Вариант заключительного этапа
11 класс

Всероссийской олимпиады школьников «Высшая проба»
по профилю «Промышленное программирование»
2024/2025 уч. г.

Блок «Теория»

Задача 1. Пирамидки чисел

макс. 5 баллов

Драконы живут высоко в горах в окружении разнообразных камней. Поэтому систему записи они придумали также основанную на камнях разного вида, составленных в пирамидки.

В распоряжении дракона Смауга есть 6 видов камней: белый, красный, желтый, зеленый, синий и фиолетовый.

Камней каждого вида очень много.

Смауг строит пирамидки вот в таком порядке (показаны несколько):

			
1	2	3	4
			
5	6	7	8
			
9	10	11	12
			
13	14	15	16
			
...	318	319	320
			
...	5184	5185	5186

Человеку привычней работать с символами, поэтому мы можем обозначить камни буквами:

Тогда 318-я пирамидка запишется в строчку так: WWRYBV

Определите, как будет выглядеть пирамидка под номером 34567.

Запишите её прописными латинскими буквами в одну строку без разделителей.

Задача 2. Квантовые драконы

макс. 5 баллов

Некоторые думают, что драконы вымерли очень давно, когда люди были ещё совсем первобытными. Другие считают, что они и по сей день прячутся где-то в малоисследованных уголках Земли или глубоких пещерах. А третьи уверены, что это всё сказки и сказочники!

Никому не приходит в голову, что драконы — квантовые объекты и существуют в суперпозиции всех этих возможностей в 11-мерном пространстве-времени.

Информация о таких драконах измеряется в питах (что-то вроде битов, только не в двоичном, а в 11-мерном пространстве событий).

Если квантовый дракон в данный момент находится исключительно в одном из 11 возможных состояний, то он несёт 1 пит информации. 2 пита соответствуют $11^2 = 121$ возможному состоянию драконов и т. д.

Килопит информации содержит в себе $11^{10} = 25937424601$ пит.

Минимально различимый нами дракон должен уметь принимать 19487171 состояние.

Определите, сколько минимально различимых драконов находится в гильбертовом пространстве, если количество информации о них равно 21 килопит.

В ответе запишите только число — количество драконов. Ничего, кроме числа, записывать в ответ не нужно.

Блок «Алгоритмы»

Задача 1. Трибуна

макс. 10 баллов

Ограничение времени	1 секунда
Ограничение памяти	256 Мб
Ввод	стандартный ввод
Вывод	стандартный вывод

Трибуна с болельщиками представляет собой прямоугольник из нескольких рядов с одинаковым числом мест в каждом ряду. Расстояние между рядами и между местами в ряду составляет 1 метр. Площадь трибуны — S квадратных метров.

Все места на трибуне заняты болельщиками. Известно, что X болельщиков со всех сторон окружены другими болельщиками (то есть их места находятся строго внутри прямоугольника-трибуны), а все остальные болельщики находятся по периметру трибуны.

Найдите количество болельщиков, сидящих по периметру трибуны.

Формат ввода
Первая строка содержит целое число S ($1 \le S \le 10^9$) — площадь трибуны в квадратных метрах.

Вторая строка содержит целое число X ($0 \le X \le 10^9$) — количество болельщиков, которые со всех сторон окружены другими болельщиками.

Гарантируется, что трибуна площади S квадратных метров с X болельщиками внутри существует.

Формат вывода
Выведите одно целое число — количество болельщиков, сидящих по периметру трибуны.

Система оценивания
Решения, работающие для случаев, где трибуна имеет форму квадрата, будут набирать не менее 4-х баллов.

Пример 1

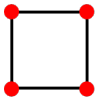
Ввод	Вывод
1	4
0	

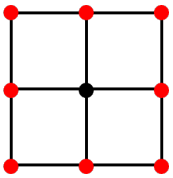
Пример 2

Ввод	Вывод
4	8
1	

Примечания

Трибуны для примеров из условия показаны на рисунке.





Задача 2. Идеальная команда

макс. 10 баллов

Ограничение времени	1 секунда
Ограничение памяти	256 Мб
Ввод	стандартный ввод
Вывод	стандартный вывод

В одной известной компании работают N сотрудников, пронумерованных от 1 до N.

