Problem A. Interactive Smiley Face

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Igor likes smiley faces a lot. He wrote a program that generates really large pictures of white and black pixels with smiley faces. Depending on Igor's mood, a picture can be a happy or a sad smiley. In the sequel white pixels are represented with 0's, and black pixels are represented with 1's.

Igor's program behaves as follows. First, a basic 41×41 rectangular pattern is fixed (patterns can be found in the archive under `Samples ZIP` tab in folder, related to this problem). Then, an odd integer $k \geq 1$ and the pattern is magnified k times. That is, the result will be a $(41 \cdot k) \times (41 \cdot k)$ rectangle, with each $k \times k$ block having the same color as the corresponding pixel of the basic pattern.

After that, a happy smile or a sad smile consisting of white pixels is put on the face (see basic patterns for smiles below). The program can choose an integer $m \geq 1$ and magnify the smile m times (magnification behaves the same way as with the face pattern). The magnified smile is then superimposed over the face in such a way that:

- All pixels of the smile should lie strictly below than the center pixel of the face (note that since $41 \cdot k$ is odd, center is defined unambigiously).
- No white pixel of the smile is put over a previously white pixel of the face pattern.
- Moreover, the border of the face should stay untouched by pixels of the smile. By border of the face we mean extreme black pixels in a row a column.

After this step, the resulting picture is placed on a $10^9 \times 10^9$ rectangular grid of initially white pixels. All pixels of the smiley face should be inside the grid. Pixels of the grid have usual integer Cartesian coordinates (x, y) ranging from 1 to 10^9. Note that the $0x$ axis is pointed rightwards, and $0y$ axis is pointed upwards, that is, the bottom pixels of the grid (and, therefore, the smiley face) have lower y coordinate.

Igor challenged you to guess the mood of his smiley face. Since the grid is very large and cannot be sent to you directly, Igor implemented a system to answer your queries. A query is a rectangle with integer coordinates of corners; the answer to the query is the number of black pixels inside the rectangle.

Your task is to guess the mood of the smiley face by asking at most 500 queries.

Interaction Protocol

Initially your program receives no input.

To ask a query, your program should print four integers x_1, y_1, x_2, y_2 ($1 \leq x_1 \leq x_2 \leq 10^9$, $1 \leq y_1 \leq y_2 \leq 10^9$) on a separate line — coordinates of the bottom left and top right corners of the rectangle respectively. Do not forget to flush your output after each query.

After each query your program is fed one integer — the number of black pixels within the rectangle.

To give the answer, your program should print -1, then at new line print "`HAPPY`" (without quotes) if smiley is happy, or "`SAD`" otherwise. Then flush your output and exit from the program.

Note

Patterns can be found in the archive under `Samples ZIP` tab. `base.dat` contains 41×41 base pattern, `happy.dat` contains happy smile pattern, `sad.dat` contains sad smile pattern, each of files `face1.txt` and `face2.txt` contains example pictures of base with added smiles.

Problem B. Tiling

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	256 mebibytes

You have a $7 \times n$ rectangular grid that you want to cover with L-triominoes and 1×1 squares. You have exactly k L-triominoes and $7n - 3k$ squares. An L-triomino can be rotated arbitrarily. Naturally, each cell of the grid should be covered by exactly one piece. Additionally, some of the cells can not be covered by a 1×1 square.

Count the number of possible tilings modulo $10^9 + 33$.

Input

The first line contains two integer n and k ($2 \le n \le 100$, $7n - 20 \le 3k \le 7n$).

The second line contains an integer m ($0 \le m \le 7n$) — the number of cells that are forbidden to cover with a square.

Each of the next m lines contains two integers x_i, y_i ($1 \le x_i \le n$, $1 \le y_i \le m$) — column and row index of the respective forbidden cell.

Output

Print the answer modulo $10^9 + 33$.

Example

standard input	standard output
2 3 9 1 1 1 2 1 3 1 5 2 1 2 2 2 3 2 4 2 5	2

Problem C. Arcade Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2.5 seconds
Memory limit:	256 mebibytes

You are playing an oldschool arcade game. The point of the game is to dodge enemy's bullets.

