Problem A. Points

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

There are two multisets U and V that contain two-dimensional points with integer coordinates.

We will define the following function $D(U, V)$ for a pair of multisets:

- $D(U, V) = -1$ if either set is empty.
- $D(U, V) = \min\limits_{\substack{(u_x,u_y)\in U \\ (v_x,v_y)\in V}} \max(u_x + v_x, u_y + v_y)$ otherwise.

In the beginning, both U and V are empty. Process Q queries of the following form:

- "`1 s x y`": Add a point (x, y) to one of the sets. If $s = 1$, add the point to U. Otherwise, add the point to V.
- "`2 s x y`": Delete a point (x, y) from one of the sets. If $s = 1$, delete the point from U. Otherwise, delete the point from V.

When deleting a point, if there are multiple points at the given coordinates, you should delete only one of them. It is guaranteed that the given point exists in the given multiset at the time of each deletion.

Your task is to process the queries. After each query, print the value $D(U, V)$.

Input

The first line contains a single integer Q ($1 \le Q \le 250\,000$).

Each of the next Q lines contains a query in the form described above. Constraints for both types of queries: $s \in \{1, 2\}$, $0 \le x, y \le 250\,000$.

Output

Output Q lines. Each line should contain the value $D(U, V)$ after the corresponding query.

Example

standard input	*standard output*
6 1 1 100 100 1 2 30 130 1 1 120 170 2 1 100 100 1 2 70 100 2 1 120 170	-1 230 230 300 270 -1

Problem B. Bingo

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

Bingo is a game on a square grid. Each player gets an $n \times n$ grid and writes a unique number in each cell. The game host then draws a random number, and each player looks for that number on their grid and, if the number is present on their grid, fills in the corresponding cell. This repeats until someone finds n filled cells on a single line, which we will call a **bingo line**.

There are $2n + 2$ possible bingo lines: n horizontal lines, n vertical lines, and 2 diagonal lines.

```
---  ...  ...  |..  .|.  ..|  \..  ../
...  ---  ...  |..  .|.  ..|  .\.  ./.
...  ...  ---  |..  .|.  ..|  ..\  /..
```

For example, the following grid has four bingo lines: two horizontal lines, one vertical line, and one diagonal line.

```
#..#.
#####
..###
#####
..###
```

Exactly when is a bingo line formed? That is completely random: you can get a line quite early if you are lucky, but on the other hand, you can fill most of the grid without getting any bingo lines. In this problem, we investigate the unfortunate case of filling k cells without making any bingo lines.

Given two integers n and k, determine whether it is possible to fill exactly k cells in an $n \times n$ grid, without making any bingo lines. If it is possible, demonstrate how to do it.

Input

The first and only line of input contains two integers, n and k ($1 \le n \le 100$, $0 \le k \le n^2$).

Output

On the first line, output "`YES`" if it is possible to fill exactly k cells of an $n \times n$ grid without making any bingo lines. Otherwise, output "`NO`".

If the answer is "`YES`", then output n rows of the grid on the next n lines. Each row should be represented by a string of n characters. The i-th character is '`#`' (ASCII 35) if the i-th cell of the row is filled, and '`.`' (ASCII 46) if it is not filled. Exactly k cells must be filled, and there cannot be any bingo lines.

If there are multiple ways to fill the grid, then output any one of them.

Examples

standard input	*standard output*
4 2	YES ##.. ....
4 16	NO

Problem C. AND PLUS OR

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

For two nonnegative integers a, b, let $a \wedge b$ be their bitwise AND, and $a \vee b$ be their bitwise OR.

You are given an array $A_0, A_1, \ldots, A_{2^N-1}$ of length 2^N consisting of nonnegative integers. Please find a pair of indices $0 \le i, j \le 2^N - 1$ such that $A_i + A_j < A_{i \wedge j} + A_{i \vee j}$, or state that no such pair exists. If there is more than one such pair, find any one of them.

Input

The first line contains an integer N ($0 \le N \le 20$).

The second line contains 2^N integers: $A_0, A_1, \ldots, A_{2^N-1}$ ($0 \le A_i \le 10^7$).

Output

If there is an answer, output two integers i and j denoting the answer. The numbers i and j should be in the range $[0, 2^N - 1]$. Otherwise, output `-1`.

