Codeforces Round #144 (Div. 1)

A. Cycles

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

John Doe started thinking about graphs. After some thought he decided that he wants to paint an undirected graph, containing exactly k cycles of length 3.

A cycle of length 3 is an unordered group of three distinct graph vertices a, b and c, such that each pair of them is connected by a graph edge.

John has been painting for long, but he has not been a success. Help him find such graph. Note that the number of vertices there shouldn't exceed 100, or else John will have problems painting it.

Input

A single line contains an integer k ($1 \le k \le 10^5$) — the number of cycles of length 3 in the required graph.

Output

In the first line print integer n ($3 \le n \le 100$) — the number of vertices in the found graph. In each of next n lines print n characters "0" and "1": the i-th character of the j-th line should equal "0", if vertices i and j do not have an edge between them, otherwise it should equal "1". Note that as the required graph is undirected, the i-th character of the j-th line must equal the j-th character of the i-th line. The graph shouldn't contain self-loops, so the i-th character of the i-th line must equal "0" for all i.

Examples

input
1

output
3 011 101 110

input
10

output
5 01111 10111 11011 11101 11110

B. Table

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

John Doe has an $n \times m$ table. John Doe can paint points in some table cells, not more than one point in one table cell. John Doe wants to use such operations to make each square subtable of size $n \times n$ have exactly k points.

John Doe wondered, how many distinct ways to fill the table with points are there, provided that the condition must hold. As this number can be rather large, John Doe asks to find its remainder after dividing by 1000000007 $(10^9 + 7)$.

You should assume that John always paints a point exactly in the center of some cell. Two ways to fill a table are considered distinct, if there exists a table cell, that has a point in one way and doesn't have it in the other.

Input

A single line contains space-separated integers n, m, k ($1 \le n \le 100$; $n \le m \le 10^{18}$; $0 \le k \le n^2$) — the number of rows of the table, the number of columns of the table and the number of points each square must contain.

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

In a single line print a single integer — the remainder from dividing the described number of ways by 1000000007 $(10^9 + 7)$.

Examples

input
5 6 1
output
45

Note

Let's consider the first test case:

The gray area belongs to both 5×5 squares. So, if it has one point, then there shouldn't be points in any other place. If one of the white areas has a point, then the other one also must have a point. Thus, there are about 20 variants, where the point lies in the gray area and 25 variants, where each of the white areas contains a point. Overall there are 45 variants.

C. Doe Graphs

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

John Doe decided that some mathematical object must be named after him. So he invented the Doe graphs. The Doe graphs are a family of undirected graphs, each of them is characterized by a single non-negative number — its order.

We'll denote a graph of order k as $D(k)$, and we'll denote the number of vertices in the graph $D(k)$ as $|D(k)|$. Then let's define the Doe graphs as follows:

- $D(0)$ consists of a single vertex, that has number 1.
- $D(1)$ consists of two vertices with numbers 1 and 2, connected by an edge.
- $D(n)$ for $n \geq 2$ is obtained from graphs $D(n - 1)$ and $D(n - 2)$. $D(n - 1)$ and $D(n - 2)$ are joined in one graph, at that numbers of all vertices of graph $D(n - 2)$ increase by $|D(n - 1)|$ (for example, vertex number 1 of graph $D(n - 2)$ becomes vertex number $1 + |D(n - 1)|$). After that two edges are added to the graph: the first one goes between vertices with numbers $|D(n - 1)|$ and $|D(n - 1)| + 1$, the second one goes between vertices with numbers $|D(n - 1)| + 1$ and 1. Note that the definition of graph $D(n)$ implies, that $D(n)$ is a connected graph, its vertices are numbered from 1 to $|D(n)|$.

The picture shows the Doe graphs of order 1, 2, 3 and 4, from left to right.

John thinks that Doe graphs are that great because for them exists a polynomial algorithm for the search of Hamiltonian path. However, your task is to answer queries of finding the shortest-length path between the vertices a_i and b_i in the graph $D(n)$.

A path between a pair of vertices u and v in the graph is a sequence of vertices $x_1, x_2, \ldots, x_k$ $(k > 1)$ such, that $x_1 = u$, $x_k = v$, and for any i $(i < k)$ vertices x_i and x_{i+1} are connected by a graph edge. The length of path $x_1, x_2, \ldots, x_k$ is number $(k - 1)$.

Input

The first line contains two integers t and n $(1 \leq t \leq 10^5;\ 1 \leq n \leq 10^3)$ — the number of queries and the order of the given graph. The i-th of the next t lines contains two integers a_i and b_i $(1 \leq a_i, b_i \leq 10^{16},\ a_i \neq b_i)$ — numbers of two vertices in the i-th query. It is guaranteed that $a_i, b_i \leq |D(n)|$.