Новоиспечённый глава HR-отдела Феофан предполагает, что единственный фактор, который влияет на то, смогут ли сотрудники сработаться друг с другом, — это их номера. А именно: Феофан считает, что сотрудники с номерами a и b смогут сработаться друг с другом, только если НОД(a, b) = 1, то есть если наибольший общий делитель их номеров равен единице.

Теперь Феофан хочет собрать идеальную команду из сотрудников так, чтобы любые два сотрудника в ней сработались. Помогите ему и найдите максимально возможный размер идеальной команды.

Формат ввода
Ввод содержит целое число N ($1 \le N \le 10^7$) — количество сотрудников в компании.

Формат вывода
Выведите одно целое число — максимально возможный размер идеальной команды.

Система оценивания
Решения, корректно работающие для $N \le 15$, будут набирать не менее 3-х баллов.

Пример 1

Ввод	Вывод
3	3

Пример 2

Ввод	Вывод
4	3

Примечания
В первом примере N = 3: можно взять всех трёх сотрудников, так как НОД(1, 2) = НОД(1, 3) = НОД(2, 3) = 1.

Во втором примере N = 4: можно взять сотрудников под номерами 1, 3, 4.

Задача 3. Кузнечик и теория чисел

макс. 10 баллов

Все языки Java 17 (Temurin JDK) + JSON Python 3.9 (PyPy 7.3.16)

Ограничение времени	3 секунды	6 секунд	6 секунд
Ограничение памяти	512 Мб	512 Мб	512 Мб
Ввод	стандартный ввод		
Вывод	стандартный вывод		

Кузнечик Чик — самый умный из представителей своего вида. Он лучше всего умеет делать две вещи: далеко прыгать и искать общие делители чисел.

Чтобы проверить навыки Чика, вы нашли N тростинок и пронумеровали их слева направо от 1 до N. i-я тростинка имеет высоту H_i. Чика вы посадили на тростинку номер 1.

Чик может прыгать с текущей тростинки на любую другую справа от неё, и за это он получит количество очков, равное НОД (наибольшему общему делителю) высот этих двух тростинок. Более формально: если Чик сейчас сидит на тростинке номер i и прыгает на тростинку номер j ($i < j$), то он получает НОД(H_i, H_j) очков.

Чик гарантированно выберет такой порядок прыжков, что в итоге он окажется на тростинке номер N и наберёт наибольшее возможное количество очков. Найдите это количество.

Формат ввода
Первая строка содержит целое число N — количество тростинок ($1 \le N \le 2 \cdot 10^5$) — количество тростинок.

Вторая строка содержит N целых чисел H_i ($1 \le H_i \le 10^7$) — высоты тростинок.

Формат вывода
Выведите одно целое число — наибольшее возможное количество очков, которое Чик сможет набрать.

Система оценивания
Решения, корректно работающие для $N \le 5000$, будут набирать не менее 4-х баллов.

Пример 1

Ввод	Вывод
2 10 15	5

Пример 2

Ввод	Вывод
3 5 6 8	3

Примечания

В первом примере у Чика нет выбора — он должен прыгнуть с первой тростинки на вторую, за что он получит НОД(10, 15) = 5 очков.

Во втором примере Чик сделает два прыжка:

1. С первой тростинки на вторую, за что получит НОД(5, 6) = 1 очко.
2. Со второй тростинки на третью, за что получит НОД(6, 8) = 2 очка.

Обратите внимание: эта задача имеет разные **временные** ограничения для разных языков программирования. Кроме того, решения по этой задаче на языке Python тестируются с помощью интерпретатора PyPy, время работы которого отличается от «классического» Python.

Блок «Данные»

Задача 1. Драгоценная опасность

макс. 10 баллов

Какие только существа не хранят своё драгоценное добро, нажитое непосильным трудом, в банке «Гоблин и Со»! Об их опасности служащие банка слагают легенды. Говорят, в пещерах банковой горы хранятся горы драгоценностей, сверкающих под лучами редкой свечи, а нечётные годы приносят своим опасным хозяевам наибольшую прибыль. Узнайте, как коррелирует опасность существа с приобретёнными драгоценными камнями в течение каждого нечётного года.

