Hash Code

Painting the Facade

Practice Problem for Hash Code 2016

Introduction

The day has come to paint a huge mural on the facade of the **Google Paris** office.

A picture has been decided on and a specialized machine has been hired to perform the painting. Unfortunately, it turns out that the painting operations supported by the machine are quite low-level.

Therefore before putting the machine to work, the target picture has to be translated into a list of instructions supported by the machine.

Task

Given the target picture, come up with a list of commands that produce it using as few commands as possible.

Problem description

Picture

The picture is a rectangular grid of square cells, each of which either must be painted in black, or must remain clear. At the beginning, the entire wall (all cells) is clear.

The cells of the picture are referred to using their coordinates. [**R**, **C**] denotes a cell in the **R**-th row and the **C**-th column of the picture. Indexing of the rows and columns is 0-based, with the cell [0, 0] located in the top-left corner of the picture.

Painting

The machine supports the following commands:

● PAINT_SQUARE **R C S** - paints all cells within the square of $(2S + 1) \times (2S + 1)$ dimensions centered at [**R**, **C**]. In particular, the command "PAINT_SQUARE **R C** 0" paints a single cell [**R**, **C**]. For the command to be valid, the entire square has to fit within the dimensions of the painting.

● PAINT_LINE **R1 C1 R2 C2** - paints all cells in a horizontal or vertical run between [**R1**, **C1**] and [**R2**, **C2**], including both ends, **as long as** both cells are in the same row or column or both. That is, at least one of the two has to be true: **R1** = **R2** or/and **C1** = **C2**.

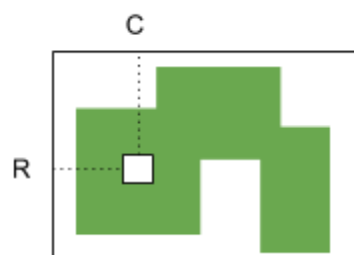● ERASE_CELL **R C** - clears the cell [**R**, **C**].
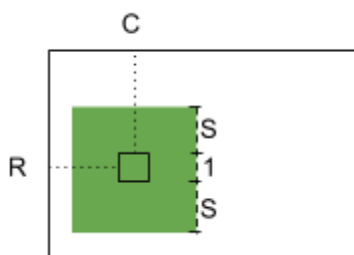
PAINT_SQUARE
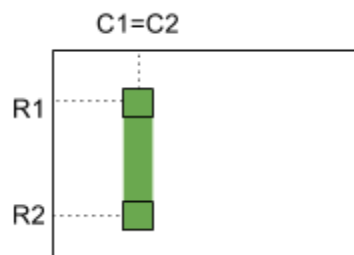
PAINT_LINE
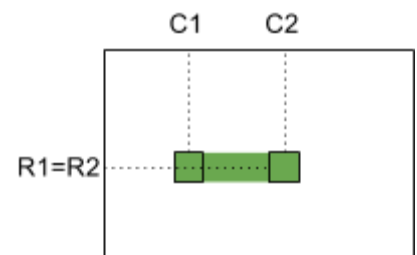
PAINT_LINE

ERASE_CELL

Input data

The input data is provided in a plain text file containing exclusively ASCII characters with lines terminated with a single '\n' character at the end of each line (UNIX-style line endings).

The file consists of:
- one line containing the following natural numbers separated by single spaces:
  - **N** denotes the number of rows of the picture ($0 < N \leq 1000$)
  - **M** denotes the number of columns of the picture ($0 < M \leq 1000$)
- **N** subsequent lines describing individual rows of the picture. Each i-th ($0 \leq i < N$) such line contains **M** characters describing picture cells in consecutive columns of the i-th row of the picture, starting with the column 0. Each character is either:
  - '.' - denoting a cell that has to remain clear
  - '#' - denoting a cell that has to be painted

Input example

`5 7`	5 rows, 7 columns.
`....#..`	First row of the picture.
`..###..`	Second row.
`..#.#..`	Third row.
`..###..`	And so on.
`..#....`	Further redundancy is redundant.

Example input file.

Submissions

Validation

For the solution to be accepted, it has to meet the following criteria:
- the format of the file must match the description below
- all cells referenced in the painting commands must fit within the dimensions of the grid
- the provided list of commands must produce the picture specified in the input file. If the resulting picture differs from the target one, the solution will not be accepted.
- the number of painting commands provided cannot be bigger than the number of cells in the picture

File format

A submission file has to be a plain text file containing exclusively ASCII characters with lines terminated with either a single '\n' character e (UNIX-style line endings) or '\r\n' characters (Windows-style line endings).

The file has to start with one line containing a single natural number **S** representing the number of painting commands ($0 \leq S \leq NM$). Then individual painting commands have to follow, each in a separate line. Each painting command has to conform with the format described in the problem description above.

Example

The following is a valid submission for the input file presented above.

```
4                       4 commands in total.
PAINT_SQUARE 2 3 1      Paint a square of 9 cells centered at [2, 3].
ERASE_CELL 2 3          Unpaint a single cell in the middle of the square.
PAINT_SQUARE 0 4 0      Paint a single cell at [0, 4].
PAINT_SQUARE 4 2 0      Paint a single cell at [4, 2].
```

Example submission file.

The following is also a valid submission for the input file presented above.

```
4                       4 commands in total.
PAINT_LINE 0 4 3 4      Paint a vertical line from [0, 4] to [3, 4].
PAINT_LINE 1 2 4 2      Paint a vertical line from [1, 2] to [4, 2].
PAINT_LINE 1 3 1 3      Paint a single cell at [1, 3].
PAINT_LINE 3 3 3 3      Paint a single cell at [3, 3].
```

Example submission file.

Scoring

The score of a valid submission is calculated as the number of cells in the picture minus the number of painting commands provided. The goal is to maximize this score.

Pictures: Eric Laignel.