Codeforces Round #238 (Div. 1)

A. Unusual Product

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

Little Chris is a huge fan of linear algebra. This time he has been given a homework about the *unusual square* of a square matrix.

The *dot product* of two integer number vectors x and y of size n is the sum of the products of the corresponding components of the vectors. The *unusual square* of an $n \times n$ square matrix A is defined as the sum of n dot products. The i-th of them is the dot product of the i-th row vector and the i-th column vector in the matrix A.

Fortunately for Chris, he has to work only in $GF(2)$! This means that all operations (addition, multiplication) are calculated modulo 2. In fact, the matrix A is binary: each element of A is either 0 or 1. For example, consider the following matrix A:

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

The unusual square of A is equal to $(1 \cdot 1 + 1 \cdot 0 + 1 \cdot 1) + (0 \cdot 1 + 1 \cdot 1 + 1 \cdot 0) + (1 \cdot 1 + 0 \cdot 1 + 0 \cdot 0) = 0 + 1 + 1 = 0$.

However, there is much more to the homework. Chris has to process q queries; each query can be one of the following:

1. given a row index i, flip all the values in the i-th row in A;
2. given a column index i, flip all the values in the i-th column in A;
3. find the unusual square of A.

To flip a bit value w means to change it to 1 - w, i.e., 1 changes to 0 and 0 changes to 1.

Given the initial matrix A, output the answers for each query of the third type! Can you solve Chris's homework?

Input

The first line of input contains an integer n ($1 \le n \le 1000$), the number of rows and the number of columns in the matrix A. The next n lines describe the matrix: the i-th line contains n space-separated bits and describes the i-th row of A. The j-th number of the i-th line a_{ij} ($0 \le a_{ij} \le 1$) is the element on the intersection of the i-th row and the j-th column of A.

The next line of input contains an integer q ($1 \le q \le 10^6$), the number of queries. Each of the next q lines describes a single query, which can be one of the following:

- 1 i — flip the values of the i-th row;
- 2 i — flip the values of the i-th column;
- 3 — output the unusual square of A.

Note: since the size of the input and output could be very large, don't use slow output techniques in your language. For example, do not use input and output streams (cin, cout) in C++.

Output

Let the number of the 3rd type queries in the input be m. Output a single string s of length m, where the i-th symbol of s is the value of the unusual square of A for the i-th query of the 3rd type as it appears in the input.

Sample test(s)

input

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

```
2 1
1 1
3
```

output

```
01001
```

B. Toy Sum

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

Little Chris is very keen on his toy blocks. His teacher, however, wants Chris to solve more problems, so he decided to play a trick on Chris.

There are exactly s blocks in Chris's set, each block has a unique number from 1 to s. Chris's teacher picks a subset of blocks X and keeps it to himself. He will give them back only if Chris can pick such a non-empty subset Y from the remaining blocks, that the equality holds:

$$\sum_{x \in X}(x - 1) = \sum_{y \in Y}(s - y)$$

"Are you kidding me?", asks Chris.

For example, consider a case where $s = 8$ and Chris's teacher took the blocks with numbers 1, 4 and 5. One way for Chris to choose a set is to pick the blocks with numbers 3 and 6, see figure. Then the required sums would be equal: $(1 - 1) + (4 - 1) + (5 - 1) = (8 - 3) + (8 - 6) = 7$.

However, now Chris has exactly $s = 10^6$ blocks. Given the set X of blocks his teacher chooses, help Chris to find the required set Y!

Input

The first line of input contains a single integer n ($1 \le n \le 5 \cdot 10^5$), the number of blocks in the set X. The next line contains n distinct space-separated integers $x_1, x_2, \ldots, x_n$ ($1 \le x_i \le 10^6$), the numbers of the blocks in X.

Note: since the size of the input and output could be very large, don't use slow output techniques in your language. For example, do not use input and output streams (cin, cout) in C++.

Output

In the first line of output print a single integer m ($1 \le m \le 10^6 - n$), the number of blocks in the set Y. In the next line output m distinct space-separated integers $y_1, y_2, \ldots, y_m$ ($1 \le y_i \le 10^6$), such that the required equality holds. The sets X and Y should not intersect, i.e. $x_i \ne y_j$ for all i, j ($1 \le i \le n$; $1 \le j \le m$). It is guaranteed that at least one solution always exists. If there are multiple solutions, output any of them.

Sample test(s)

input
3 1 4 5
output
2 999993 1000000

input
1 1
output
1 1000000

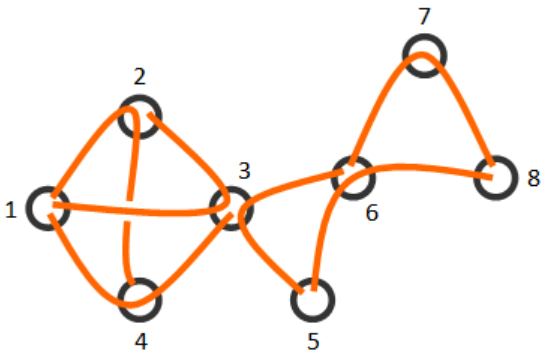
C. Graph Cutting

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Little Chris is participating in a graph cutting contest. He's a pro. The time has come to test his skills to the fullest.

Chris is given a simple undirected connected graph with n vertices (numbered from 1 to n) and m edges. The problem is to cut it into edge-distinct paths of length 2. Formally, Chris has to partition all edges of the graph into pairs in such a way that the edges in a single pair are adjacent and each edge must be contained in exactly one pair.

For example, the figure shows a way Chris can cut a graph. The first sample test contains the description of this graph.