База данных банка «Гоблин и Со» состоит из нескольких таблиц.

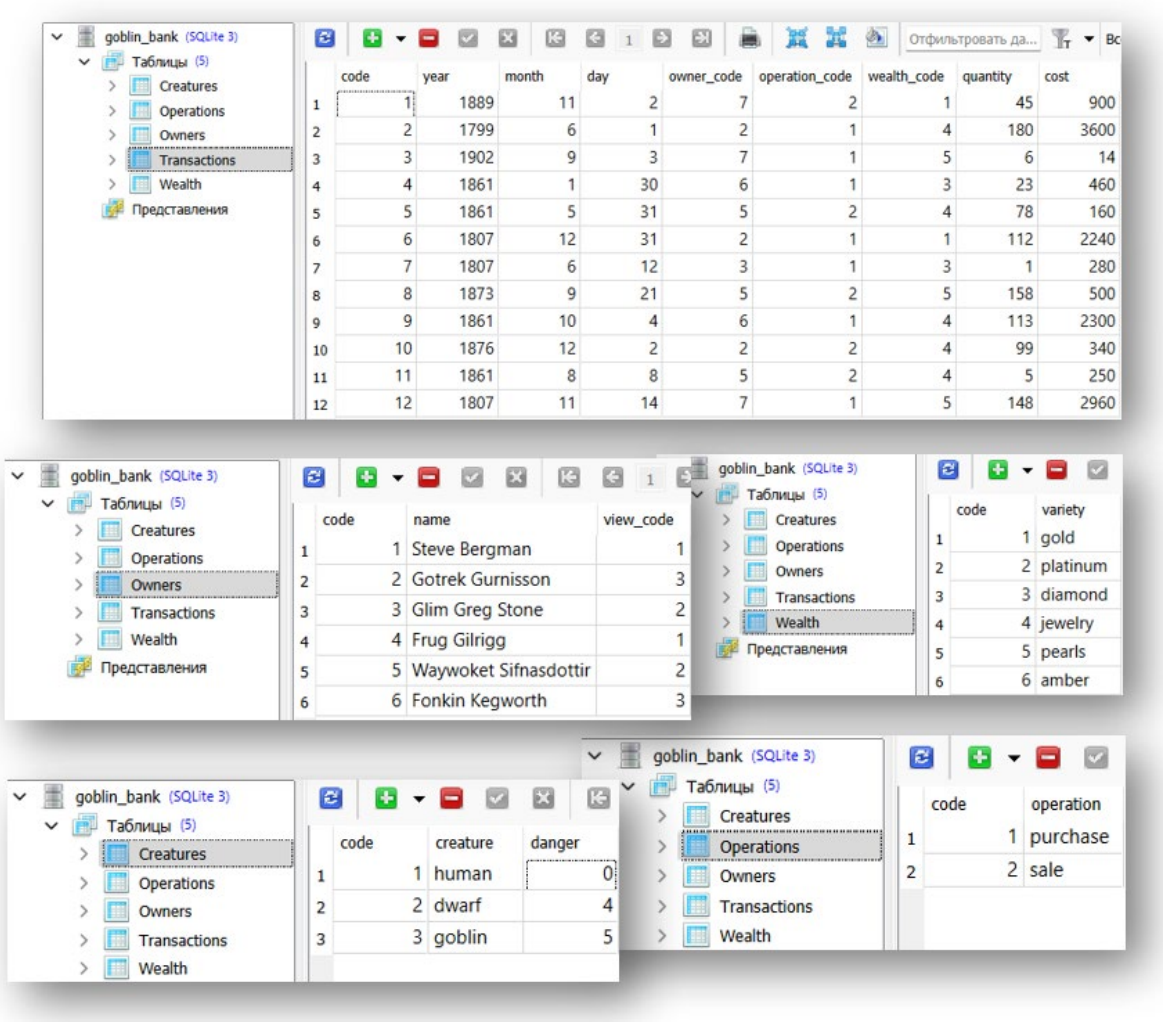
Transactions (Транзакции)
код, год, месяц, день, код_владельца, код операции, код богатства, количество, стоимость этого количества
code, year, month, day, owner_code, operation_code, wealth_code, quantity, cost

Owners (Владельцы)
код, имя, код_вида
code, name, view_code

Wealth (Виды богатства)
код, разновидность
code, variety

Creatures (Виды существ)
код, существо, опасность
code, creature, danger

Operations (Виды операций)
код, операция
code, operation



Напишите SQL-запрос, который вернёт год и для каждого нечётного года, в который были приобретения камней, сумму отношений количества приобретённых (purchase) драгоценных камней к опасности (danger) существа, которое их приобрело в данной транзакции, округлённую до двух знаков после запятой.