Examples

standard input	*standard output*
2 0 1 1 2	-1
2 0 1 1 3	2 1
0 100	-1

Problem D. Two Bullets

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	1024 mebibytes

There is a two-person video game about destroying buildings in a ruined city. In the game, there are N buildings over flat ground. Buildings are indexed from 1 to N from left to right. The height of building i is A_i $(1 \le i \le N)$, which is a positive integer between 1 and N. No two buildings have the same height.

Initially, both players are to the left of all buildings. At every integer moment $i(\ge 1)$, both players fire one shot each at the same time, and the bullets fly horizontally to the right from where they were fired. Both bullets have the same speed. The player selects the bullet's firing height as the distance H from the ground. The height H is an integer between 1 and $N+1$. Both players can choose the same height.

If a player's bullet firing height is H, the leftmost building that satisfies $A_i \ge H$ and has not yet been destroyed will be destroyed by this bullet. If no building satisfies this condition, nothing happens. If both players hit the same building, only that one building is destroyed. For example, if $A_1 = 2$, $A_2 = 1$, and both players initially set $H = 1$ as the firing height, building 1 will be destroyed, but building 2 will stay in place.

Given the heights of N buildings, find the minimum time to destroy all the buildings, and provide any sequence of firing heights that achieves this.

Input

The first line contains a single integer N $(1 \le N \le 100\,000)$.

The next line contains N integers $A_1, A_2, \ldots, A_N$ $(1 \le A_i \le N$, all A_i are distinct).

Output

On the first line, output a single integer T denoting the minimum time.

On each of the next T lines, output two integers denoting the firing heights of the two players. Both integers should be at least 1, and at most $N+1$. The two integers can be equal.

Examples

standard input	*standard output*
8 4 3 8 2 1 7 6 5	4 4 8 3 7 2 6 1 5
8 5 6 7 1 2 8 3 4	4 5 6 7 8 1 2 3 4
4 1 2 4 3	2 4 1 2 3

Problem E. Yet Another Interval Graph Problem

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

You are given N closed intervals. The i-th interval covers $[s_i, e_i]$ and has a positive integer weight w_i. Consider the undirected graph of N vertices, where each vertex corresponds to an interval, and there exists an edge between two vertices if and only if the corresponding pair of intervals has a nonempty intersection. For a given list of intervals, we call this graph the *interval graph*.

Jaehyun wants the graph to not have a connected component of size greater than K. To accomplish this, he can delete zero or more intervals. Jaehyun is lazy, so over all possible ways to delete intervals, he will select the way that minimizes the weight of the intervals he has to delete. Print the weight of the deleted intervals after he has made sure all connected components of the interval graph have size at most K.

Input

The first line contains two integers N and K ($1 \le K \le N \le 2500$).

Each of the next N lines contains three integers s_i, e_i, w_i ($1 \le s_i \le e_i \le 10^9$, $1 \le w_i \le 10^9$).

Output

Output the sum of the weights of the deleted intervals after Jaehyun's deletions.

Example

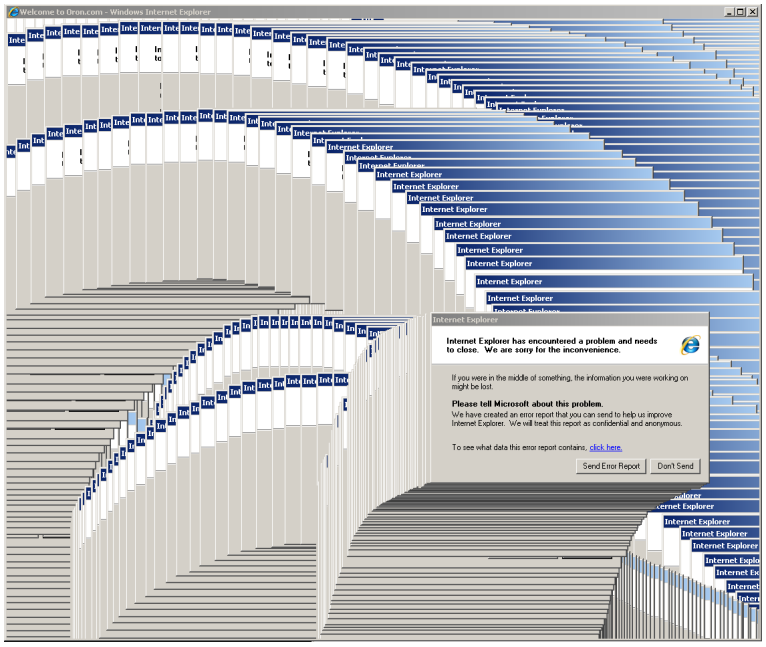
standard input	*standard output*
5 2 1 4 1 3 6 2 5 8 5 7 10 2 9 12 1	3

Note

One possible solution is to remove the second and fifth intervals.

