MATERIALIZER

Chef Christopher just made his dream come true. He opened a company that produces custom-made cookie cutters. First customers already placed their orders (not all of the shapes are suitable for cookie cutters, but oh well – customer is always right).

Christopher bought a special machine – Materializer 3000. It can produce any given three-dimensional shape, you have to program it beforehand though.

Every order consists of **a** layers. Every layer is a table of cubes with edge length 1, with **b** rows and **c** columns. For every cube customer specifies whether he wants it filled or not. Layers are numbered starting with 1, lowest to highest, row starting with 1 and top to bottom, and columns starting with 1, left to right.

Materializer works as follows: at the start, specialized part of the machine (called extruder) is on the first layer, first row and first column. Extruder can move around, and fill any cube with a special material. First layer is at the very bottom – a shape is produced starting from the bottom towards the top. Extruder can move freely within a layer, but after moving up a layer it can't go lower again.

Materializer accepts the following commands, and executes them in sequence:

- `M` x y – move the extruder by x columns and y rows. ($x, y \in \{-1, 0, 1\}$),
- `N` – move the extruder up a layer,
- `E` – fill the current cube with material,
- `C` – end.

To produce a stable shape, you have to satisfy some requirements. A cube on layer w, in column x and row y can be filled with material only if at the time of executing `E` command one of the following conditions is true:

- $w = 1$ (extruder is on the first layer),
- cube in the position $(w-1, x, y)$ is already filled,
- there exists a sequence of positions $(x, y) = (x_0, y_0), (x_1, y_1), ..., (x_n, y_n)$ such that:
  - $1 \leq n \leq$ **k**,
  - for $i \geq 1$ it is true that $|x_{i-1} - x_i| + |y_{i-1} - y_i| = 1$,
  - for $i \geq 1$, cube in the postition (w, x_i, y_i) is filled,
  - cube in the position $(w-1, x_n, y_n)$ is filled.

It's possible that you can't fulfill a given order without violating those constraints. To deal with this, you can produce additional cubes (acting as a "scaffolding"), and then remove it. Please help Chris – find a sequence of move that will produce given orders quickly, without wasting too much material.

Input

The first line contains a single integer **t**, denoting number of testcases. Then, testcases follow.

First line of the testcase consist of four integers **c**, **b**, **a**, **k** ($1 \leq$ **c**, **b**, **a** $\leq 100, 1 \leq$ **k** ≤ 3) – dimensions of the given order and the stability coefficient. Then, layers follow.

Description of a layer consist of **b** lines. Every line is a string of length **c**. Character in the line number y and column x corresponds to a cube in x–th column and y–th row – `#` denotes that cube has to be filled, `_` means that this cube may remain empty.

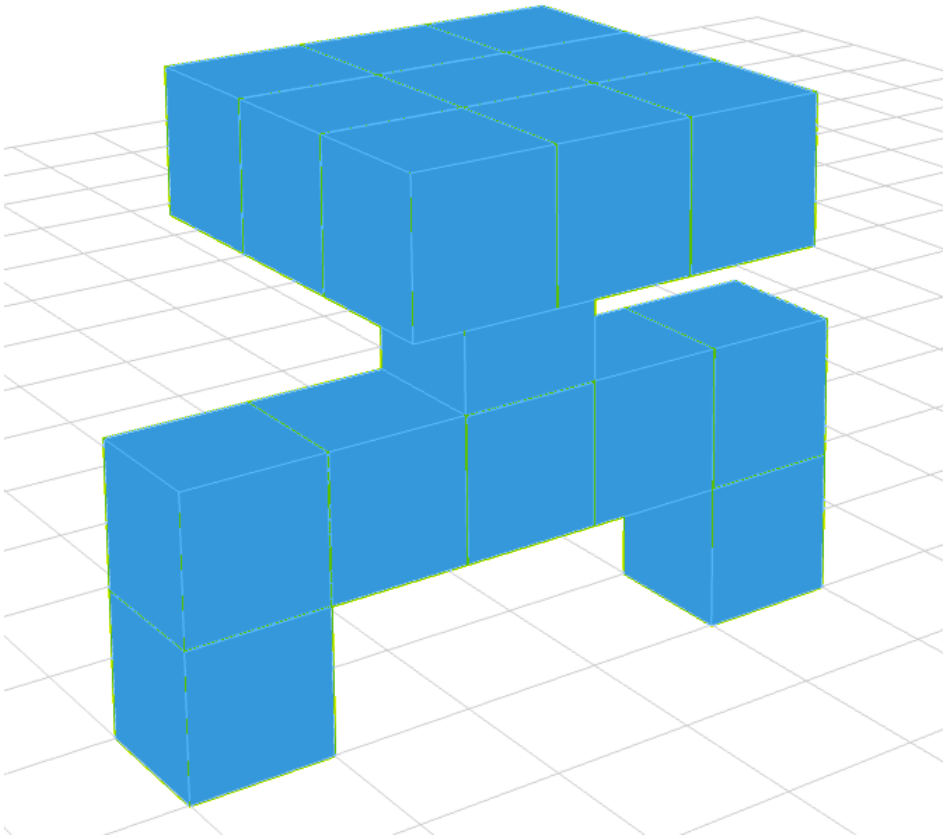
Output

For every testcase you should output a sequence of moves that fills out all cubes specified in the order. Given sequence can also fill out any cubes that are not specified there (doesn't matter if it's easy to remove them later). Extruder cannot move outside a cuboid containing the order (**a** $\times$ **b** $\times$ **c**). Sequence of moves has to end with a command `C`. Maximum allowed number of moves is $4 \cdot$ **a** $\cdot$ **b** $\cdot$ **c**.

Example

Input	Output
1 5 3 4 1 _____ #___# _____ _____ ##### _____ _____ __#__ _____ _###_ _###_ _###_	M 0 1 E M 1 0 M 1 0 E M 1 0 M 1 0 E N E M -1 0 E M -1 0 E M -1 0 E M -1 0 E N M 1 0 E M 1 0 E M 1 0 E N E M -1 0 E M -1 0 E M 0 1 E M 1 0 E M 1 0 E M 0 -1 M 0 -1 E M -1 0 E M -1 0 E C

Sample visualization



Scoring

If there are r commands given, and there are n additional cubes, then a score for the given order is $r+10n$. Final testcase score is a sum of scores for all of the orders. In the sample testcase $r = 45$, $n = 3$ so the score is 75.

This is a minimization task – the less points, the better.