The screen is n cells wide and 10^6 cells high. The cells are indexed by their column index x and row index y; indices are 1-based. You control a spaceship in shape of a rectangle 1 cell wide and H cell high. Your spaceship's bottom edge is aligned with the bottom edge of the screen, that is, your ship always occupies cells $(x, 1)$, $(x, 2)$, ..., (x, H) for your current x.

All l bullets that you have to dodge are already on the screen, i-th of them is located at the center of the cell (x_i, y_i). The game consists of m rounds, each round proceeds as follows:

- First, you can move your ship one cell to the left or to the right; you are also free to stay still. You cannot move outside the screen.
- Next, all bullets move down one cell, except for the bullets in the bottom row that instead disappear.

If at any point any cell of your ship contains a bullet, you immediately lose. If you stay alive after m-th round concludes, you win.

You have played m games. In i-th of these games the starting column of your ship was a_i. You are now wondering, how many of these games you could have won should you play optimally?

Input

The first line of input contains five integers n, m, k, l, and h ($1 \le n, k, l, h \le 10^6$, $1 \le m \le 10^3$).

The second line contains k space-separated integers a_i — initial positions of the spaceships in each of your games.

Next l lines describe positions of bullets. Each of these lines contains two integers x_i and y_i — column and row index of the respective bullet ($1 \le x_i \le n$, $H + 1 \le y_i \le 10^6$).

Output

Print one integer — the answer to the problem.

Examples

standard input	standard output
3 3 2 2 2 1 2 1 3 2 3	1
6 10 7 7 3 1 2 2 3 4 5 6 6 10 2 4 2 6 3 5 4 4 5 6 6 10	6

Problem D. Sequence

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

The sequence $A(x)$ is built as follows:

- $A_1 = x$.
- $A_{n+1} = A_n \oplus \lfloor A_n/2 \rfloor$, where $\oplus$ is XOR operation.

For a given k count all integer x such that $A_{k+1} = A_1 = x$ holds. Since the answer may be very big, print it modulo $10^9 + 7$.

Input

Input contains one integer k ($1 \le k \le 10^9$).

Output

Print one integer — answer to the problem modulo $10^9 + 7$. If for the given k there are infinitely many possible x, print -1 instead.

Examples

standard input	standard output
2	3
260	15

Problem E. Eulerian Orientation

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

It is well known that an undirected graph is *eulerian* if and only if each vertex has an even degree.

Yuuka has an undirected graph with n vertices and m edges. The vertices are conveniently labeled with $1, 2, \ldots, n$. All edges are initially blue. Yuuka plans to paint some of the edges red, and leave other edges blue. If the subgraph formed by the red edges is *eulerian*, she will add x^2 to the counter, where x is the number of red edges.

Let the counter account for all 2^m ways to paint the edges. Yuuka would like to know the total value of the counter modulo $(10^9 + 7)$.

Input

The input contains zero or more test cases, and is terminated by end-of-file. For each test case:

The first line contains two integers n and m ($1 \le n \le 2 \cdot 10^5$, $0 \le m \le 2 \cdot 10^5$).

The i-th of the following m lines contains two integers a_i and b_i which denote an edge between vertices a_i and b_i ($1 \le a_i, b_i \le n$).

It is guaranteed that neither the sum of all n nor the sum of all m exceeds $2 \cdot 10^5$.

Output

For each test case, output an integer which denotes the result.

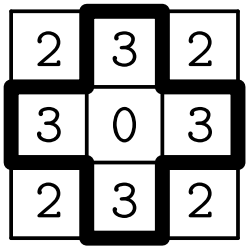
Example

standard input	standard output
4 4 1 2 1 3 1 4 2 3 6 6 1 2 2 3 3 1 4 5 5 6 6 4 2 3 1 1 1 2 1 2	9 54 14

Problem F. Tube Master II

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Yuuka is playing "Tube Master". The game field is divided into $n \times m$ cells and $(n+1) \times (m+1)$ crossings connected by $(n+1) \times m$ horizontal tubes and $n \times (m+1)$ vertical ones. The cells are conveniently labeled with (i, j) for $1 \le i \le n$, $1 \le j \le m$, and the crossings are labeled with (i, j) for $1 \le i \le (n+1)$, $1 \le j \le (m+1)$. Additionally, each cell (i, j) contains an integer $count_{i,j}$.



