Problem A. Achromatic number of graph

Input file:	achromatic.in
Output file:	achromatic.out
Time limit:	2 seconds
Memory limit:	64 megabytes

Graph coloring is called *correct* if there are no vertices of the same color connected by an edge. Chromatic number of a graph is a minimal number of colors needed for correct graph coloring. Finding chromatic number of graph is known to be hard problem.

Correct graph coloring is called *achromatic* if for any pair of different colors there exist an edge between vertices of such colors. *Achromatic number* of graph is a maximal number of colors which can be used for achromatic graph coloring.

For example, achromatic number of a triangle graph is equal to three because if all vertices are colored in different colors then every pair of colors can be found on vertices connected by an edge.

You are given a number n. Find the achromatic number of a single cycle with length n and print it's achromatic coloring.

Input

Input file contains one integer number n ($3 \le n \le 1000$).

Output

First line of the output file should contain one number a — the achromatic number of single cycle graph with length n. Second line should contain n integer numbers in range from 1 to a and describe correct achromatic graph coloring.

Examples

achromatic.in	achromatic.out
3	3 1 2 3

Problem B. Binary search

Input file:	binary.in
Output file:	binary.out
Time limit:	2 seconds
Memory limit:	64 megabytes

Binary search — is an algorithm which can be used for searching a given element in a sorted array. Consider the following pseudocode of binary search algorithm («/» stands for integer division):

```
input: a[0..n - 1], x

l = 0;
r = n;
while (l < r - 1) {
    m = (l + r) / 2;
    if (a[m] <= x)
        l = m;
    else
        r = m;
}
if (a[l] == x)
    return true;
else
    return false;
```

Sometimes the binary search is capable to find an occurence of an element in an array even if it is not sorted. You are given a number n. Count the pairs $\langle a, x \rangle$ where a is an array of length n which contains integers in range from 1 to n and x is an integer number from 1 to n such that presented procedure returns "`true`" if called with a and x as arguments.

Input

Input file contains one integer number n ($1 \le n \le 1000$).

Output

Output one integer number — answer for the stated problem.

Examples

binary.in	binary.out
1	1
2	5

Problem C. Coins game

Input file:	coins.in
Output file:	coins.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Ann and Betty playing following game with coins. They have a rectangular grid of size $m \times n$ with $m \times n$ coins on the table. Some of the coins are turned upside and some are turned downside. Coins can be indexed with pairs (x, y) where $1 \le x \le m$ and $1 \le y \le n$. After that players start playing. Ann's turn is first.

Every turn, player chooses one of upside coins and flips it. Let coin in position (i, j) ($1 \le i \le m$, $1 \le j \le n$) was flipped. Then player chooses numbers i_1 and j_1 such that $0 \le i_1 < i$, $0 \le j_1 < j$ and flips coins in positions (i_1, j), (i, j_1) and (i_1, j_1) (if one of coordinates is zero player just ignore this position).

Player who can not make turn because all coins are turned downside loses.

You are given an initial state of the coins. Find who will

win if both players play opimally. If Ann wins find her first turn.

Input

First line of the input file contains m and n ($1 \le m, n \le 50$). Next m lines contain n symbols each, j-th one in i-th row equals «1» if coin in position (i, j) turned upside and «0» if it's turned downside.

Output

First line of output file should contain name of the winner: («Ann» or «Betty»). If Ann is the winner then the second line should contain i and j and the third line i_1 and j_1 which describes first turn which Ann should make to win. If there are many first winning turns print any of them.

Examples

coins.in	coins.out
3 3 000 000 001	Ann 3 3 0 0
2 2 11 01	Betty

Problem D. Cut!

Input file:	cut.in
Output file:	cut.out
Time limit:	2 seconds
Memory limit:	64 megabytes

You have a checkered paper. You need to cut this paper to pieces such that every peace contains exactly k grid cells of source paper. You can cut only by grid lines.

Input

The first line of the input file contains two integer numbers n and m ($1 \le n, m \le 100$) — height and width of paper in cells respectively. Second line contains number k ($1 \le k \le 1000$).