Драгоценными камнями считаются diamond, jewelry, pearls.

Значения должны быть отсортированы по годам по возрастанию.

Ваше решение должно содержать **только скрипт**, написанный на языке SQL (диалект SQLite), работающий с базой данных, аналогичной по структуре goblin_bank.db.

Ваш запрос не должен изменять существующую базу данных.

Пример базы данных goblin_bank.db для отладки решения можно скачать здесь.
Ваш запрос должен вернуть данные по годам и суммам, названия возвращаемых полей могут быть любыми, важно, чтобы их было два: год и сумма.

Примечания

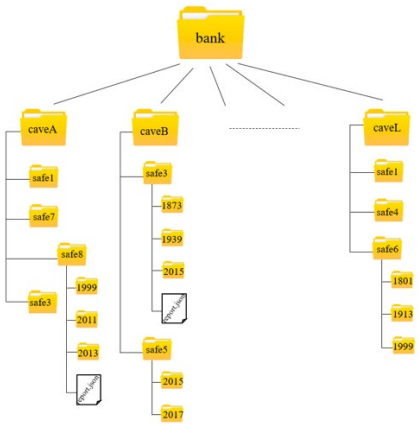
Ваш запрос будет запущен на различных тестовых данных. Для базы данных из примера для отладки выводимый результат должен быть таким:

Year	Result
1799	36
1807	0.25
1861	27.2

Задача 2. Гоблинский банк

макс. 10 баллов

В банке «Гоблин и Со» есть множество секретных пещер в недрах горы для хранения богатств избранных клиентов. В бухгалтерии на каждую такую пещеру заведена своя папочка. А в каждой папочке — сейфы с файлами с идентификационными данными и папки по транзакциям каждого года хранения богатств.



На рисунке показана иерархическая структура хранения данных в банке (папка верхнего уровня). Для каждой пещеры есть своя папка (второй уровень), а в каждой пещере находится несколько сейфов, информация о владельце каждого из которых записана в файле report.json (словарь с ключами: id, owner, who is Mr, benefits, address, contact (id, владелец, кто он, льготы, адрес, контакт для связи)) и папки с транзакциями по годам в виде набора csv-файлов, в названии которых записана дата транзакций в формате *trans<YYYY-MM-DD>.csv* с заголовками (разделители — запятые):

id, expense, wealth, amount, outcome
id, купля/продажа, вид богатства, количество, сумма

Может оказаться, что сейф меняет владельца. Тогда файл json удаляется, а папки с транзакциями по годам могут ещё оставаться некоторое время до перезаписи под нового владельца.

Вам доступен архив bank.zip со всеми папками и файлами данных. Определите, сколько (суммарное значение по полю amount) золота в любом виде (gold присутствует в названии вида богатства) положили в банк (транзакция purchase) гномы (dwarf по ключу who is Mr в json-файле) в период с 1900 по 2020 год включительно.

В ответ запишите **только число** — количество золота. Решение прикладывать не нужно.

Пример
Для архива bank_example.zip ответ должен быть 87.

