CSE221 Assignment 04 Summer 2025

A. Adjacency Matrix Representation

time limit per test: 1 second
memory limit per test: 256 megabytes

You are given a **directed weighted** graph with ***N*** nodes and ***M*** edges. The nodes are numbered from 1 to N. Each edge represents a direct connection between two nodes. There is no self loop or multi edge.

Input

The first line contains two integers N and M $(1 \le N \le 100, 0 \le M \le \frac{N(N-1)}{2})$ — the number of vertices and the total number of edges.

The next M lines will contain three integers $u_i, v_i, w_i (1 \le u_i, v_i \le N, 1 \le w_i \le 1000)$ — denoting there is an edge from node u_i to v_i with cost w_i.

Output

The output consists of an $N \times N$ adjacency matrix representing the directed weighted graph. Each row corresponds to a node, and each column represents its directed edges to other nodes. The value at position (i, j) denotes the weight of the edge from node i to node j. If there is no edge, the value is 0.

Examples

input

```
6 7
1 5 6
6 3 5
1 3 9
3 4 7
4 6 1
5 6 8
6 1 6
```

output

```
0 0 9 0 6 0
0 0 0 0 0 0
0 0 0 7 0 0
0 0 0 0 0 1
0 0 0 0 0 8
6 0 5 0 0 0
```

input

```
4 3
1 3 8
3 2 5
1 4 2
```

output

```
0 0 8 2
0 0 0 0
0 5 0 0
0 0 0 0
```

B. Adjacency List Representation

time limit per test: 1 second
memory limit per test: 256 megabytes

You are given a **directed weighted** graph with ***N*** nodes and ***M*** edges. The nodes are numbered from 1 to N. Each edge represents a direct connection between two nodes. There is no self loop or multi edge.

Input

The first line contains two integers N and M $(1 \le N \le 100, 0 \le M \le \frac{N(N-1)}{2})$ — the number of vertices and the total number of edges.

The second line contains M integers $u_1, u_2, u_3 \dots u_m$ $(1 \le u_i \le N)$ — where the i-th integer represents the node that is one endpoint of the i-th edge.

The third line contains M integers $v_1, v_2, v_3 \dots v_m$ $(1 \le v_i \le N)$ — where the i-th integer represents the node that is other endpoint of the i-th edge.

Thr fourth line contains M integers $w_1, w_2, w_3 \dots w_m$ $(1 \le w_i \le 1000)$ — where the i-th integer represents the weight of the i-th edge.

The i'th edge of this graph is from the i'th node in the second line to the i'th node in the third line, their weight is the i'th value in the fourth line.

Output

For the given input, the output should be the Adjacency List representation of the graph as shown in the sample output.

Examples

input

```
4 5
4 1 4 3 3
3 2 2 2 1
4 4 10 8 5
```

output

```
1: (2,4)
2:
3: (2,8) (1,5)
4: (3,4) (2,10)
```

input

```
4 4
3 3 2 4
2 1 1 3
9 5 8 10
```

output

```
1:
2: (1,8)
3: (2,9) (1,5)
4: (3,10)
```

C. Graph Metamorphosis

time limit per test: 1 second
memory limit per test: 256 megabytes

You are given a **directed unweighted** graph with ***N*** nodes in an adjacency list format. The nodes are numbered from 0 to N-1. Your task is to convert it into an adjacency matrix representation.

Input

The first line contains a integer N $(1 \le N \le 100)$ — the number of vertices.

The next N lines describe the adjacency list:

1. The i-th line starts with an integer k, indicating the number of nodes adjacent to node i.
2. The next k space-separated integers represent the nodes adjacent to node i.
3. Nodes are numbered from 0 to N-1.

Output

Print an N×N adjacency matrix, where the cell at row i and column j

- 1 if there is an edge between nodes i and j
- 0 otherwise.