Output

If it is impossible to cut a given paper in described way print «-1». Otherwise, print n lines with m integer numbers each. Every number stands for a number of piece which contain this cell. Pieces should be enumerated with consecutive natural numbers starting from one.

Examples

cut.in	cut.out
2 2 1	1 2 4 3
2 2 2	1 1 2 2
2 2 3	-1
4 4 4	1 1 1 4 1 4 4 4 2 2 3 3 2 2 3 3

Problem E. Diagram

Input file:	diagram.in
Output file:	diagram.out
Time limit:	2 seconds
Memory limit:	64 megabytes

In this problem we need to build a radar chart. Suppose we have n variables with values p_i. In order to find vertices of radar chart we fix n rays from origin of coordinate system with equal angles between adjacent rays. Then on i-th ray find point with a distance to origin p_i. Rays are enumerated in counter-clockwise order.

Input

First line of input file contains one integer number n — count of variables ($3 \le n \le 100$). Second line contains n integer numbers $p_1, p_2, \ldots, p_n$ — values for variables ($1 \le p_i \le 100$).

Output

Print all vertices on the radar chart in counter-clockwise order — n lines with pairs of real numbers separated by space. First vertex should be on Ox axis and has a positive x coordinate.

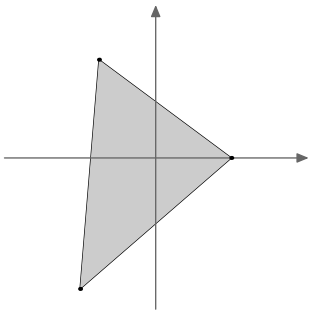
Answer considered correct if the absolute error is no more that 10^{-6}.

Examples

diagram.in	diagram.out
3 2 3 4	2.000000 0.000000 -1.500000 2.598076 -2.000000 -3.464102
3 20 30 40	20.000000 0.000000 -15.000000 25.980762 -20.000000 -34.641016
4 100 50 100 50	100 0 0 50 -100 0 0 -50

Note

First sample:



Problem F. Planet wars

Input file:	planetwars.in
Output file:	planetwars.out
Time limit:	2 seconds
Memory limit:	64 megabytes

Game «Planet Wars» gains more and more popularity. Rules of the game are simple. There are n planets on the plane. Every planet has coordinates of its location x_i, y_i and its speed of the spaceships production s_i. Two players participate in the game. At the begining, every player owns one planet and all other planets are neutral. There are p_i spaceships on the i-th planet originally. The neutral planets have neutral armies while the players planets have the corresponding player armies.

The game is played in rounds. Each round starts with an appearance of additional s_i spaceships on not neutral planets. Player who own this planet can use these spaceships on the same turn they appear.

Then players start to send their armies to other planets in such way that player can send spaceships only from the planets he owns and no more than there are spaceships on this planet. When player sends spaceships from planet with coordinates (sx, sy) to planet with coordinates (dx, dy) then spaceships will arrive to destination planet after $\lceil\sqrt{(sx-dx)^2+(sy-dy)^2}\rceil$ rounds ($\lceil a\rceil$ stands for minimal integer number no less than a).

Then spaceships arrives on planets.

At the end of the round if on any planet stays armies of both players then they start to fight. In result every player lose $x = \min(p_1, p_2)$ spaceships where p_1 — count of spaceships of first player on this planet and p_2 — of second player. If some player army is left, it fights the neutral army, if still exists, with the same rules.

After all fights if there is positive count of spaceships of some player on this planet then he starts to own this planet. Otherwise, the owner of planet doesn't change. So the round ends.

Write program to simulate game events and response on queries about i-th planet, i.e., how many spaceships on this planet and who owns this planet.

Input

The first line of the input file contains integer number n ($2 \le n \le 100$) — the number of planets. The following n lines contains four integer numbers x, y, s and p each. The numbers x and y ($|x|, |y| \le 20$) — the coordinates of the planet, s ($0 \le s \le 10$) — the speed of spaceships production, p ($0 \le p \le 100$) — the number of spaceships on the planet when game starts.