Задача 3. Уплата налогов

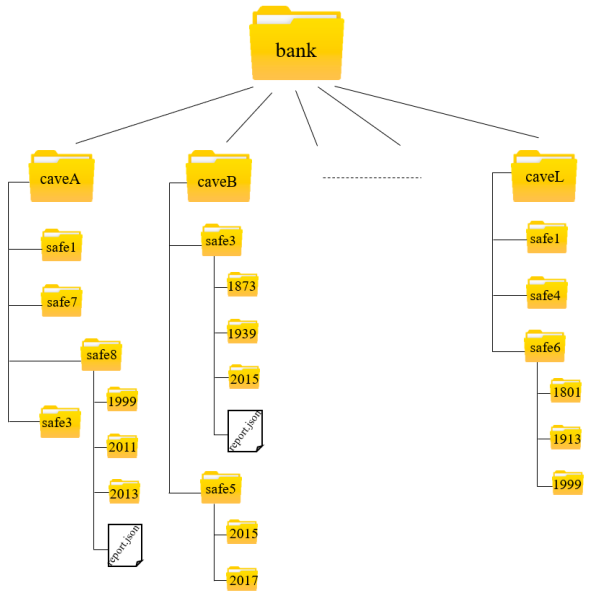
макс. 10 баллов

Ограничение времени	1 секунда
Ограничение памяти	64 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Банк «Гоблин и Со» оказывает услуги по хранению богатств в хорошо укреплённых и охраняемых драконами пещерах внутри горы.

Банк предлагает новую услугу: уплату налогов за клиентов. Для этого ему нужно уметь рассчитывать их персональный налог на прибыль в течение этого года, то есть на суммарное значение по полю outcome для проданных (sale) в этом году богатств.

Процентная ставка налога может быть снижена, если у владельца есть льготы (поле benefits в json-файле). Льгота вычитается из ставки налога, в результате налог может быть снижен, но не до нуля, минимальная остающаяся ставка — 5%.



В папке верхнего уровня находятся каталоги с сейфами по каждой пещере; в папках каждого сейфа находятся папки транзакций владельца по годам: имя каталога — год; также может быть файл json, если владелец актуальный и это его транзакции хранятся в сейфе, тогда файл report.json содержит словарь с ключами:
id, owner, who is Mr, benefits, address, contact
id, владелец, кто он, льготы, адрес, контакт для связи

В папках по годам находятся csv-файлы с транзакциями в текущий день, имена файлов записаны в формате trans<YYYY-MM-DD>.csv с заголовками (разделители — запятые):
id, expense, wealth, amount, outcome
id, купля/продажа, вид богатства, количество, сумма

Напишите для банка такую программу.

Базу данных из примера можно скачать здесь: bank0.zip

Формат ввода
Вводятся:
имя файла архива с каталогами, имя владельца, год, процент налога на прибыль в этом году.

Гарантируется, что такой владелец сейфа и такой год в сейфе существуют, но один владелец может иметь несколько сейфов в одной или нескольких пещерах. У одного и того же владельца одинаковые json-файлы с данными.

Формат вывода
Для указанного владельца для введённого года нужно найти персональный налог: из введённого значения налога в этом году вычесть его личную льготу, но снижение не может быть ниже 5%. Найти абсолютное значение налога на прибыль: суммарный доход (сумму значений по полю outcome для всех sale транзакций этого года) умножить на персональное значение налога, разделить на 100 и отбросить дробную часть (то есть округлить вниз).

Вывести полученный результат.

Пример

Ввод	Вывод
bank0.zip Amergin Clornhrim 1918 20	20

Примечания
Если при обходе архива вы будете использовать временные каталоги, не забудьте их удалить после работы.

Блок «Бэкенд»

Задача 1. Коллоквиум

макс. 10 баллов

Ограничение времени	10 секунд
Ограничение памяти	159.3671875 Мб

Маги очень заняты повседневными делами, у них всегда полно заказов: кому-то нужно недоброжелателя превратить в жабу, кому-то — поменять водопровод, не разрушая дом, кому-то — превратить порцию свинца в золото. Но и спасать мир тоже нужно, а для этого нужна сила всех магов!

Напишите сервис, отвечающий на порту 8080, который определит ближайшую к указанной дату и вернёт её в формате dd.mm.yyyy, когда все маги свободны в течение всего дня (с 0 часов до 23:59 по Гринвичу) и могут собраться вместе, чтобы спасти мир. Если таких дат несколько, надо найти самую позднюю.

В файле input.txt в формате JSON-Lines в каждой строке указана дата события в поле "date" в формате dd.mm.yyyy, в поле "time" время в формате hh:mm±hh:mm, в поле "duration" продолжительность в минутах, "name" содержит название события и "participant" имя участника, кому принадлежит это событие.

В запросе /find передаётся параметр: date — дата, ближе всего к которой нужно найти свободный для всех день.

