Go Play Wall-nut Bowling!

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	512 mebibytes

The famous computer game "Plants vs. Zombies" has a Wall-nut Bowling mode.

In this mode, the zombies move from right to left, and the player can launch wall-nuts at them, which bounce off the zombies on impact. Your task is to simulate the Wall-nut bowling game.

Please note that the rules used in this problem are **different** from the rules of the original game!

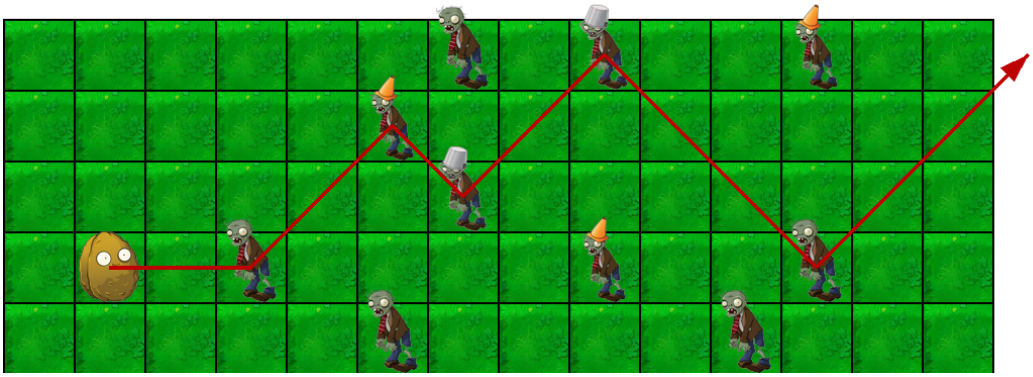
The game takes place on a playfield of size $M \times N$ unit cells. At any given time, there may be zombies in some cells of the field, with no more than one zombie in each cell.

The player can launch the wall-nut from any cell of the field. Immediately after launch, the nut rolls horizontally to the right until the first encounter with the zombies.

After each collision with a zombie, the nut rebounds and continues to move diagonally: either in the right-up or in the right-down direction.

If this collision is not the first, then the wall-nut continues to move along a different diagonal: if it rolled from the bottom-left, then it bounces down-right, and vice versa. If this is the first collision, then one of the two diagonals is chosen depending on how the player launched the nut.

The movement of the nut ends when it rolls out of the field to the right, down or up.



Thus, the nut draws a zigzag on the field. If the player launches a nut from the cell that has a zombie in it, then the nut immediately hits that zombie and bounces off of it. In this problem, we will assume that the nut is rolling very fast, and the start of the nut takes zero time.

All zombies move from right to left at unit speed, moving one square every second. When a zombie leaves the field, it enters your house and queues up to eat your brains, and does not participate in the game anymore.

In addition, new zombies appear in the game from time to time.

The only way to protect yourself from zombies is to beat them with wall-nuts!

Each zombie has an armour level. Each time a wall-nut hits a zombie, its armour level is reduced by one.

If, due to the impact, the armour level is reduced to zero, then this zombie collapses, i.e. completely disappears from the game.

The initial field is empty.

Your task is to perform Q queries of the following three types:

- Wait t seconds of time, letting the zombie to move to the left and/or enter the house.
- Launch a wall-nut from a given cell, hitting and possibly destroying some zombies.
- Add a new zombie in the given cell.

Launching a wall-nut and adding zombies is done in a negligible amount of time, however, all requests must be performed strictly in the given sequence.

Input

The first line of the input contains three integers: M is the number of the rows, N is the number of columns, and Q is the number of the queries ($1 \le M, N \le 10^7$, $1 \le Q \le 3 \cdot 10^5$).

The following Q lines contain the queries, in order, one query per line. Each query consists of the command description, followed by some integers.

time t — wait t seconds ($1 \le t \le 10^7$). You shall print one integer — the number of zombies that left the playfield and joined the queue to eat your brains for that time interval.

deliver r c s — shoot the nut. Here r and c — the row and the column of the starting cell for the nut ($1 \le r \le M$, $1 \le c \le N$), and s determines the direction of bounce: -1 for right-up direction and 1 for right-down direction. You shall print two integers — the number of bounces and the number of zombies destroyed by this shoot.

add r c h — add the zombie. Here r and c are the row and the column of the cell where a zombie is added ($1 \le r \le M$, $1 \le c \le N$), and h is the initial level of the armour ($1 \le h \le 10^6$).

When the nut moves down, the row index is increasing, when the nut moves right, the column index is increasing.

You may assume that a zombie will never be added to the cell with another zombie. You may assume that the sum of t in the `time` queries does not exceed 10^9.

Output

In the first line of the output print the text `stdout ducks` because the zombies in this game use the ducks.

Then print the answers on the `time` and `deliver` queries, one answer per a line.

Example

standard input	standard output
6 8 23 add 2 4 2 add 3 7 1 add 4 6 3 add 5 3 3 add 6 4 3 deliver 1 1 1 deliver 5 1 1 deliver 5 1 -1 deliver 2 4 1 deliver 5 1 1 time 2 add 4 4 3 deliver 2 1 1 deliver 6 1 -1 time 7 add 2 3 1 add 2 6 1 add 3 3 1 add 4 4 1 deliver 2 5 1 add 3 7 10 deliver 3 2 1 time 5	stdout ducks 0 0 3 0 1 0 3 1 3 2 0 2 1 2 1 1 1 1 2 2 1