You are given a chance to compete with Chris. Find a way to cut the given graph or determine that it is impossible!

Input

The first line of input contains two space-separated integers n and m ($1 \le n, m \le 10^5$), the number of vertices and the number of edges in the graph. The next m lines contain the description of the graph's edges. The i-th line contains two space-separated integers a_i and b_i ($1 \le a_i, b_i \le n$; $a_i \ne b_i$), the numbers of the vertices connected by the i-th edge. It is guaranteed that the given graph is simple (without self-loops and multi-edges) and connected.

Note: since the size of the input and output could be very large, don't use slow output techniques in your language. For example, do not use input and output streams (cin, cout) in C++.

Output

If it is possible to cut the given graph into edge-distinct paths of length 2, output $\frac{m}{2}$ lines. In the i-th line print three space-separated integers x_i, y_i and z_i, the description of the i-th path. The graph should contain this path, i.e., the graph should contain edges (x_i, y_i) and (y_i, z_i). Each edge should appear in exactly one path of length 2. If there are multiple solutions, output any of them.

If it is impossible to cut the given graph, print "`No solution`" (without quotes).

Sample test(s)

input

```
8 12
1 2
2 3
3 4
4 1
1 3
2 4
3 5
3 6
5 6
6 7
6 8
7 8
```

output

```
1 2 4
1 3 2
1 4 3
5 3 6
5 6 8
6 7 8
```

input

```
3 3
1 2
2 3
3 1
```

output

```
No solution
```

input

```
3 2
1 2
2 3
```

output

```
1 2 3
```

D. Hill Climbing

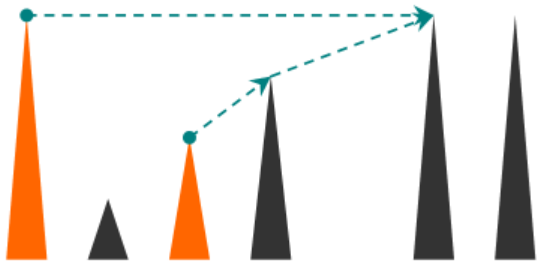
time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem has nothing to do with Little Chris. It is about hill climbers instead (and Chris definitely isn't one).

There are n hills arranged on a line, each in the form of a vertical line segment with one endpoint on the ground. The hills are numbered with numbers from 1 to n from left to right. The i-th hill stands at position x_i with its top at height y_i. For every two hills a and b, if the top of hill a can be seen from the top of hill b, their tops are connected by a rope. Formally, the tops of two hills are connected if the segment connecting their top points does not intersect or touch any of the other hill segments. Using these ropes, the hill climbers can move from hill to hill.

There are m teams of climbers, each composed of exactly two members. The first and the second climbers of the i-th team are located at the top of the a_i-th and b_i-th hills, respectively. They want to meet together at the top of some hill. Now, each of two climbers move according to the following process:

1. if a climber is at the top of the hill where the other climber is already located or will come eventually, the former climber stays at this hill;
2. otherwise, the climber picks a hill to the right of his current hill that is reachable by a rope and **is the rightmost possible**, climbs this hill and continues the process (the climber can also climb a hill whose top is lower than the top of his current hill).



For each team of climbers, determine the number of the meeting hill for this pair!

Input

The first line of input contains a single integer n ($1 \le n \le 10^5$), the number of hills. The next n lines describe the hills. The i-th of them contains two space-separated integers x_i, y_i ($1 \le x_i \le 10^7$; $1 \le y_i \le 10^{11}$), the position and the height of the i-th hill. The hills are given in the ascending order of x_i, i.e., $x_i < x_j$ for $i < j$.

The next line of input contains a single integer m ($1 \le m \le 10^5$), the number of teams. The next m lines describe the teams. The i-th of them contains two space-separated integers a_i, b_i ($1 \le a_i, b_i \le n$), the numbers of the hills where the climbers of the i-th team are located. It is possible that $a_i = b_i$.

Output

In a single line output m space-separated integers, where the i-th integer is the number of the meeting hill for the members of the i-th team.

Sample test(s)

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

E. Hamming Triples

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

Little Chris is having a nightmare. Even in dreams all he thinks about is math.

Chris dreams about m binary strings of length n, indexed with numbers from 1 to m. The most horrifying part is that the bits of each string are ordered in either ascending or descending order. For example, Chris could be dreaming about the following 4 strings of length 5:

00011
00000
11110
11111

The *Hamming distance* $H(a, b)$ between two strings a and b of length n is the number of positions at which the corresponding symbols are different.

Chris thinks that each three strings with different indices constitute a single triple. Chris's delusion is that he will wake up only if he counts the number of such string triples a, b, c that the sum $H(a, b) + H(b, c) + H(c, a)$ is maximal among all the string triples constructed from the dreamed strings.

Help Chris wake up from this nightmare!

Input

The first line of input contains two space-separated integers n and m ($1 \le n \le 10^9$; $3 \le m \le 10^5$), the length and the number of strings. The next m lines contain the description of the strings. The i-th line contains two space-separated integers s_i and f_i ($0 \le s_i \le 1$; $1 \le f_i \le n$), the description of the string with index i; that means that the first f_i bits of the i-th string are equal to s_i, and the remaining n - f_i bits are equal to 1 - s_i. There can be multiple equal strings in Chris's dream.

Output

Output a single integer, the number of such string triples among the given that the sum of the Hamming distances between the strings of the triple is maximal.

Sample test(s)

input
5 4 0 3 0 5 1 4 1 5
output
3

input
10 4 1 5 0 5 0 5 1 5
output
4