Problem F. Lag

Input file:	*standard input*
Output file:	*standard output*
Time limit:	8 seconds
Memory limit:	1024 mebibytes



You are using Paint on an old Windows computer. The screen of Paint is a grid with cells called pixels. The coordinates of the bottom left pixel are $(1, 1)$, and the coordinates of the pixel that is in a-th column from the left and b-th row from the bottom are (a, b). On the initial screen, N rectangles with vertical and horizontal sides are drawn. A rectangle with bottom left pixel (x_1, y_1) and top right pixel (x_2, y_2) contains all pixels (x, y) such that $x_1 \le x \le x_2$ and $y_1 \le y \le y_2$.

A total of M move commands will be performed on N rectangles. The movement of the rectangle is represented by direction and distance. Each direction is one of the following: east, west, south, north, northeast, northwest, southeast, and southwest (the latter four are 45 degrees to the horizontal axis). Each distance is a positive integer d.

Suppose that the original coordinates of the bottom left pixel of the rectangle are (a, b). A movement by a distance of d in the east, north, west, and south directions causes this pixel to move toward the coordinates $(a + d, b)$, $(a, b + d)$, $(a - d, b)$, and $(a, b - d)$, respectively. In addition, a movement by a distance of d in the northeast, northwest, southwest, and southeast directions causes this pixel to move toward the coordinates $(a + d, b + d)$, $(a - d, b + d)$, $(a - d, b - d)$, and $(a + d, b - d)$, respectively.

Moving by distance d of the rectangle R on the screen is implemented by quickly displaying the shape of R every time when R moves by distance 1. However, our computer is very old, so moving R is very laggy. As a result, all of the R drawn in the movement of R remains on the screen. Therefore, if R moves by the distance d, d rectangles are newly created on the screen. For example, if the rectangle moves in the northeast direction by a distance of 3, 3 rectangles are created, leaving a total of 4 rectangles on the screen. Of course, after moving, the rectangle at the northeast end becomes R.

After executing M move commands, Q queries will be given. Each query is given as a pixel p on the plane. Print the number of rectangles containing the pixel p as the answer to the query.

Input

The first line contains three integers N, M, and Q ($1 \le N \le 250\,000$, $0 \le M \le 250\,000$, $1 \le Q \le 250\,000$).

Each of the next N lines contains four integers x_1, y_1, x_2, and y_2, denoting a rectangle with bottom left pixel (x_1, y_1) and top right pixel (x_2, y_2) ($1 \le x_1 \le x_2 \le 250\,000$, $1 \le y_1 \le y_2 \le 250\,000$).

Each of the next M lines contains three integers v_i, x_i, and d_i, denoting that the x_i-th rectangle moved in the direction v_i by distance d_i ($0 \le v_i \le 7$, $1 \le x_i \le N$, $1 \le d_i \le 250\,000$).

The directions are:

- 0: $(+1, 0)$
- 1: $(+1, +1)$
- 2: $(0, +1)$
- 3: $(-1, +1)$
- 4: $(-1, 0)$
- 5: $(-1, -1)$
- 6: $(0, -1)$
- 7: $(+1, -1)$

Each of the next Q lines contains two integers x and y, denoting the query on pixel (x, y).

All coordinates are positive integers between 1 and 250 000. Any pixels contained in a rectangle at any time satisfy these constraints. Queried pixels also satisfy these constraints.

Output

For each queried pixel, output a single integer denoting the number of rectangles containing the given pixel.