The following line contains two integer numbers — the identifiers of planets of first and second player, respectively. Planets are enumerated from one in order of appearance in input file. It is guaranteed that the coordinates of all planets are different.

The following line contains one integer number m ($0 \le m \le 1000$) — the number of queries for processing. Every query takes one line and starts with integer number t for type of query.

If $t = 0$ then it is the end of the round. All fights occur at this moment.

If $t = 1$ then sending of spaceships starts. The query consists of three positive integers *sp*, *dp* and *ships* — the identifier of a planet from which the spaceships departure, the identifier of a planet on which spaceships should arrive and the number of spaceships to send. It is guaranteed, that the planet *sp* has at least *ships* spaceships, planet *sp* is not neutral and planets *sp* and *dp* are different.

If $t = 2$ then the query consists of one integer *planet* — the identifier of a planet to get the information.

Output

For every query of $t = 2$ print in a separate line the

name of its owner and the number of spaceships at the moment of query. Follow the format of samples. All queries occur after spaceships production and before arrival of spaceships on planets.

Examples

planetwars.in	planetwars.out
3 0 0 1 100 1 0 10 10 2 0 0 100 1 3 26 1 1 2 60 1 3 2 50 2 1 2 2 2 3 0 2 1 2 2 2 3 0 2 1 2 2 2 3 1 1 2 1 2 1 2 2 2 3 0 0 2 1 2 2 2 3 1 3 2 31 0 0 2 2	Player1 41 Neutral 10 Player2 50 Player1 42 Neutral 10 Player2 50 Player1 43 Neutral 0 Player2 50 Player1 42 Neutral 0 Player2 50 Player1 44 Player1 11 Player2 50 Player2 20

Note

In sample spaceships departed in the first round will arrive to planet 2 only at the end of second round. At the end of the second round planet 2 will contain 10 neutral spaceships, 60 spaceships of first player and 50 — of second player. After fight of players spaceships there will be 10 neutral spaceships and 10 spaceships of first player. After the second fight on planet 2 the planet will be neutral and have 0 spaceships.

In the third round the first player sends one spaceship to the planet 2 and starts to own the planet at the end of fourth round.

In the fifth round the second player sends 31 spaceship to the planet 2 but when they arrive, the planet 2 already contains 31 spaceships of the first player. Spaceships destroy each other and the planet stays to be owned by the first player.

Problem G. Playlist

Input file:	playlist.in
Output file:	playlist.out
Time limit:	2 seconds
Memory limit:	64 megabytes

Vasya likes to listen music from own playlist when he does homework. Playlist consists of N songs. Every song lasts for integer number of seconds.

Initially Vasya listened songs in particular order but he got bored with it because he knew which song he will hear next. Vasya's player has «shuffle» mode. This mode changes the order of the songs to random order, but a song can not start playing second time before all song from playlist played once. Now Vasya listens to music in «shuffle» mode.

One time when Vasya started to do homework and listened music Mom asked him for help. He returned back only after T seconds. He was interested if it is possible that his favourite song will be playing at this time. Song counted as playing at moment T if it will play at least during one next second.

Input

The first line of the input file contains two integer numbers N and T ($1 \le N \le 100$, $0 \le T < \sum_{i=1}^{N} l_i$) — the number of songs in playlist and the duration of time in seconds which Vasya did not listen music.

The second line contains N integer numbers l_i ($1 \le l_i \le 500$) — the duration of a song in seconds. Vasya's favourite song is first in this list.

Output

Print «Yes» if playing of Vasya's favourite song is possible at the moment after T seconds and «No» otherwise.

Examples

playlist.in	playlist.out
3 10 3 5 7	No
4 5 2 3 4 5	Yes

Problem H. Roads

Input file:	roads.in
Output file:	roads.out
Time limit:	2 seconds
Memory limit:	64 megabytes

The road system of New-Flatcity is simple. All roads are the segments of length one with ends in points with integer coordinates.

