Лабораторна #1: GitHub setup & Game of life

**Організаційні кроки:**

1. Прийміть запрошення до GitHub Academy.
2. Створіть свій репозиторій для завдання Lab 1 ([invite link](https://classroom.github.com/a/Dd94-viJ)).
3. Налаштуйте [GitHub Actions](https://github.com/features/actions) так, щоб тести запускалися на кожен push та pull request.
4. Створіть pull request з одним unit test'ом, що проходить успішно в GitHub Actions.
5. Створіть pull request з одним unit test'ом, що падає в GitHub Actions.
6. Не зливайте в main тестові pull request'и. Вони просто демонструють, що GitHub Actions налаштовані правильно.
7. Виконайте алгоритмічне завдання.
8. 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.

**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: Oct 8, 2021, 19:00 CEST.

За несвоєчасно виконану роботу нараховується не більше 50% балів.

Списана робота оцінюється в 0 балів.

**Алгоритмічне завдання:**

Необхідно реалізувати гру "Життя" (Conway's Game of Life).

Правила гри:

* Місце дії цієї гри - «всесвіт» - це розмічена на клітини поверхня. Поверхня замкнена - лівий край поля зрощено з правим, а верхній з нижнім:

..a..

.....

c...d

.....

..b..

Клітина "**a"** є нижнім сусідом клітини "**b"**, а клітина "**c"** є правим сусідом клітини "**d"**.

* Кожна клітина на цій поверхні може перебувати в двох станах: бути «живою» (заповненою) або бути «мертвою» (порожньою). Клітка має вісім сусідів, що оточують її:

...

.x.

...

* Розподіл живих клітин на початку гри називається першим поколінням. Кожне наступне покоління розраховується на основі попереднього за такими правилами:
  + в порожній (мертвій) клітині, поруч з якою рівно три живі клітини, зароджується життя;
  + якщо у живої клітини є дві або три живі сусідки, то ця клітина продовжує жити; в іншому випадку, якщо сусідів менше двох або більше трьох, клітина вмирає («від самотності» або «від перенаселеності»).

# Формат вводу / виводу

Вхідний файл містить кількість поколінь, розміри поля і стан поля з першим поколінням. Позначення клітин:

. - порожня клітина

x - заповнена клітина

**Приклади вводу/виводу**

| **Input** | **Output** |
| --- | --- |
| 3  8 5  ........  ..x.....  ..x.....  ..x.....  ........ | ........  ........  .xxx....  ........  ........ |

# Приклад плану розв'язання

1. Зчитати перші три числа з вхідного файлу, зберегти кількість поколінь і розміри поля.
2. Створити двовимірний масив відповідно до розмірів поля.
3. Зчитати початковий стан поля в масив.
4. Створити метод, який приймає в якості аргументів стан поля і координату однієї клітини та повертає стан клітини в наступному поколінні (див. правила гри). Потрібно звернути увагу на випадок, коли клітина знаходиться на краю поля (зірочками позначено “сусідів”):

**...\*x\*.. ........**

**...\*\*\*.. \*.....\*\***

**........ \*.....\*x**

**........ \*.....\*\***

**...\*\*\*.. ........**

1. Створити метод, який приймає в якості аргументу поточний стан поля та повертає стан поля в наступному поколінні. Для цього потрібно пройти по всім клітинам поля і викликати для кожної метод з пункту (4).
2. Створити метод, який приймає в якості аргументів початковий стан поля і кількість поколінь, а повертає стан поля з заданим поколінням.
3. Викликати метод з пункту (6) з початковим полем і заданою кількістю поколінь. Записати отримане поле у вихідний файл.