Examples

standard input	*standard output*
1 8 3 2 1 2 1 0 1 1 1 1 1 2 1 1 3 1 1 4 1 1 5 1 1 6 1 1 7 1 1 1 1 2 1 4 2	0 2 1
2 0 3 3 3 7 7 4 4 6 6 5 5 3 7 8 8	2 1 0

Problem G. Critical Vertex

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	1024 mebibytes

You are given a connected undirected graph G with N vertices and M edges where vertices are numbered from 1 to N and edges are numbered from 1 to M.

For each vertex v, if the graph is disconnected after the vertex v is removed, we call v a *critical* vertex. Note that the original connectedness of the graph does not matter.

For each i ($1 \le i \le M$), please compute the number of critical vertices in G when edge i is removed. Note that the removal is only temporary, and **does not** affect other queries.

Input

The first line contains two integers N and M ($2 \le N \le 250\,000$, $1 \le M \le 1\,000\,000$).

In the next M lines, edges of the graph will be given. The i-th line contains two integers x_i and y_i, denoting the edge i connecting vertex x_i and vertex y_i ($1 \le x_i, y_i \le N, x_i \ne y_i$). The graph **may have** multiple edges. The graph is connected.

Output

Output M lines. On the i-th line, output a single integer denoting the number of critical vertices in G when edge i is removed.

Example

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

Problem H. Endless Road

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	1024 mebibytes

Whenever students of KAIST go outside of campus, they usually follow a straight road named the *Endless Road*. The name comes from the fact that most students feel like the road is *endless*. One factor for this is due to the boring scenery of the road. Here, we model the Endless Road as a straight line of length 10^9. The position over the straight line can be denoted as a single real number $x \in [0, 10^9)$.

To cope with this, the members of RUN@KAIST decided to plant different kinds of colorful flowers along this road. The N members of RUN@KAIST are numbered from 1 to N. In the last regular meeting, each member was assigned a nonempty segment $[L_i, R_i)$ over the straight line, containing all positions x such that $L_i \le x < R_i$.

The members with higher indices bear more responsibility for the tasks on the club. Thus, **the length of the segments is nondecreasing** as the indices of the members increase. In other words, $R_i - L_i \le R_j - L_j$ for $1 \le i < j \le N$.

The planting is done in N turns. In each turn, we select one previously unselected member to plant flowers. The selected member plants the flowers in their assigned segment $[L_i, R_i)$, except where other members have already planted flowers before.

To make the work distribution more equitable, the selection is done in the following way:

- Pick the member who will plant the minimum number of flowers. Here, the number of flowers is determined by the total length of the intervals where this member will plant new flowers, which could be zero.
- If there is more than one such member, pick the member with minimum index.

Jaeung should now announce the planting schedule. For this, he has to find an order in which the N members will plant the flowers, according to the above selection rules. Please help him do it.

Input

The first line contains an integer N ($1 \le N \le 250\,000$).

The next N lines each contain two integers L_i and R_i ($0 \le L_i < R_i \le 10^9$, $R_i - L_i \le R_j - L_j$ for all $1 \le i < j \le N$).

All intervals are distinct. In other words, $(L_i, R_i) \ne (L_j, R_j)$ for all $1 \le i < j \le N$.

Output

Output N integers denoting the order of planting. The i-th integer denotes the index of the member who will plant flowers in the i-th turn.

Examples

standard input	*standard output*
6 1 2 2 3 3 4 4 5 1 3 3 5	1 2 5 3 4 6
4 3 7 10 14 1 6 6 11	1 3 2 4

Problem I. Streetlights

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	1024 mebibytes

There are N streetlights along the straight road. The initial height of the i-th streetlight is a positive integer A_i ($1 \le i \le N$).

You are trying to install an electric wire between two streetlights. To install an electric wire between the streetlights i and $j (> i)$, the following conditions must be satisfied:

- $A_i = A_j$,
- For every $i < k < j$, $A_k < A_i$.

The city may adjust the height of a streetlight Q times. Each adjustment is given as a pair of positive integers (x, h), which indicates that the height of x-th streetlight was adjusted to h. In other words, $A_x = h$.

Before the updates, and also after each update, find the number of pairs $1 \le i < j \le N$ such that you can install an electric wire between streetlights i and j.

Input

The first line contains two integers N and Q ($2 \le N \le 100\,000$, $1 \le Q \le 250\,000$).

