C. Game

Two players play the following game. Initially, the players have a knife and a rectangular sheet of paper, divided into equal square grid cells of unit size. The players make moves in turn, the player who can't make a move loses. In one move, a player can take the knife and cut the paper along any segment of the grid line (not necessarily from border to border). The part of the paper, that touches the knife at least once, is considered cut. There is one limit not to turn the game into an infinite cycle: each move has to cut the paper, that is the knife has to touch the part of the paper that is not cut before.

Obviously, the game ends when the entire sheet is cut into 1×1 blocks. During the game, the pieces of the sheet are not allowed to move. It is also prohibited to cut along the border. The coordinates of the ends of each cut must be integers.

You are given an $n \times m$ piece of paper, somebody has already made k cuts there. Your task is to determine who will win if the players start to play on this sheet. You can consider that both players play optimally well. If the first player wins, you also need to find the winning first move.

Input

The first line contains three integers n, m, k ($1 \leq n, m \leq 10^9, 0 \leq k \leq 10^5$) — the sizes of the piece of paper and the number of cuts. Then follow k lines, each containing 4 integers xb_i, yb_i, xe_i, ye_i ($0 \leq xb_i, xe_i \leq n, 0 \leq yb_i, ye_i \leq m$) — the coordinates of the ends of the existing cuts.

It is guaranteed that each cut has a non-zero length, is either vertical or horizontal and doesn't go along the sheet border.

The cuts may intersect, overlap and even be the same. That is, it is not guaranteed that the cuts were obtained during any correct game.

Output

If the second player wins, print "SECOND". Otherwise, in the first line print "FIRST", and in the second line print any winning move of the first player (the coordinates of the cut ends, follow input format to print them).

Examples

input

```
2 1 0
```

output

```
FIRST
1 0 1 1
```

input

```
2 2 4
0 1 2 1
0 1 2 1
1 2 1 0
1 1 1 2
```

output

```
SECOND
```