Examples

input

```
5
2 1 2
1 0
1 0
1 4
1 3
```

output

```
0 1 1 0 0
1 0 0 0 0
1 0 0 0 0
0 0 0 0 1
0 0 0 1 0
```

input

```
5
0
2 2 3
3 1 3 4
2 1 2
1 2
```

output

```
0 0 0 0 0
0 0 1 1 0
0 1 0 1 1
0 1 1 0 0
0 0 1 0 0
```

D. The Seven Bridges of Königsberg

time limit per test: 1 second
memory limit per test: 256 megabytes

You are given an **undirected unweighted connected** graph with ***N*** nodes and ***M*** edges. There can be self loop or multiple edges. Your task is to determine whether an Eulerian Path exists in the graph.

In graph theory, an Eulerian path (also called an Eulerian trail or Eulerian walk) is a path in a graph that visits every edge exactly once and may start and end at different vertices. However, a vertex can be visited multiple times.

Input

The first line contains two integers N and M $(1 \le N \le 2 \times 10^5, 0 \le M \le 3 \times 10^5)$ — the number of vertices and the total number of edges.

The second line contains M integers $u_1, u_2, u_3 \dots u_m$ $(1 \le u_i \le N)$ — where the i-th integer represents the node that is one endpoint of the i-th edge.

The third line contains M integers $v_1, v_2, v_3 \dots v_m$ $(1 \le v_i \le N)$ — where the i-th integer represents the node that is other endpoint of the i-th edge.

The i'th edge of this graph is between the i'th node in the second line and the i'th node in the third line.

Output

If an Eulerian Path exists, print YES. Otherwise, print NO.

Examples

input

```
5 10
5 5 5 2 2 2 3 3 4 2
2 3 1 3 4 1 4 1 2 4
```

output

```
YES
```

input

```
5 4
1 4 3 2
4 3 2 5
```

output

```
YES
```

input

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

output

```
NO
```

input

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

output

```
YES
```

E. Edge Queries

time limit per test: 1 second
memory limit per test: 256 megabytes

You are given a **directed unweighted** graph with ***N*** nodes and ***M*** edges. The nodes are numbered from 1 to N. Your task is to find the difference of indegree and outdegree of each node in the graph.

Input

The first line contains two integers N and M $(1 \le N \le 2 \times 10^5, 0 \le M \le 3 \times 10^5)$ — the number of vertices and the total number of edges.

The second line contains M integers $u_1, u_2, u_3 \dots u_m$ $(1 \le u_i \le N)$ — where the i-th integer represents the node that is one endpoint of the i-th edge.

The third line contains M integers $v_1, v_2, v_3 \dots v_m$ $(1 \le v_i \le N)$ — where the i-th integer represents the node that is other endpoint of the i-th edge.

The i-th edge of this graph is from the i-th node in the second line to the i-th node in the third line.

Output

Output a single line with N space-separated integers, where the i-th integer is the difference of indegree and outdegree of node i.