The next line contains N integers $A_1, A_2, \ldots, A_N$ ($1 \le A_i \le 10^9$).

Each of the next Q lines contains two integers x and h, denoting that $A_x = h$ after the query ($1 \le x \le N$, $1 \le h \le 10^9$). It is guaranteed that h is different from the height of the x-th streetlight immediately prior to the requested update.

Output

Output $Q + 1$ lines. On the i-th line ($1 \le i \le Q + 1$), output the number of pairs you can install an electric wire between after processing the first $i - 1$ update queries.

Example

standard input	*standard output*
6 2 4 2 2 2 4 6 4 6 6 4	3 2 2

Problem J. Diameter Pair Sum

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	1024 mebibytes

For an unweighted tree T, a simple path P is a *diameter* if there is no simple path longer than it. Two paths are different if some vertex is in one path but not the other.

Consider a set of paths D_T where $P \in D_T$ if and only if P is a diameter. Given two paths D and E, let $f(D, E)$ be the number of vertices that belong to both D and E.

You are given an undirected forest (a graph with no cycles) with N vertices and M edges. Process Q queries of the following form:

- "`1` x y": Connect two vertices x and y with an edge ($1 \le x, y \le N$). It is guaranteed that there is no path between x and y at the time of the query.
- "`2` x y": Remove an edge between two vertices x and y ($1 \le x, y \le N$). It is guaranteed that such an edge exists at the time of the query.
- "`3` x": Let F be the connected component containing the vertex x. Output the value $\sum_{D \in D_F} \sum_{E \in D_F} f(D, E)$ modulo $10^9 + 7$ ($1 \le x \le N$).

Input

The first line of the input consists of three integers N, M, and Q ($2 \le N \le 100\,000$, $0 \le M \le N - 1$, $1 \le Q \le 100\,000$).

Each of the next M lines consists of two integers x and y denoting an edge connecting vertices x and y ($1 \le x, y \le N$, $x \ne y$). It is guaranteed that there are no cycles in the given graph.

Each of the next Q lines contains a query in the form described above.

Output

For each query of type 3, output the answer modulo $10^9 + 7$.

Example

standard input	*standard output*
7 5 5 1 2 1 3 2 4 2 5 3 6 3 1 1 3 7 3 1 2 2 1 3 1	18 64 21

Problem K. Fake Plastic Trees 2

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

You are given a tree with N vertices numbered from 1 to N. The tree is vertex-weighted. In other words, each vertex of the tree is assigned a nonnegative integer weight.

We will delete some edges from the tree. After the deletion, for each connected component, the sum of vertex weights should be in the range $[L, R]$.

For all integers $0 \le i \le K$, determine if we can achieve this goal by deleting **exactly** i edges.

Input

The first line contains a single integer T, the number of test cases. Then T test cases follow, each following the given specification:

The first line of each test case contains four integers N, K, L, and R ($1 \le N \le 1000$, $0 \le K \le \min(50, N-1)$, $0 \le L \le R \le 10^{18}$).

The next line contains N integers $A_1, A_2, \ldots, A_N$, where A_i denotes the weight of vertex i ($0 \le A_i \le 10^{18}$).

Each of the next $N-1$ lines contains two integers x, y, denoting the pair of vertices connected by an edge ($1 \le x, y \le N$, $x \ne y$). It is guaranteed that the given graph is a tree.

For all test cases, the sum of N is at most 1000.

Output

For each test case, output a binary string of length $K+1$. The i-th character should be '1' if it is possible to achieve the desired goal by deleting exactly $i-1$ edges. Otherwise, the i-th character should be '0'.

