{EPITECH}
LEARN DIFFERENT*

SETTING UP

ELEMENTARY PROGRAMMING IN C

* apprendre autrement

SETTING UP

binary name: setting_up
language: C
compilation: via Makefile, including re, clean and fclean rules
Authorized functions: open, read, write, close, malloc, free, stat

- ✓ The totality of your source files, except all useless files (binary, temp files, objfiles,...), must be included in your delivery.
- ✓ All the bonus files (including a potential specific Makefile) should be in a directory named bonus.
- ✓ Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

Lore

> Your first day in this basic programming module begins with a sense of anticipation. You have worked hard to be here, and now that you have walked through the doors of this prestigious school, everything seems possible.
>
> You are led into a large room. Around you, there are tables, chairs, cluttered machines, and cables snaking in every direction. Your guide stops and makes a sweeping gesture with his arm. “This is your laboratory,” he says with a wink. “At Epitech, we believe in learning by doing. It’s up to you to transform this space into a place conducive to creation and innovation.”
> You look around. The space is vast but messy. You realize that to succeed, it is essential to create an optimal working environment. Your first challenge: to delineate the largest possible square space in the center of the laboratory. This is where you will build, experiment, and bring your ideas to life.
>
> Fortunately, you are not entirely alone in this task. You are given a file with the current layout of the room. Your mission is to use dynamic programming to determine the largest free square. This will be the foundation for all your future projects and inventions at Epitech.
>
> **— Narrator —**

The project

Objectives

You must find the largest possible square on a board while avoiding obstacles.
The board can be obtained two ways. The first one is by reading a file passed as the program's argument.

The file is valid if it is respecting those constraints:
- its first line contains the number of lines on the board (and only that),
- '.' (representing an empty place) and 'o' (representing an obstacle) are the only two characters for the other lines,
- all of the lines are of the same length (except the first one),
- it contains at least one line,
- each line is terminated by '\n'.

You program must print the board, with some "." replaced by "x" to represent the largest square you found.

If ever there are several solutions, you have to represent only the highest square. If they are still several solutions, choose the square to the left.

Generating your own

The second way to obtain a board is to generate one based on given parameters. The parameters will be a number, representing the width and height of the board, and a pattern that will be repeated line by line along the board.
You will print the solved board.

Examples

```
Terminal
B-CPE-110> cat -e example_file
9$
.........................$
....o....................$
............o............$
.........................$
....o....................$
............o............$
.........................$
......o............o.....$
..o.......o..............$
```

Classic execution:

```
Terminal
B-CPE-110> ./setting_up example_file | cat -e
.....xxxxxxx.............$
....oxxxxxxx.............$
.....xxxxxxxo............$
.....xxxxxxx.............$
....oxxxxxxx.............$
.....xxxxxxx...o.........$
.....xxxxxxx.............$
......o............o.....$
..o.......o..............$
```

Generating execution:

```
Terminal
B-CPE-110> ./setting_up 6 ‘‘..o..’’ | cat -e
..oxx.$
.o.xx.$
o....o$
....o.$
...o..$
..o...$
```

{EPITECH}
LEARN DIFFERENT*

* apprendre autrement