Task: LUC
Luck of the Paw

AACPP SuSe 2025 Round 5 Memory: 256MiB 2025.06.17 – 2025.06.24

The neighbourhood cats love to play a unique feline card game – Pawker. Pawker is played 1v1, with one cat always being the clear winner. The cats treat the game very seriously. Every night a cat that wins the most pawker games against other cats gets the nickname "Felix"[1] for the next day. In case of a tie, the cats vote who was the luckiest among the top scorers.

Last night, Dexter earned this nickname for the first time. However, after running around after squirrels this morning he completely forgot how many games he won. He remembers who played against whom how many times, but not the outcome of any of the games.

Since this is a very momentous occasion, Dexter would like to find out how many games he had to win and scratch it down somewhere for posterity. Moreover, he'd like to make sure he got the number correct and determine outcomes of all matches that would make his win possible. Since Dexter just ran off to chase another squirrel, it's up to you to solve his problem!

Input

The first line of input contains two integers, n and m, the number of cats and the number of pawker rounds played. Dexter doesn't remember which number he is.

The next m lines describe the rounds. Each line contains two integers, $1 \leq x_i, y_i \leq n$, meaning cats x_i and y_i played against each other. Cats don't play themselves ($x_i \neq y_i$), however **the same pair of cats can play multiple rounds between each other**.

Output

The first line of output should contain a single integer – the minimal number of pawker rounds that Dexter had to win to be elected the Felix.

The next m lines should contain the description of a possible set of outcomes of all rounds. If in the i-th round x_i won against y_i, then the line should contain the single number 1; otherwise, if y_i won against x_i, it should contain the single number 0. There might be more than one valid set of outcomes – your program may output any of them.

Examples

For the input:

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

a correct output is:

```
1
0
0
0
1
```

Wins (i-th row is the number of wins of i against all others)

	1	**2**	**3**	**4**	**Σ**
1		1	0	0	1
2	1				1
3	1				1
4	1				1

[1] Meaning "lucky" in an old language used by ancient Roman cats.

One game was enough to become Felix this night. In this assignment every cat won exactly one game. One could flip the assignment of round 1 and 4 between cats 1 and 2 and get a valid assignment as well.

For the input:

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

Wins (i-th row is the number of wins of i against all others)

	1	**2**	**3**	**4**	**Σ**
1		2	0	0	2
2	0		1	1	2
3	1	0		0	1
4		0	1		1

a correct output is:

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

There is no set of outcomes where only 1 win would suffice to become the Felix. There are other valid assignments for 2 wins, for example 011001.

Additional examples

The following initial tests are also available:

- `0c` – $n = 10$, $m = 10$, cats $1 \leq i < 10$ played exactly one game with $i + 1$, and 10 played with 1 once;
- `0d` – $n = 100$, $m = 990$, 1 played ten games with everyone else;
- `0e` – $n = 10\,000$, $m = 9\,998$, for $1 \leq i < 5\,000$ the cat $2i$ played two games with cat $2i - 1$;
- `0f` – $n = 10\,000$, $m = 199\,980$, for $1 \leq i < 10\,000$ played twenty games with cat $i + 1$.

Limits

Your solution will be evaluated on a number of hidden test cases divided into groups. Points for a group are awarded if and only if the submission returns the correct answer for each of the tests in the group within the allotted time limit. These groups are organised into subtasks with the following limits and points awarded.

Partial points

If your solution outputs the optimal number of wins (first line of output), and the other lines are left blank or are not correct, it will receive 50% of the points for a given test group.

Subtask	Limits	Points
1.	$1 \leq n \leq 100$, $1 \leq m \leq 1\,000$	2
2.	$1 \leq n \leq 10\,000$, $1 \leq m \leq 10\,000$	4
3.	$1 \leq n \leq 10\,000$, $1 \leq m \leq 200\,000$	4