether Ivan can save princess, and if he can how to do it.

Input

The first line of the input file contains n, h, s, p and m ($1 \le n \le 50$, $1 \le h \le 50$, $0 \le s, p, m \le 50$). The following n lines contain three integer numbers each — s_i, p_i, and m_i ($1 \le s_i, p_i, m_i \le 50$).

Output

If Ivan cannot save princess, output "`UNLUCKY`". In the other case output n characters, the i-th character must be '`D`' if Ivan must defeat the i-the monster, or '`E`' if he must enchant it.

Example

princess.in	princess.out

3 12 5 5 6
5 5 2
6 5 2
6 7 3
	DED

3 11 5 5 6
5 5 2
6 5 2
6 7 3
	UNLUCKY

Problem I. Radio Waves

Input file:	`radio.in`
Output file:	`radio.out`
Time limit:	1 second
Memory limit:	64 megabytes

It is well known, that radio waves interfere, so if two closely located radio transmitters use the same frequency, the quality of the programs is poor.

The company *NateRadio* is planning to transmit its programs in Flatland. The Ministry of Health and Communications has issued the license to the company that allows it to use two different frequencies for the transmission.

The company has n slots for radio transmitters located at different points of the country. However, the transmitters themselves are not yet bought. The managers of the company want to know what power the transmitters should be. The power of the transmitter affects the range it transmits the radio program to.

All transmitters will have the same power. Transmitters will be installed into the slots and each will be tuned to one of the two allowed frequencies. The power of the transmitters must be selected in such a way that the programs from no two transmitters tuned to the same frequency are simultaneously received at any point. Each slot must be used.

Since the company wants the programs to be received by most people, it wants to install as powerful transmitters as possible — the range of the transmission must be maximal.

Input

The first line of the input file contains n — the number of slots for the transmitters ($3 \le n \le 1200$). The following n lines contain two integer numbers each — the coordinates of the corresponding slots. Coordinates do not exceed 10^4 by their absolute value. All slots are different.

Output

On the first line of the output file print the maximal possible range of the transmitters. On the second line print n integer numbers — for each transmitter slot print 1 if the transmitter in this slot must use the first frequency, or 2 if it must use the second one.

You answer must be accurate up to 10^{-6}. All inaccurances under this level will be resolved in your favor.

Example

radio.in	radio.out

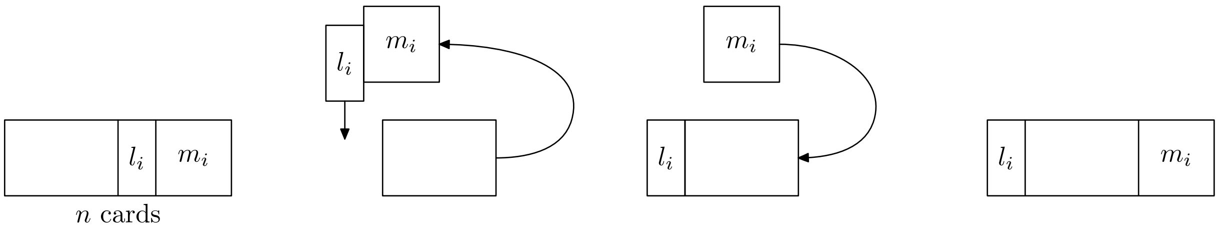
4
0 0
0 1
1 0
1 1
	0.70710678118654752
	1 2 2 1

Problem J. Cheater's Shuffle

Input file:	`shuffle.in`
Output file:	`shuffle.out`
Time limit:	1 second
Memory limit:	64 megabytes

Many card cheaters tricks are based on the shuffling of the deck. Collecting the cards from the table and shuffling the deck in a special way, the cheater may get the desired order of the cards. In this problem you have to model the process to see that the cheater can often reach his goal.

The deck contains n cards. The shuffling consists of exactly t *moves*. There are p ways to make each move. The i-th way to make some move is the following. The shuffler picks $m_i + l_i$ cards from the top of the deck and moves them to the bottom of the deck. There he drops the last l_i cards picked, letting them stay at the bottom of the deck, and returns the other m_i cards to the top of the deck. The order of the cards within the blocks of l_i and m_i cards, and the cards staying in the deck remains the same.



