Привет!

Спасибо за интерес к нам.

Предлагаю выполнить тестовое задание, которое поможет нам провести оценку компетенций перед звонком с командой. Если есть какие-то вопросы перед тем, как приступить к заданию, спрашивайте!

**Важно:** во всех заданиях зависимости (внешние библиотеки, утилиты и пр.) должны подключаться через стандартные менеджеры зависимостей:

* для Java — через **Gradle** (build.gradle) или **Maven** (pom.xml),
* для JavaScript/TypeScript — через **npm** или **yarn** (package.json).

Решение должно быть воспроизводимым: достаточно одной команды (gradle build, npm install, и т.д.), чтобы подтянуть все зависимости и запустить проект без ручной установки библиотек**.**

**Крайне приветствуется реализация всех заданий без использования фреймворков.**

**1. Задача “сортировка жидкостей”**

Реализовать программу, решающую задачи популярной головоломки “сортировка жидкостей”.

Игра хорошо известна, но сформулируем правила игры математически. Дано некоторое количество ***N*** пробирок одинакового объёма ***V***. В задаче объём дискретный, измеряется в количестве некоторых минимальных единиц жидкости, назовём их “капли”, поэтому ***V*** - натуральное число. Дано некоторое количество ***M*** разноцветных жидкостей (***N > M***) объёма ***V***. Разноцветные жидкости перемешаны и разлиты в ***M*** пробирок. При этом капли жидкостей разного цвета не смешиваются, а выстраиваются в пробирке вертикально в ряд. ***(N - M)*** пробирок пусты. Задача: переливая жидкости из пробирок в пробирки сортировать их таким образом, чтобы каждая жидкость одного цвета оказалась в отдельной пробирке. За один ход можно перелить жидкость из пробирки A в пробирку B по следующим правилам:

* в пробирке A есть жидкость
* в пробирке B есть пустое место
* переливать можно только верхние капли и только одного цвета
* переливать жидкость из пробирки A можно только в том случае, если в пробирке B верхняя жидкость того же цвета, либо пробирка B пуста
* если в пробирке A сверху находится более одной капли одного цвета, их можно перелить либо все, либо столько, сколько есть места в пробирке B

Пример исходного и конечного состояния:

 

Пронумеруем все пробирки слева-направо сверху-вниз от нуля, обозначим парой (A, B) один ход (переливание из пробирки A в пробирку B). Тогда пример решается следующим образом за 43 хода:

( 0, 12) ( 0, 13) ( 2, 13) ( 6, 0) ( 2, 6) ( 3, 2) ( 7, 13) ( 8, 13)

(10, 0) ( 4, 10) ( 4, 7) (11, 2) ( 3, 11) (12, 3) ( 1, 12) ( 1, 4)

( 5, 1) ( 5, 12) ( 5, 3) ( 1, 5) ( 4, 1) ( 4, 8) ( 6, 4) ( 6, 1)

( 7, 6) ( 8, 4) (11, 7) (11, 5) (11, 12) ( 2, 11) ( 2, 8) ( 3, 2)

( 3, 11) ( 7, 3) ( 7, 12) ( 9, 7) ( 9, 2) ( 9, 7) ( 6, 9) (10, 7)

(10, 8) ( 3, 10) ( 6, 8)

Требования к программе:

* исходные данные должны задаваться двумерным массивом размерности ***NxV***
* цвета жидкостей могут быть обозначены числами и/или буквами
* результат выполнения программы - последовательность ходов, приводящих заданное исходное состояние к отсортированному состоянию

Примечание: от программы не требуется, чтобы она приводила оптимальное решение.

**2. Сервис обмена файлами**

Сделать сервис, который позволяет загружать файлы и получать временную ссылку на скачивание.

**Требования:**

1. Frontend
   1. Стек: Vanilla JS, CSS, HTML
   2. Форма загрузки файла
   3. Отображение статуса загрузки
   4. Отображение ссылки
2. Backend
   1. Стек: NodeJS (TypeScript) / Java
   2. Автоматическое удаление устаревших файлов (устаревшими файлами считать те, которые не скачивали в течение определенного срока, например, 30 дней)
   3. Генерация уникальной ссылки на скачивание

**Будет плюсом:**

* Статистика по файлам
* Загрузка файлов доступна только авторизованным пользователям
* **Не** использовать Vue.js, React, Spring Framework

**3. Сервис прогноза погоды по городам**

Необходимо реализовать сервис, который будет получать текущий прогноз погоды для указанного города, используя открытое API (<https://open-meteo.com>). Сервис должен кэшировать данные о погоде в Redis, а также визуализировать данные о температуре в виде графика, отображающего изменения температуры по часам.

* Сервис должен предоставлять API по следующему пути:  
    
   GET /weather?city={city}

Где {city} — это название города (например, Moscow, Berlin и т.д.).

При первом запросе на этот путь сервис должен:

1. Получить координаты города (широту и долготу) через API геокодинга.
2. Получить прогноз погоды **на ближайшие 24 часа** через API (с использованием координат).
3. Сохранить данные о прогнозе погоды в Redis с временем жизни 15 минут.

При последующих запросах в течение 15 минут сервис должен отдавать данные из Redis без повторного обращения к API.

Если данные о погоде устарели (или не кэшированы), сервис должен обновить информацию, получив её из API.

Визуализировать изменения температуры в виде графика, показывающего температуру по часам в течение дня.

Используемые внешние API:

* **Геокодинг города**:

https://geocoding-api.open-meteo.com/v1/search?name={city}

* **Прогноз погоды**:

https://api.open-meteo.com/v1/forecast?latitude={lat}&longitude={lon}&hourly=temperature\_2m

* **Требования**

1. Стек: NodeJS (TypeScript) / Java.
2. Допустимо использование сторонних библиотек для создания графиков и работы с JSON.
3. При желании можно реализовать своё in-memory кэширование (вместо Redis).

* **Будет плюсом:**

1. **Не** использовать Vue.js, React, Spring Framework

🤓 Как мы будем оценивать выполненные задания? Мы будем обращать внимание на полноту решения задания (1), чистоту кода (2), понимание принципов ООП (3), простоту запуска приложений (4) и работу с GitHub (5).

Решения будем ожидать на Github с README файлом, в котором будет краткое описание и инструкция по запуску. Все приложения должны запускаться из командной строки без необходимости установки каких-либо IDE.

Совет: разбей задание на отдельные коммиты, не надо пушить в репозиторий сразу финальную версию.

📧 Пришли ссылку с опубликованным решением на GitHub Ксюше (HR) на [почту](mailto:sadchikova@doczilla.ru) или в [телеграм](https://t.me/sadchikova_00).

Удачи!