Mayor of the city wants to build several new roads to minimise path from his home to city hall. New roads should be the segments of length one with ends in points with integer coordinates as well.

Find the minimal number of roads to build in order to achieve mayor's goal.

Input

The first line of the input file contains n — the number of roads in New-Flatcity ($0 \le n \le 100$). Then n lines follow with four integer numbers each: x_i, y_i, x_j, y_j — coordinates of ends of corresponding road ($0 \le x_i, y_i, x_j, y_j \le 100$). The last line contains two integer numbers m_x and m_y — coordinates of mayor's home ($0 \le m_x, m_y \le 100$). City hall is located at point $(0, 0)$.

All roads go either horizontally or vertically and have length one. It is possible to go in both direction on every road.

Output

Print the minimal number of roads to build in New-Flatcity.

Examples

roads.in	roads.out
1 0 0 1 0 1 1	1
5 0 0 1 0 1 0 1 1 1 1 0 1 0 1 0 2 0 2 1 2 1 2	1

Problem I. DNA

Input file:	dna.in
Output file:	dna.out
Time limit:	2 seconds
Memory limit:	64 megabytes

Deoxyribonucleic acid (DNA) — is a molecule that carries the genetic instructions used in the growth, development, functioning and reproduction of all known living organisms and many viruses.

The two DNA strands are represent two polynucleotides since. Each of which are composed of simpler monomer units called nucleotides. Each nucleotide is composed of one of four nitrogen-containing nucleobases: cytosine (C), guanine (G), adenine (A), or thymine (T). Every strand has an orientation. Two strands in DNA always have opposite orientations. So, the first nucleotide of the first strand is connected with the last nucleotide of the other strand, second with the one before last and so on.

Also, DNA satisfies the *complementary principle*: each type of the nucleobase on one strand specifies uniquely the type of the connected nucleobase on the other strand. For example, for adenine on the one strand always corresponds thymine on the other. For example, the strand AGC *complementary* to the strand GCT.

Michael knows only one strand s of his DNA but wants to know if there is a superman gene t in his DNA. DNA has gene if it contains as substring in one of it's strands.

Input

First line of the input file contains the string s with length no more than 200 characters.

Second line of the input file contains the string t with length no more than 20 characters.

Both strings contain only letters «ATGC».

Output

Print «Yes» if Michael has the superman gene or «No» otherwise.

Examples

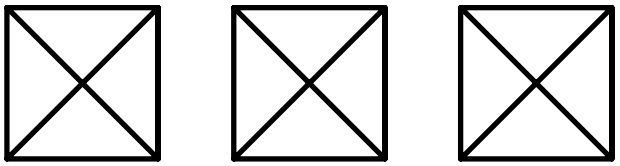
dna.in	dna.out
ATGCATGC TGC	Yes
ATGCATGC GCATGCAT	Yes
ATGCATGC TTT	No

Problem J. Simple problem

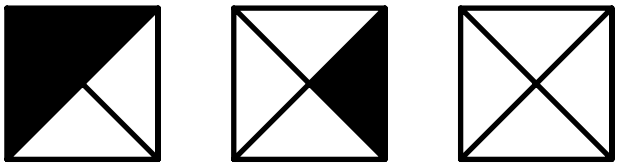
Input file:	simple.in
Output file:	simple.out
Time limit:	2 seconds
Memory limit:	64 megabytes

Vova is lazy to solve hard problems and decides to solve some simple problem.

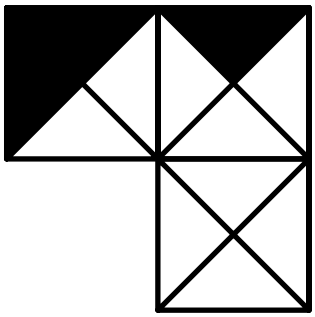
Vova asked his friend Sereja to take three square papers and draw diagonals of this squares on it.



Then Sereja colored every of four triangles to white or black color.



So the Vova's task is to write a program given the colorings of the squares will decide if it is possible to combine these squares by edges such that adjacent trianges of different squares have the same color. Notice that you can rotate squares to achieve your goal.