Examples

input

```
5 10
2 5 4 3 2 4 3 4 1 3
5 1 5 5 1 2 2 1 3 4
```

output

```
2 0 -2 -2 2
```

input

```
5 4
5 3 3 2
1 1 2 4
```

output

```
2 0 -2 1 -1
```

input

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

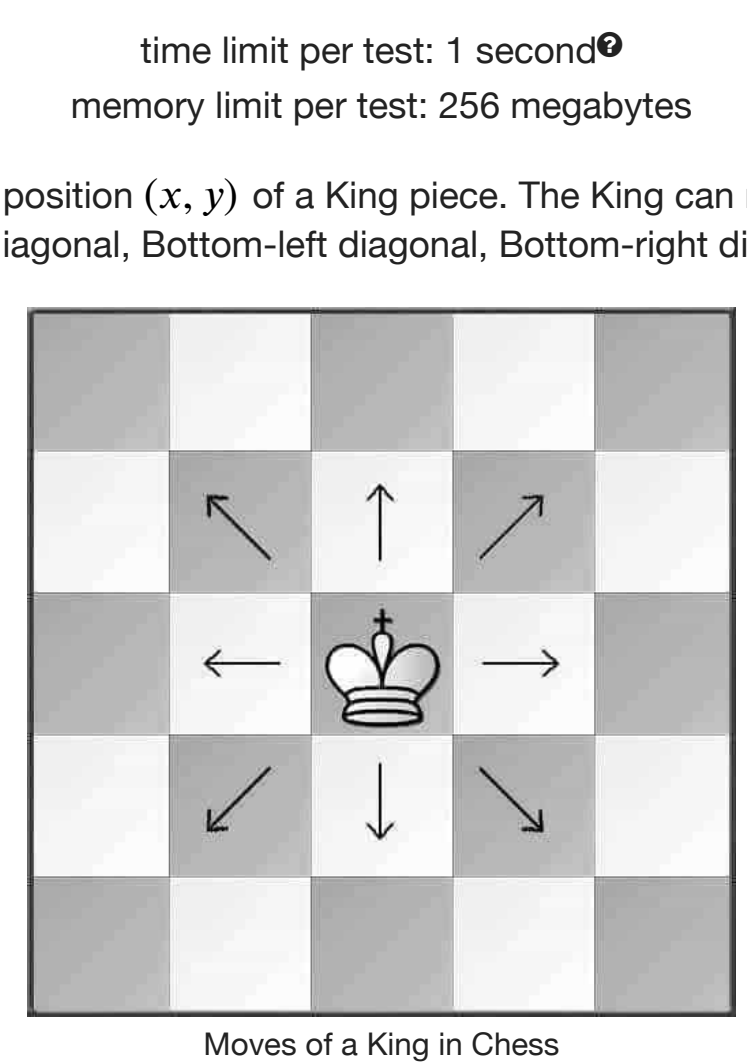
output

```
-2 1 1 0 1 1 -3 1
```

F. The King of Königsberg

time limit per test: 1 second
memory limit per test: 256 megabytes

You are given an $N * N$ chessboard and the initial position (x, y) of a King piece. The King can move one step in any of the 8 possible directions: Up, Down, Left, Right, Top-left diagonal, Top-right diagonal, Bottom-left diagonal, Bottom-right diagonal.



Moves of a King in Chess

Your task is to determine the number of valid moves the King can make in one move. A move is valid if it remains inside the board.

Input

The first line contains an integer $(1 \le N \le 2 \times 10^5)$ — the size of the chessboard.

The second line contains two integers $(1 \le x, y \le N)$ — the initial position of the King on the chessboard.

Output

First, print an integer K — the number valid moves the King can make in one move.

Next, print K lines, each containing two integers representing a valid move in ascending order. A move (a, b) is smaller than (c, d) if $a < c$ or if $a = c$ and $b < d$.

Examples

input

```
8
1 1
```

output

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

input

```
8
1 2
```

output

```
5
1 1
1 3
2 1
2 2
2 3
```

input

```
8
2 2
```

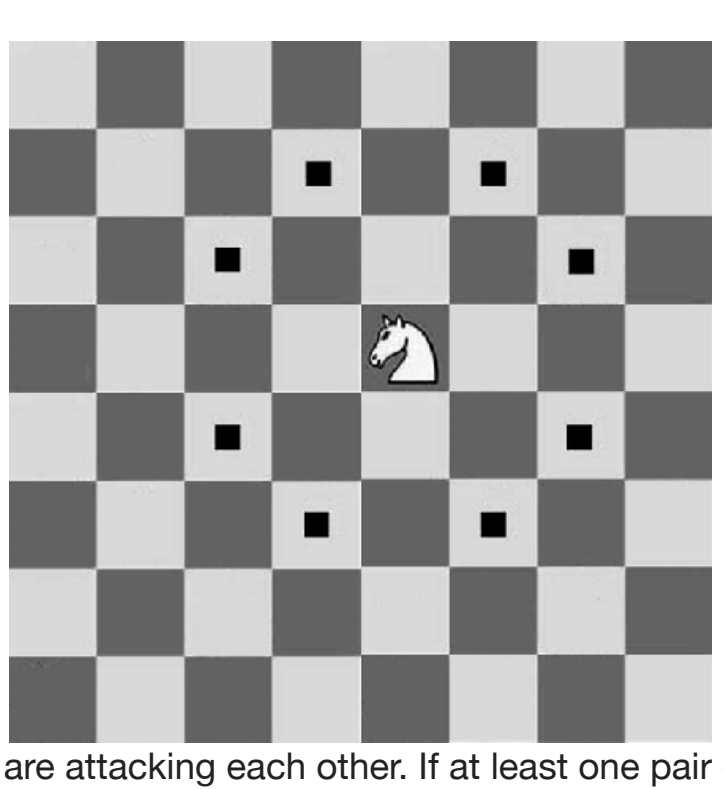
output

```
8
1 1
1 2
1 3
2 1
2 3
3 1
3 2
3 3
```