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

Output

For each query print a single integer on a single line — the length of the shortest path between vertices a_i and b_i. Print the answers to the queries in the order, in which the queries are given in the input.

Examples

input
10 5 1 2 1 3 1 4 1 5 2 3 2 4 2 5 3 4 3 5 4 5
output
1 1 1 2 1 2 3 1 2 1

D. Fence

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

John Doe has a crooked fence, consisting of n rectangular planks, lined up from the left to the right: the plank that goes i-th $(1 \le i \le n)$ (from left to right) has width 1 and height h_i. We will assume that the plank that goes i-th $(1 \le i \le n)$ (from left to right) has index i.

A *piece of the fence* from l to r $(1 \le l \le r \le n)$ is a sequence of planks of wood with indices from l to r inclusive, that is, planks with indices $l, l + 1, \ldots, r$. The width of the piece of the fence from l to r is value $r - l + 1$.

Two pieces of the fence from l_1 to r_1 and from l_2 to r_2 are called matching, if the following conditions hold:

- the pieces do not intersect, that is, there isn't a single plank, such that it occurs in both pieces of the fence;
- the pieces are of the same width;
- for all i $(0 \le i \le r_1 - l_1)$ the following condition holds: $h_{l_1 + i} + h_{l_2 + i} = h_{l_1} + h_{l_2}$.

John chose a few pieces of the fence and now wants to know how many distinct matching pieces are for each of them. Two pieces of the fence are distinct if there is a plank, which belongs to one of them and does not belong to the other one.

Input

The first line contains integer n $(1 \le n \le 10^5)$ — the number of wood planks in the fence. The second line contains n space-separated integers $h_1, h_2, \ldots, h_n$ $(1 \le h_i \le 10^9)$ — the heights of fence planks.

The third line contains integer q $(1 \le q \le 10^5)$ — the number of queries. Next q lines contain two space-separated integers l_i and r_i $(1 \le l_i \le r_i \le n)$ — the boundaries of the i-th piece of the fence.

Output

For each query on a single line print a single integer — the number of pieces of the fence that match the given one. Print the answers to the queries in the order, in which the queries are given in the input.

Examples

input
10 1 2 2 1 100 99 99 100 100 100 6 1 4 1 2 3 4 1 5 9 10 10 10
output
1 2 2 0 2 9

E. Quick Tortoise

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

John Doe has a field, which is a rectangular table of size $n \times m$. We assume that the field rows are numbered from 1 to n from top to bottom, and the field columns are numbered from 1 to m from left to right. Then the cell of the field at the intersection of the x-th row and the y-th column has coordinates $(x; y)$.

We know that some cells of John's field are painted white, and some are painted black. Also, John has a tortoise, which can move along the white cells of the field. The tortoise can get from a white cell with coordinates $(x; y)$ into cell $(x + 1; y)$ or $(x; y + 1)$, if the corresponding cell is painted white. In other words, the turtle can move only along the white cells of the field to the right or down. The turtle can not go out of the bounds of the field.

In addition, John has q queries, each of them is characterized by four numbers x_1, y_1, x_2, y_2 ($x_1 \le x_2$, $y_1 \le y_2$). For each query John wants to know whether the tortoise can start from the point with coordinates $(x_1; y_1)$, and reach the point with coordinates $(x_2; y_2)$, moving only along the white squares of the field.

Input

The first line contains two space-separated integers n and m ($1 \le n, m \le 500$) — the field sizes.

Each of the next n lines contains m characters "#" and ".": the j-th character of the i-th line equals "#", if the cell $(i; j)$ is painted black and ".", if it is painted white.

The next line contains integer q ($1 \le q \le 6 \cdot 10^5$) — the number of queries. Next q lines contain four space-separated integers x_1, y_1, x_2 and y_2 ($1 \le x_1 \le x_2 \le n$, $1 \le y_1 \le y_2 \le m$) — the coordinates of the starting and the finishing cells. It is guaranteed that cells $(x_1; y_1)$ and $(x_2; y_2)$ are white.

Output

For each of q queries print on a single line "Yes", if there is a way from cell $(x_1; y_1)$ to cell $(x_2; y_2)$, that meets the requirements, and "No" otherwise. Print the answers to the queries in the order, in which the queries are given in the input.

Examples

input
3 3 ... .## .#. 5 1 1 3 3 1 1 1 3 1 1 3 1 1 1 1 2 1 1 2 1
output
No Yes Yes Yes Yes

input
5 5 ..... .###. ..... .###. 5 1 1 5 5 1 1 1 5 1 1 3 4 2 1 2 5 1 1 2 5
output
Yes Yes Yes No Yes