Input

Each of three lines of the input file contains a coloring of one square. Each coloring is described with four numbers (zero or one) — the colors of triangles corresponding upper, right, bottom and left edges of the square. You can think that zero is a white color and one is black.

Output

Print «Yes» if the squares can be combined by edges such that colors of triangles on adjacent edges are the same. Otherwise, print «No».

Examples

simple.in	simple.out
1 1 0 0 1 0 0 0 0 0 0 0	Yes

Problem K. Card trick

Input file:	trick.in
Output file:	trick.out
Time limit:	2 seconds
Memory limit:	64 megabytes

Jim loves to show card tricks.

Recently he invented a new one. There is a deck of n different card. A spectator chooses one card from the deck, remembers it, returns back and shuffles deck thoroughly.

Then magic action starts. Jim takes the shuffled deck with covers up. Then he places cards to m stacks so that the upper card in deck goes to first stack, second one — to second stack, $m+1$-st card, if there is one, goes to the first stack again, $m+2$-nd to the second and so on. After that Jim asks the spectator in which stack the card he remembered is. Let this card is in i-th stack. After that Jim collects all stack to the deck so that i-th stack is on the top of the deck, $i+1$-st under it and so on till n-th stack, after that the first stack and so on till $i-1$-st. The order in every stack is respected so that the first card placed to the stack becomes the upper one in stack, the second — under it and so on. After the repetition of this operation several times Jim tells that the remembered card is turned out to be the upper one in the deck. And it is true.

Let us see an example. Let $n = 6$ and cards are denoted by the numbers from 1 to 6 and $m = 2$. Let the remembered spectator's card is 1 and the shuffled deck is $(4, 2, 1, 5, 6, 3)$. After the first distribution to stacks we get $(4, 1, 6)$ and $(2, 5, 3)$. Then Jim collects then to the deck $(4, 1, 6, 2, 5, 3)$. Next step stacks are $(4, 6, 5)$ and $(1, 2, 3)$, after that deck is $(1, 2, 3, 4, 5, 6)$. So with magic remembered card turned to be the upper one.

The number of repetitions of the magic action depends on the remembered card and how it was shuffled. But there is minimal number k such that for every location of card in deck and every remembered card it is enough to repeat placements k times to get remembered card on the top.

Write a program which can find minimal k from n and m in input.

Input

First line of input file contains two integer numbers n and m ($2 \le m \le n \le 10^9$).

Output

Print one number k — the minimum number of the repetitions to get any remembered card on the top of the deck.

Examples

trick.in	trick.out
6 2	3
21 3	3

Problem L. Wall

Input file:	wall.in
Output file:	wall.out
Time limit:	2 seconds
Memory limit:	256 megabytes

After Edgeland was conquered by the governments of Flatland, the allies decided to divide capital to influence zones. They decided to build a wall to separate a zone of Flatland from zones of allies. The wall should be of circle shape.

Every conquerer pointed several points on map which he wanted to have in his zone. Wall should be built in such way that the points chosen by Flatland were on one side of the wall and the points chosen by allies on the other. It is ok for conquerers to have points on the wall.

Help them to build the wall.

Input

The first line of the input file contains n and m — the number of points of Flatland and allies respectively ($2 \le n, m \le 120$). Next n lines contains two integer numbers each: coordinates of points wanted by Flatland. Then m lines follow for points of allies. Coordinates are bounded by 10^4 in absolute value. All points are different.

Output

If it is possible to build a wall print «YES» on the first line. On the second line print three real numbers: coordinates of the center and radius of wall circle. Answer should be printed with the accuracy at least 10^{-6} but it's recommended to print as many digits after point as possible.

If it s impossible to build a wall print «NO» on the first line.

Examples

wall.in	wall.out
2 2 0 0 0 1 1 0 1 1	YES 0 0.5 0.5
2 2 0 0 1 1 1 0 0 1	YES 0.5 0.5 0.70710678118654752
4 2 0 0 2 2 2 0 0 2 1 1 5 5	NO