Лабораторна #3: Tetris

**Організаційні кроки:**

1. Прийміть запрошення до GitHub Academy (якщо ще не зареєстровані).
2. Створіть свій репозиторій для завдання Lab 3 ([invite link](https://classroom.github.com/a/Vy9kIynD)).
3. Налаштуйте [GitHub Actions](https://github.com/features/actions) так, щоб тести запускалися на кожен push та pull request.
4. Виконайте алгоритмічне завдання.
5. Додайте файл README.md з інструкцією для налаштування середовища і запуску тестів. За цією інструкцією, людина, що перевіряє повинна бути здатна однією командою налаштувати середовище/встановити залежності та іншою - запустити тести.
6. Додайте інтеграцію інструменту, що вимірює code coverage. Зробіть screenshot результатів вимірювання і додайте його у README.md

**Критерії оцінювання робіт**

**О**цінюється 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).

Deadline: Nov 08, 2021, 21:00 CEST.

За несвоєчасно виконану роботу нараховується не більше 50% балів.

Списана робота оцінюється в 0 балів.

**Алгоритмічне завдання:**

Необхідно реалізувати логіку гри, що схожа на Тетріс. На вхід подається початковий стан екрану на якому є одна “підвішена” фігура і “ландшафт” - залишки попередніх фігур. Необхідно вивести стан екрану в грі, коли фігура “впала” (тобто гравець не давав жодних команд, а просто чекав коли фігура зіткнеться з “ландшафтом”).

# Формат вводу / виводу

Вхідний файл містить розміри екрану та знімок початковому стану гри. У ньому:

. - порожній піксель

p - піксель “підвішеної” фігури ("p" від слова "piece")

# - піксель “ландшафта” (залишків фігур, які впали раніше)

**Приклади вводу/виводу**

| **Input** | **Output** |
| --- | --- |
| 7 8  ..p.....  .ppp....  ..p.....  ........  ...#....  ...#...#  #..##### | ........  ........  ..p.....  .ppp....  ..p#....  ...#...#  #..##### |
| 5 6  ..p...  ##p.##  ##pp##  ##..##  ##..## | ......  ##..##  ##p.##  ##p.##  ##pp## |

Перевіряти коректність вводу (наприклад, чи вказані розміри співпадають з реальними розмірами екрану) не потрібно. Фігура завжди тільки одна.

# Приклад плану розв'язання

1. Зчитати розміри екрану і зберегти їх.
2. Зчитати початковий стан екрану і створити дві колекції: одну з точок “фігури”, іншу - з точок “ландшафту”. Точка це структура даних, що містить координати одного пікселя.
3. Зібрати розміри екрану, “фігуру” та “ландшафт” у одну структуру даних (наприклад, клас), яку далі будемо називати “поле”
4. Створити функцію, яка приймає на вхід “поле”, зміщує кожен піксель “фігури” на одну позицію вниз і перевіряє чи нова координата не накладається на один з пікселів “ландшафту” або виходить за межі поля. Якщо накладається або виходить - повертає незмінний стан “поля”, якщо ні - повертає новий стан “поля” (зі зміщеною “фігурою”).
5. Створити функцію, яка циклічно викликає функцію з минулого пункту до того моменту, як “поле” перестає змінюватись.
6. Створити функцію, яка віддає “поле” у текстовому вигляді.
7. Вивести результат у файл.