G. The Knights of Königsberg

time limit per test: 1 second
memory limit per test: 256 megabytes

You are given a chessboard of size $N \times M$. There are K knights placed on distinct cells of this board. The standard movement of a knight in chess is as follows: it moves in an "L" shape—either two squares in one direction and then one square perpendicular to that, or one square in one direction and then two squares perpendicular to that. A knight can jump over other pieces.



Your task is to determine whether any pair of knights are attacking each other. If at least one pair of knights are in positions such that one can attack the other in a single move, print "YES". Otherwise, print "NO".

Input

The first line contains three integers N, M, and K $(1 \le N, M \le 1000, 0 \le K \le 10^5)$ — the number of rows, the number of columns, and the number of knights on the board.

Each of the next K lines contains two integers x_i, y_i $(1 \le x_i \le N, 1 \le y_i \le M)$ — the position of the i-th knight on the board.

It is guaranteed that all K knights are placed on distinct positions.

Examples

input

```
6 4 4
2 3
4 2
1 2
3 3
```

output

```
YES
```

input

```
4 4 4
1 1
1 3
2 2
2 4
```

output

```
NO
```

H. Coprime Graph

time limit per test: 2 seconds
memory limit per test: 256 megabytes

You are given an integer N. Construct an undirected graph with N nodes, where each node i is connected to all node j such that $gcd(i, j) = 1$ where $1 \le i, j \le N$ and $i \ne j$.

For example,for $N = 6$, the graph will be, $G = [[2,3,4,5,6],[1,3,5],[1,2,4,5],[1,3,5],[1,2,3,4,6],[1,5]]$.

Now, there will be Q queries. Each query consists of two integers X and K. For each query, you have to determine the $K-th$ smallest node connected to node X.

Input

The first line contains two integers N and Q $(1 \le N \le 2 \times 10^3, 1 \le Q \le 3 \times 10^5)$ — the number of vertices and the total number of queries.

The next Q lines contain two integers X and K $(1 \le X \le N, 1 \le K \le 10^6)$, representing a query.

Output

For each query, output the $K-th$ smallest node connected to node X. If there are fewer than K neighbors of X, then print -1.

Examples

input

```
5 6
1 3
3 1
4 2
5 5
3 4
5 2
```

output

```
4
1
3
-1
5
2
```

input

```
2000 3
903 24
702 563
942 50
```

output

```
41
1829
149
```

input

```
1 1
1 1
```

output

```
-1
```

input

```
2 1
2 1
```

output

```
1
```

Note

Explanation of the First Sample (Let's go through the queries):

Query (1, 3): The neighbors of node 1 are [2, 3, 4, 5]. Sorted: [2, 3, 4, 5]. The 3rd smallest is 4. Output: 4.

Query (3, 1): The neighbors of node 3 are [1, 2, 4, 5]. Sorted: [1, 2, 4, 5]. The 1st smallest is 1. Output: 1.

Query (4, 2): The neighbors of node 4 are [1, 3, 5]. Sorted: [1, 3, 5]. The 2nd smallest is 3. Output: 3.

Query (5, 5): The neighbors of node 5 are [1, 2, 3, 4]. There are only 4 neighbors, so the 5th smallest does not exist. Output: -1.

Query (3, 4): The neighbors of node 3 are [1, 2, 4, 5]. Sorted: [1, 2, 4, 5]. The 4th smallest is 5. Output: 5.

Query (5, 2): The neighbors of node 5 are [1, 2, 3, 4]. Sorted: [1, 2, 3, 4]. The 2nd smallest is 2. Output: 2.