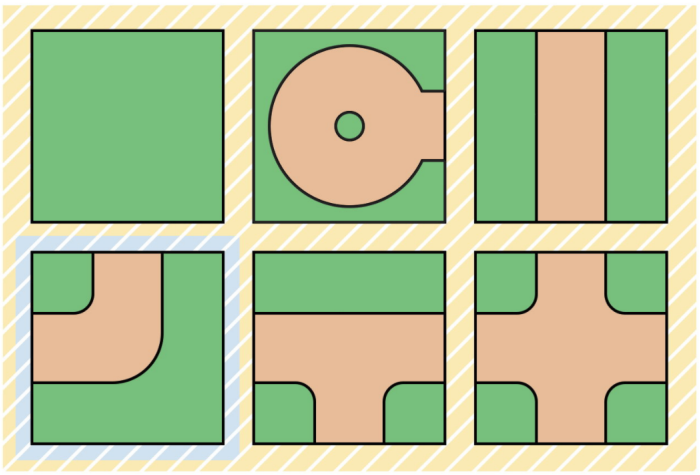
Example

standard input	*standard output*
3 4 3 1 2 1 1 1 1 1 2 2 3 3 4 4 3 1 2 1 1 1 1 1 2 1 3 1 4 4 3 0 0 1 1 1 1 1 2 1 3 1 4	0111 0011 0000

Problem L. Curly Racetrack

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	1024 mebibytes

You are creating a new *KartRider* racetrack. The racetrack is a $H \times W$ rectangular board. Each cell of the board is either empty, or contains a tile which is described in the picture below.



Note that each of the tiles can be rotated, and all except one have roads going out of the tile. We denote the fourth type of tile (highlighted in blue color) as a *curly* tile.

In a valid *KartRider* racetrack, every cell should contain a tile, and there should be no dead-ends over the roads in the tile. In other words, each road going out of the tile should be connected to a road on the neighboring tile. Also, the road should not face outside the board.

Currently, there are some curly tiles already placed on the board. You can place zero or more **curly** tiles on the board and submit the racetrack. Then, the administrator will try to fill all empty cells with tiles (not necessarily curly) with their best effort, so that it becomes a valid *KartRider* racetrack. Note that you can only place the curly tiles, and you should not remove, replace, or rotate the curly tiles already placed in the board.

Additionally, for some cells, you can not place the tiles, and for some cells, you must place the curly tiles.

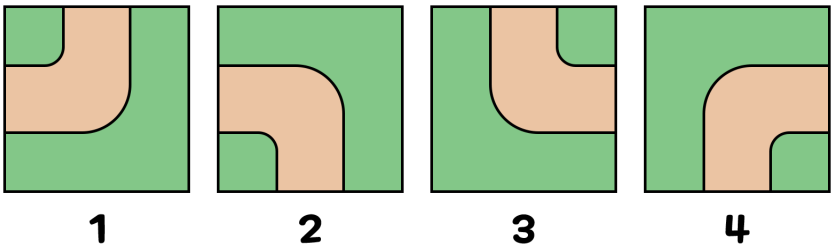
Please find the maximum possible number of curly tiles you can place in the racetrack so that the administrator can fill the remaining cells to create a valid *KartRider* racetrack. You should report the total number of curly tiles in the racetrack, not the number of curly tiles you have newly added. If there is no way to create a valid *KartRider* racetrack, print -1.

Input

The first line contains two integers H and W denoting the size of the board. ($1 \le H, W \le 100$)

The next H lines contain the information about the board. Each line consists of W characters, denoting the following:

- "`1`": Curly tile containing road connecting upper and left cells,
- "`2`": Curly tile containing road connecting lower and left cells,
- "`3`": Curly tile containing road connecting upper and right cells,
- "`4`": Curly tile containing road connecting lower and right cells,
- "`o`": Empty cell where curly tile must be placed,
- "`x`": Empty cell where you can not place the tile,
- "`.`": Empty cell which has no restrictions.



Output

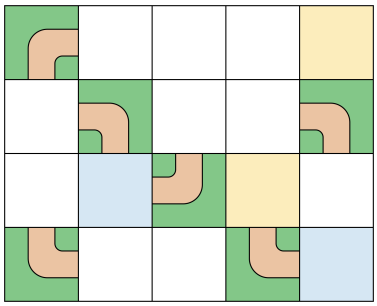
Output the maximum possible number of curly tiles you can place on the racetrack so that the administrator can fill the remaining cells to create a valid *KartRider* racetrack. You should report the total number of curly tiles on the racetrack, not the number of curly tiles you have newly added. If there is no way to create a valid *KartRider* racetrack, print -1.

Examples

standard input	*standard output*
4 5 4...x .2..2 .o1x. 3..3o	12
2 3 4o2 3x1	-1

Note

The following image illustrates the original state of the board in the first sample input.



The following image illustrates the final state of the board in the optimal solution. Tiles placed by the administrators are highlighted in yellow.