Лабораторна #4: 2048

**Організаційні кроки:**

1. Цю лабораторну роботу потрібно виконувати використовуючи техніку парного програмування. Більше про неї ви можете дізнатися у [лекції #8](https://youtu.be/YNCrILnn_yc) або у цій [статті](https://martinfowler.com/articles/on-pair-programming.html).
2. Прийміть запрошення до GitHub Academy (якщо ще не зареєстровані).
3. Виберіть свого партнера і створіть свій репозиторій для завдання Lab 4 ([invite link](https://classroom.github.com/a/EhCLhIsU)). Завдання групове, тому ви і ваш партнер будете працювати над одним репозиторием.

Після кожної зміни ролей робіть commit. Ваша історія комітів повинна виглядати як “комміт від А, коміт від Б, коміт від А, ...”.

1. Налаштуйте [GitHub Actions](https://github.com/features/actions) так, щоб тести запускалися на кожен push та pull request.
2. Виконайте алгоритмічне завдання.
3. 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: Nov 26, 2021, 21:00 CEST.

За несвоєчасно виконану роботу нараховується не більше 50% балів.

Списана робота оцінюється в 0 балів.

**Алгоритмічне завдання:**

Вхідний файл містить початковий стан поля у грі 2048. У цій грі поле має розмір 4 на 4 клітинки. У кожній клітинці міститься число, що є степенем двійки (2, 4, 8, тощо). Поле можуть зсувати вліво, вправо, вгору та вниз. Якщо стикаються два однакових числа, то вони зливаються у одну клітину з наступним степенем двійки. Приклад:

| 0 0 0 0  2 0 2 0  1 1 1 1  0 4 0 8 | **зсуваємо праворуч** | 0 0 0 0  0 0 0 4  0 0 2 2  0 0 4 8 |
| --- | --- | --- |

Якщо стикаються різні числа - вони не зливаються.

# Формат вводу / виводу

Вхідний файл містить початковий стан поля та послідовність ходів розділених пробілами. Ходи позначені літерами:

L - зсув ліворуч

R - зсув праворуч

U - зсув вгору

D - зсув вниз

Перевіряти коректність вхідного файлу не потрібно (усі числа - або 0, або степені двійки починаючи з 2, поле завжди 4 на 4, зайвих символів немає).

**Приклади вводу/виводу**

| **Input** | **Output** |
| --- | --- |
| 0 0 0 0  2 0 2 0  0 0 0 0  0 4 0 8  R U R | 0 0 0 8  0 0 0 8  0 0 0 0  0 0 0 0 |
| 1 1 1 1  0 0 0 0  0 0 0 0  0 0 0 0  R D L U | 4 0 0 0  0 0 0 0  0 0 0 0  0 0 0 0 |

# Приклад плану розв'язання

1. Зчитати початковий стан ігрового поля у двовимірний масив, та команди в одновимірний масив.
2. Написати функцію, яка приймає на вхід ігрове поле і номер рядка та повертає цей рядок у вигляді списка.
3. Написати функцію, яка приймає на вхід ігрове поле і номер стовпчика та повертає цей стовпчик у вигляді списка.
4. Написати функцію, яка приймає на вхід список і напрямок зсуву та повертає результат зсуву. Для цього йдемо по вхідному списку і дивимось на поточну комірку і наступну: якщо вони однакові, то знаходимо наступну степінь двійки (множимо на 2) і додаємо результат у результуючий список. Якщо неоднакові або ми дійшли до кінця - додаємо поточну комірку до результуючого списку. Якщо довжина результуючого списку менше 4, то доповнюємо його нулями з необхідної сторони в залежності від напряму зсуву.
5. Написати функцію, яка приймає на вхід початковий стан поля та напрям зсуву, та повертає новий стан поля. Для реалізації користуємося функціями з попередніх пунктів.
6. Написати функцію, яка приймає на вхід стан поля та повертає рядок з його текстовим виглядом.