Given the initial order of the cards in the deck, the desired order, and the parameters of the shuffle, find out whether it is possible to reorder the cards in the specified way.

Input

The first line of the input file contains n, t, and p ($2 \le n \le 36$, $1 \le t \le 15$, $1 \le p \le 5$). The next two lines contain the initial and the desired order of the cards respectively. Cards are identified by integer numbers from 1 to n, all cards are different, the cards are listed from the bottom of the deck to its top.

The following p lines contain two numbers each — m_i and l_i ($1 \le l_i < n$, $0 \le m_i < n$, $l_i + m_i < n$).

Output

If it is impossible to reoreder the cards in the required way using exactly t shuffling moves, output "`Impossible`" on the first line of the output file. In the other case output t integer numbers — in which way to perform each move.

Example

`shuffle.in`	`shuffle.out`

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

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

Problem K. Gone Swimming

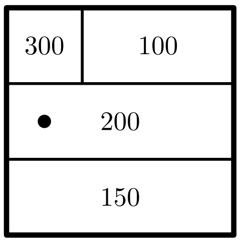
Input file:	`swimming.in`
Output file:	`swimming.out`
Time limit:	1 second
Memory limit:	64 megabytes

Macrohard head Bill Hates has recently decided to build the new swimming pool at his villa. He called his designer and asked him to design the pool. Since there are many people in Bill's family, the pool must have several sections of different depths.

Today the construction of the pool is completed. It has the form of a rectangle and consists of several rectangular sections. Sections are separated by the borders, top side of each border is on the ground level. Each section has some depth.

Now Bill wants to fill the pool with water. The water gets to the pool from the source located in some section. It brings s liters of water per minute to the pool. When the section is filled with water, the waters starts to flow to the adjacent sections. The amount of water flowing over the border is proportional to its length. When several adjacent sections are filled, they act as one section, i.e. as if there were no border between them. The water flowing over the outer border of the pool is carried away by a special system of the pipes, thus it leaves the pool.

Bill wants to know, what is the time required to fill the pool, if it's initially empty. Help him!



For example, let the pool be designed as shown on the picture, number in each section specifies its depth in centimeters, source is located in the center section of the pool. Let the amount of water getting from the source be 1000 liters per minute. The center section is filled after 6 minutes and the water starts to flow to adjacent sections. The section on the top left gets 125 liters per minute, the section on the top right gets 250 liters per minute and the section on the bottom gets 375 liters per minute. Another 250 liters per minute flow over the outer borders.

In 8 minutes the section on the top right gets filled. Now it is joined with the center section, their common border length is 10 meters, so 200 liters per minute flows to the section on the top left, and 300 to the section on the bottom. 500 liters per minute flows over the outer pool borders.

Another 5 minutes are required for the bottom section to get filled. Now the bottom section is joined with the main section and they act as the whole. The border length is 12 meters, so now $166\frac{2}{3}$ liters per minute gets to the top right section. It takes another 6 minutes to fill it completely.

Input

The first line of the input file contains n, x, y, p and s — the number of sections of the pool, its size $x \times y$ in meters, the number of the section where the water source is located, and the number of liters getting out of the source per minute ($1 \le n \le 100$, $1 \le x, y \le 100$, $1 \le s \le 100\,000$).

The following n lines contain descriptions of the sections: each section is described with five integer numbers: x_1, y_1, x_2, y_2 and d — the coordinates of its bottom-left and top-right corner in meters and its depth in centimeters ($1 \le d \le 1000$). Sections do not overlap and cover the whole pool.

Output

Output a real number — the number of minutes required to fill the whole pool with water. Your answer must be accurate up to 10^{-4}.

Example

swimming.in	swimming.out

4 3 3 2 1000
0 0 3 1 150
0 1 3 2 200
0 2 1 3 300
1 2 3 3 100
	25.000000