В файле не более 100 000 строчек.

Пример запроса: http://127.0.0.1:8080/find?date=02.01.2024

Примечания
Пример input.txt

{"date": "01.01.2024", "time": "03:30+03:00", "duration": 120, "participant": "Merlin", "name": "Round table"}
{"date": "02.01.2024", "time": "05:15+04:00", "duration": 120, "participant": "Merlin", "name": "1x1 Merlin-Artur"}
{"date": "03.01.2024", "time": "17:00+03:00", "duration": 120, "participant": "Harry Potter", "name": "Quidditch"}

Пример запросов для предоставленного input.txt

http://127.0.0.1:8080/find?date=31.12.2023

31.12.2023
http://127.0.0.1:8080/find?date=01.01.2024
31.12.2023
http://127.0.0.1:8080/find?date=02.01.2024
04.01.2024
http://127.0.0.1:8080/find?date=03.01.2024
04.01.2024
http://127.0.0.1:8080/find?date=04.01.2024
04.01.2024

Для языка **Python** доступны библиотеки:

1. Flask
2. aiohttp
3. FastAPI

Для языка **Java** доступны библиотеки:

1. json-simple
2. jackson
3. async-http-client

Для языка **C++** доступна библиотека **httplib.h**

Шаблоны оформления задач можно скачать по ссылке.

Задача 2. Молчащая мандрагора

макс. 10 баллов

Ограничение времени 15 секунд
Ограничение памяти 159.3671875 Мб

Мандрагоры — очень капризные магические растения. Их нельзя переливать, но и пересушивание они выносят недолго. А если что, начинают кричать так, что мало не покажется.

Напишите сервис, отвечающий на порту 8080, который будет составлять расписание полива мандрагор.

Необходимо реализовать выполнение запросов:

http://127.0.0.1:8080/add/<plant:str>/<interval:int>/<maxdelay:int>/?date=01.01.2024 — добавить мандрагору (строковый идентификатор растения, интервал полива в днях, допустимое количество дней пересушивания, дата добавления).

http://127.0.0.1:8080/task/?date=01.01.2024 — вернуть список растений (идентификаторы растений в лексикографическом порядке через запятую без пробелов), которые нужно полить в указанную дату с учётом пропущенных поливов, если растение ещё живое.

http://127.0.0.1:8080/watering/<plant>/?date=01.01.2024 — полив растения в указанную дату.

Так как дата передаётся в каждый запрос текущая, то она никогда не убывает.

Гарантируется, что в системе менее 100 000 растений, а интервал не превышает 5 недель.

Пример:

http://127.0.0.1:8080/add/1/2/1/?date=01.01.2024
http://127.0.0.1:8080/add/2/2/2/?date=01.01.2024
http://127.0.0.1:8080/task/?date=01.01.2024
http://127.0.0.1:8080/task/?date=02.01.2024
http://127.0.0.1:8080/task/?date=03.01.2024
1,2
http://127.0.0.1:8080/task/?date=04.01.2024
1,2
http://127.0.0.1:8080/task/?date=05.01.2024
2
http://127.0.0.1:8080/watering/2/?date=05.01.2024
http://127.0.0.1:8080/task/?date=07.01.2024
2

http://127.0.0.1:8080/watering/2/?date=07.01.2024
http://127.0.0.1:8080/task/?date=08.01.2024
http://127.0.0.1:8080/watering/2/?date=08.01.2024
http://127.0.0.1:8080/task/?date=08.01.2024
http://127.0.0.1:8080/task/?date=09.01.2024
http://127.0.0.1:8080/task/?date=10.01.2024

Примечания

Для языка **Python** доступны библиотеки:

1. Flask
2. aiohttp
3. FastAPI

Для языка **Java** доступны библиотеки:

1. json-simple
2. jackson
3. async-http-client

Для языка **C++** доступна библиотека **httplib.h**.

Шаблоны оформления задач можно скачать по ссылке.

Блок «Фронтенд»

Задача 1. Треугольная жизнь с хищниками

макс. 10 баллов

Игра «Жизнь», также известная как «Жизнь Конвея», — это клеточный автомат, придуманный британским математиком Джоном Конвеем в 1970 году. Это игра с нулевым числом игроков, то есть её эволюция определяется начальным состоянием и не требует дальнейшего ввода данных.