The above figure shows a game field with $n = m = 3$ (the third sample).

Yuuka decides to use some of the tubes. However, the game poses several weird restrictions.

1. Either 0 or 2 tubes connected to each crossing are used.
2. No two consecutive horizontal tubes are used simultaneously, and no consecutive vertical tubes are used simultaneously. Two tubes are consecutive if and only if they share the same crossing.
3. Exactly $count_{i,j}$ tubes surrounding cell (i, j) are used.

Using the tube connecting crossing (i, j) and $(i, j+1)$ costs $a_{i,j}$, and using the tube connecting crossing (i, j) and $(i+1, j)$ costs $b_{i,j}$. Yuuka would like to find a configuration satisfying the above constrains with the minimum possible total cost.

Input

The input contains zero or more test cases, and is terminated by end-of-file. For each test case:

The first line contains two integers n and m ($1 \le n, m \le 100$).

The i-th of the following n lines contains m integers $count_{i,1}, count_{i,2}, \ldots, count_{i,m}$ ($0 \le count_{i,j} \le 4$).

The i-th of the next $(n+1)$ lines contains m integers $a_{i,1}, a_{i,2}, \ldots, a_{i,m}$.

The i-th of the last n lines contains $(m+1)$ integers $b_{i,1}, b_{i,2}, \ldots, b_{i,m+1}$.

The constraints are: $1 \le a_{i,j}, b_{i,j} \le 10^9$.

It is guaranteed that the total sum of $n \cdot m$ in all test cases does not exceed 10^4.

Output

For each test case, output an integer which denotes the minimum cost of the configuration. If there is no valid configuration, output "`-1`" instead.

Example

standard input	standard output
1 3 4 2 4 1 1 1 1 1 1 1 1 1 1 1 2 3 3 1 1 1 1 1 1 1 3 3 2 3 2 3 0 3 2 3 2 1 2 3 4 5 6 7 8 9 11 12 13 1 2 3 4 5 6 7 8 9 10 11 12	8 -1 79

Problem G. Parabolas

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

There are n parabolas in the plane determined by equations $y = (x - a_i)^2 + b_i$. Parabolas cut the plane into several parts (either finite or infinite) of positive area. Your task is to count the number of those parts.

Input

The first line of the input contains one integer n ($1 \le n \le 1000$) — number of parabolas. Each of the next n lines contains two integers — coefficients a_i and b_i ($|a_i| \le 10^9, |b_i| \le 10^9$).

Output

Print one integer — the number of positive area parts that the plane is cut into.

Examples

standard input	standard output
4 0 0 1 1 0 1 2 1	10
3 2 4 3 4 2 4	4

Problem H. Flooding

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 megabytes

The Flatland Forest can be represented as a one-dimensional segment divided into regions, with either a tree or a water source in each region. Trees have positive height, each water source is located on the ground, that is, at height 0.

Sometimes a flood happens in the forest. The water starts spreading from the water sources to both sides. If $H \geq 0$ is the water level height, then the water floods all regions until it encounters a tree with height at least H or a border of the segment. A region is considered *wet* if it is flooded by at least one source. Note that the regions containing are wet even when $H = 0$.

You must answer the following queries: given L, R and H, check if at least one region between regions L and R (inclusive) is wet.

Input

First line of the input contains two integers n and m — the number of regions in the forest and number of queries, respectively ($1 \leq n, m \leq 10^6$).

Second line consists of n integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i \leq 10^9$) that describe objects in the forest. If $a_i = 0$, there is a water source at i-th region, otherwise there is a tree of height a_i.

Each of the next m lines describes one query and contains three space-separated integers L, R, H — number of leftmost and rightmost region in the segment and the water level height ($1 \leq L \leq R \leq n$, $0 \leq H \leq 10^9$).

Output

For each query print "`Yes`" if there is at least one wet region among regions L through R, and "`No`" otherwise.

Example

standard input	standard output
6 7 0 2 3 1 4 0 1 1 0 4 4 2 2 5 2 2 5 3 3 5 3 3 5 4 5 5 4	Yes No No Yes No Yes No

Problem I. Parallelogram Dissection

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Given is a strictly convex polygon, that is, it is convex and no three vertices are collinear. Check if it is possible to cut it into 2017 parallelograms.

