Лабораторна #4: Tetris++

Організаційні кроки:

1. Створіть свій репозиторій для завдання Lab 4 (invite link).
2. Скопіюйте увесь вміст репозиторію Lab 3 у ваш новий репозиторій
3. Продивіться відео з прикладом виконання і репозиторій з прикладом
    a. Підказочка: відео можна дивитись на швидкості 1.5-2x
4. Виберіть бібліотеку, що полегшує роботу з test doubles. Приклади:
    a. JavaScript: Jest
    b. Java: Mockito
    c. TypeScript: Jest, ts-mockito
    d. Rust: mockall
5. Замініть mock'и, що були створені у Lab 3 на mock'и з бібліотеки
6. Виконайте алгоритмічне завдання

Алгоритмічне завдання:
Потрібно доробити програму зроблему у лабораторній #3 і додати параметр при передачі якого, буде виведено не тільки останій екран гри, а і екрани на кожному кроці гри. При цьому потрібно зберегти і початковий режим роботи (коли виводиться тільки останій екран).

Формат вводу / виводу

Без параметру "другувати усі ходи"

Input	**Output**
7 8 ..p..... .ppp.... ..p..... ....#.... ...#...# #..#####	....p..... .ppp.... ..p#.... ...#...# #..#####

З параметром "другувати усі ходи"

Input	**Output**
7 8 ..p..... .ppp.... ..p..... ....#.... ...#...# #..#####	STEP 0 ..p..... .ppp.... ..p..... ....#.... ...#...# #..##### STEP 1 .....

```
..p.....
.ppp....
..p.....
...#....
...#...#
#..#####

STEP 2
........
........
..p.....
.ppp....
..p#....
...#...#
#..#####
```

Критерії оцінювання робіт

Оцінюється якість тестів, а не код, що вирішує задачу. Це означає, що вирішена задача без тестів оцінюється в 0 балів.

За виконання роботи на одній з наступних мов нараховуються додаткові бали: Kotlin, TypeScript, C#.

За виконання роботи на одній з наступних мов нараховуються подвійні додаткові бали: Rust, Haskell, Erlang, F#, OCaml, Scheme (or any other lisp).

Списана робота оцінюється в 0 балів.