Популярность игры «Жизнь» и даже культовый статус в 1970-х годах и позже были обусловлены её появлением в то же время, когда компьютеры становились всё более доступными. Она является полной по Тьюрингу, то есть теоретически столь же мощной, как и любой компьютер с неограниченной памятью и без ограничений по времени.

Эта задача основана на игре «Жизнь», но имеет некоторые отличия в правилах и визуализации — вам необходимо реализовать такую симуляцию.

Данная форма жизни состоит из треугольных клеток двух цветов. Красные («хищники», #f20008) охотятся за зелеными («травоядными», #00a83d) и поедают их.

Закономерности поведения клеток:

- У каждой ячейки есть 12 соседей (3 по сторонам и ещё по 3 на каждый угол).
- Если вокруг пустой ячейки есть 4 или 8 соседей одного вида, то на следующем шаге на этом месте рождается клетка соответствующего вида.
- Если вокруг пустой ячейки есть достаточное количество и «травоядных», и «хищников», то на следующем шаге приоритет рождения у «травоядных».
- Если вокруг любой живой клетки есть 4, 5 или 6 соседей её вида, то на следующем шаге она выживает.
- Если рядом с «травоядной» клеткой есть «хищная», то на следующем шаге её «поедают» (независимо от количества «травоядных» соседей) и ячейка становится пустой.
- Если рядом с «хищной» клеткой есть 4 и больше «травоядных», то на следующем шаге «хищная» клетка погибает от «переедания» (независимо от количества «хищных» соседей), и ячейка становится пустой.
- Во всех остальных случаях (когда соседей 3 и меньше или 7, 9 и больше) клетки умирают от «одиночества» или «перенаселения» и ячейка становится пустой.

Симуляция происходит на поле из треугольных ячеек шириной 28 пикселей и высотой 24 пикселя («почти равносторонних»). Пустые ячейки раскрашены в «шахматном» порядке в цвета #d8dbe0 и #f7f8fa. См. изображение:

Поле «закольцовано» по всем сторонам — т. е., не смотря на фиксированные размеры в количестве ячеек, самые крайние ячейки являются соседями. Если что-то двигается за правый край поля, то оно появляется слева (и наоборот). Аналогично, если что-то двигается за нижний край, то оно появляется сверху (и наоборот).

Т. к. треугольники ориентированы вершинами вверх или вниз, то левый и правый края поля «разрезают» соответствующие ячейки пополам. С точки зрения «закольцованности» поля эти половинки нужно трактовать как две половинки одной ячейки. Соответственно, поле всегда имеет чётное количество ячеек в ширину. Так же можно учитывать, что поле не может быть меньше, чем 6 x 6.

См. изображение, на котором живыми являются клетки в первой колонке ячеек:

Формат ввода

Начальная конфигурация поля задаётся в window.data.field в виде массива массивов одинаковой длины, где значениями являются числа: 0 (пустая клетка), 1 («травоядная» клетка) и 2 («хищная» клетка).

Например:

- поле размером 2 x 2 со всеми пустыми клетками

```
window.data = {
  field: [
    [0, 0],
    [0, 0]
  ]
}
```

- поле размером 3 x 3 с травоядной клеткой посредине

```
window.data = {
  field: [
    [0, 0, 0],
    [0, 1, 0],
    [0, 0, 0]
  ]
}
```

Кроме этого, в части тестов определено числовое поле window.data.steps. Оно указывает на количество шагов симуляции, которые нужно сделать сразу, до вызова функции готовности решения (window.onSolutionReady()). Если количество шагов симуляции не определено, то нужно просто отрисовать начальное состояние.

Часть тестов проверяет эффективность решения на больших объёмах данных.

Примечания
Решение должно представлять из себя один HTML-файл (все нужные стили должны быть включены внутрь).

HTML-элемент с полем симуляции должен иметь класс Field.

В конце работы вашего JS-кода нужно вызвать глобальную функцию window.onSolutionReady(), которая декларируется автоматически в тестовом окружении (не нужно её переопределять).

Рекомендуется использовать заготовку в архиве. В качестве решения можно отправить доработанный файл stub.html из данного архива.