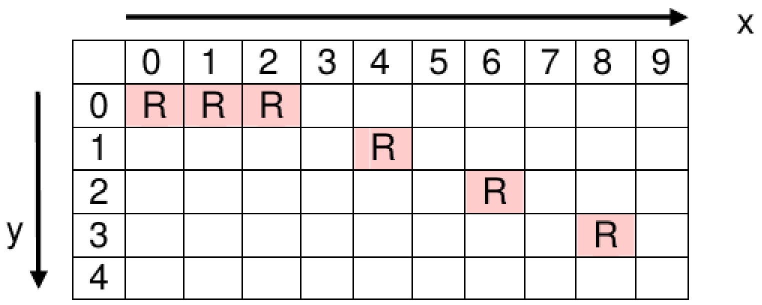
2735 Spots

Painting tiles is hard work. John has convinced his two sons Raul and George to do a painting job, by agreeing to pay one dollar per tile painted by each of them. Raul will paint red tiles and George green tiles at the places indicated by their father. However, they are still not convinced: how are they going to divide up the area to be painted, what happens if both want to paint the same tile at the same time, what are the rules, and, last, but not least, how much will they receive in the end?

To avoid arguments and to have some fun, John would like to show them a simulation of the problem before the hard work begins. The simulation will start with each son at opposite ends of a rectangular grid of width N and height M ($N, M \geq 2$), Raul at (0,0) and George at ($N-1, M-1$). The two sons will begin by painting a spot in their respective colours on their starting tiles. Next, Raul and George are each given a series of individual instructions for moving to the next tile to paint. Each move can be repeated one or more times and is defined by a pair of increments along the horizontal x-axis and vertical y-axis, in this order; these increments can be positive, negative, or zero.

For example, assume that Raul is on his initial tile (0,0) on a grid with $N = 10$, $M = 5$, and receives the instructions to hop 2 times in the direction of (1,0) and then 3 times in the direction of (2,1). He will begin by painting a red spot on the tile (0,0). Then, according to these instructions, he will successively land on and paint red spots on the following tiles: (1,0), (2,0), (4,1), (6,2), (8,3). The following diagram uses the letter 'R' to mark the tiles spotted in red by Raul according to his instructions:



At each simulation step the two sons move in lock-step, each hopping according to his own instructions. To avoid conflicts, each time Raul and George are about to hop to the next tile the simulation must check whether they would land on the same tile. If so, the simulation ends without them moving, and the landing tile remains painted in its previous colour, if any. Otherwise, the simulation will end as soon as one of the sons ends his instructions.

Each tile can contain only one colour spot, either red or green. Since it is possible that Raul and George land on the same tile at different times, the simulation should only count the tile towards the son landing there last.

To avoid Raul and George falling off the edge of the tile grid, they are allowed to wrap around, in all directions. For example, for a grid of the same size as above, one hypothetical move from (8,3) in the direction (2,3) results in the tile (0,1).

Your task is to write a program that computes the results of such a simulation given the grid size and the instructions for each son.

Input

The input consists of one or more scenarios. Each scenario consists of 3 lines. The first line contains two numbers separated by a space, N and M, $2 \leq N, M \leq 1000$, respectively representing the width and the height of the grid. The second line contains the instructions for Raul and the third line the instructions for George. Each instruction line starts with a number C, in the range 1 to 100, followed by C groups of 3 numbers. In each group the first number is a repetition count in the range 1 to $1,000,000,000$, the second number is an increment along the x-axis, and the third number an increment along the y-axis — both increments are in the range $-1,000$ to 1,000. All numbers are separated by single spaces. The end of the input is indicated by a "grid of size 0", i.e., a '0' on a line by itself.

Note: The input data for this program may contain lines up to 2000 characters long.

Output

Output one line for each input scenario. Each output line should show two numbers separated by a single space, representing the earnings of Raul and George, in this order.

Sample Input

```
10 5
2 2 1 0 3 2 1
3 4 -2 0 1 0 -9 2 1 0
10 10
2 1 15 5 1 0 0
2 1 0 0 2 -4 -4
10 7
2 1000 2 -1 1000 0 -1
2 1000 -1 0 1000 -1 0
10 10
5 1000 2 -1 1000 1 0 2 5 5 3 1 1 10 1 1
5 1000 -1 0 1000 0 1 3 -4 -4 2 -1 -1 10 -1 -1
0
```

Sample Output

```
5 6
2 1
30 10
18 18
```