Input

First line of the input contains one integer n ($3 \le n \le 100$).

Next n lines contains coordinates of the vertices of the polygon in clockwise order. i-th of those lines contains two integers x_i and y_i — coordinates of i-th vertice ($-10^9 \le x_i, y_i \le 10^9$).

Output

Print "`yes`", if it is possible to cut the given polygon into 2017 parallelograms, or "`no`" otherwise.

Examples

standard input	standard output
4 0 0 0 1 1 1 1 0	yes
3 0 0 0 1 1 0	no

Problem J. Build The Tree

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Consider a tree on n vertices. Each pair of vertices is connected by a simple path. Your task is to construct a tree on n vertices such that the total length of paths between all unordered pairs of vertices is exactly m. The length of a path is the number of its edges.

Input

First line of the input contains two integers n and m ($2 \le n \le 20$, $1 \le m \le 10000$).

Output

If it is impossible to construct such a tree, printf "NO".

Otherwise, print "YES" on the first line. Each of next $n-1$ lines must describe one edge and contain two integers between 1 and n — numbers of vertices connected by the corresponding edge.

If more than one correct tree exists, you may print any of them.

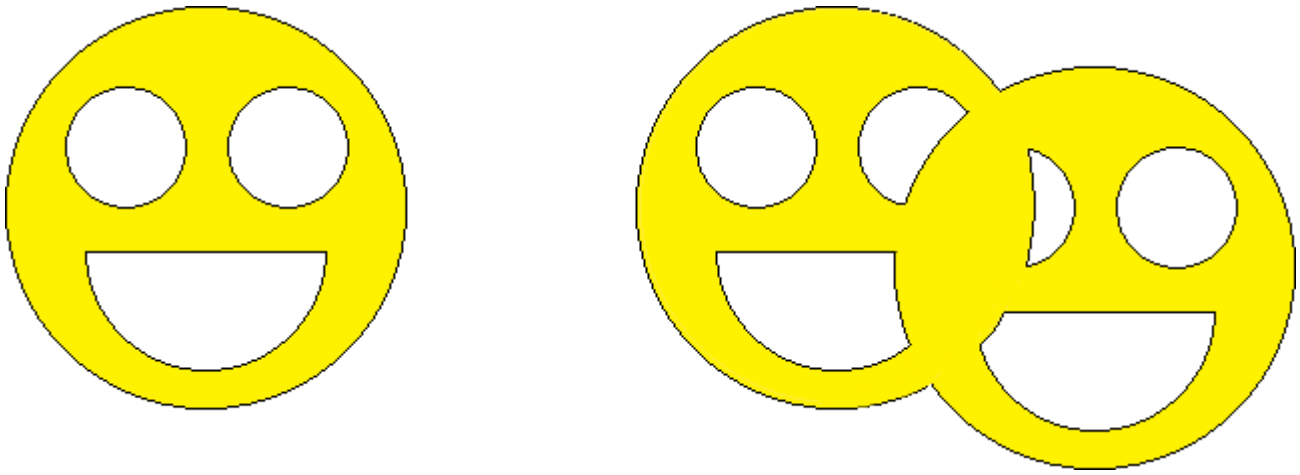
Examples

standard input	standard output
3 4	YES 1 2 2 3
3 5	NO

Problem K. Smileys Intersection

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

A *smiley* with center (x_0, y_0) is the circle of radius 100 and center (x_0, y_0), with two circular holes of radius 30 and centers $(x_0 - 40, y_0 + 30)$ and $(x_0 + 40, y_0 + 30)$, and a semicircular hole that is the lower half of the circle of radius 60 and center $(x_0, y_0 - 20)$ (look at the picture for clarity).



You are given descriptions of two smileys. Calculate area of their intersection.

Input

Input contains four space-separated integers $-1000 \le x_1,\ y_1,\ x_2,\ y_2 \le 1000$, where $(x_1,\ y_1)$ are coordinates of center of first smiley, and $(x_2,\ y_2)$ — coordinates of center of second smiley.

Output

Print area of intersection of two smileys with absolute error 10^{-4} or less.

Examples

standard input	standard output
-1000 -1000 1000 1000	40212.3859659494
0 0 0 0	20106.1929829747
0 0 -10 0	23899.0852307386