Чтобы не испугаться следующего задания, вы должны сначала понять, почему такое приложение может существовать, как оно будет использоваться и в чем его смысл. Без такого понимания некоторые требования могут показаться вам чуждыми, но в этом есть определенный смысл.

Так:

1. Игрок хочет сыграть в обычную нетранзитивную игру в кости. Пользователи указывают, какие кости они хотят использовать. Основная идея заключается в возможной нетранзитивности кубиков (второй кубик выигрывает у первого, третий выигрывает у второго, но третий проигрывает первому). Кубики не могут быть жестко запрограммированы в коде; кубики передаются в качестве аргументов.

2. Можно играть 3-мя кубиками, или 4-мя кубиками, или 17-ю кубиками, или любым количеством кубиков, превышающим 2.

3. Игрок хочет иметь доказательства того, что компьютер не жульничал, потому что у второго игрока может быть серьезное преимущество. Но в то же время, конечно, игрок хочет играть "с интересом", т.е. игрок не хочет иметь возможность всегда выбирать максимально возможное значение. Если это понятно, то все дальнейшее тривиально и может быть реализовано за час или два.

Пожалуйста, обратите внимание, что следующая задача - последняя из "консольных" задач, которые помогают нам изучить некоторые базовые вещи (например, требования, хэши, HMAC и т.д.), но, начиная с задачи № 4, мы перейдем к веб-приложениям, что является одной из наших основных целей.

**ЗАДАНИЕ №3 (ДЛЯ ВСЕХ ГРУПП)**

Для тех, кто уже отправил задания №1 и №2.

Используя язык по вашему выбору - из набора C#/JavaScript/TypeScript/Java/PHP/Ruby/Python/Rust, пожалуйста, - напишите консольный скрипт, который реализует обобщенную игру в кости (с поддержкой произвольных значений на кубиках). Конечно, рекомендуется использовать язык вашей "специализации", т.е. C# или JavaScript/TypeScript, но это не обязательно.

При запуске с параметрами командной строки - аргументами метода main или Main в случае Java или C# соответственно, sys.argv в Python, процесс.argv в Node.js и т.д. - принимает 3 или более строк, каждая из которых содержит 6 целых чисел, разделенных запятыми.

Например:

python game.py 2,2,4,4,9,9 6,8,1,1,8,6 7,5,3,7,5,3.

Если аргументы неверны, вы должны отобразить четкое сообщение об ошибке, а не трассировку стека - что именно не так и пример того, как это сделать правильно (например, пользователь указал только два кубика или вообще не указал кубиков, использовал нецелые числа и т.д.). Все сообщения должны быть на английском языке.

**Важно: конфигурация костей передается в качестве аргументов командной строки; вы не "разбираете" ее из входного потока.**

Победа определяется следующим образом: компьютер и пользователь выбирают разные игральные кости, выполняют свои «броски», и побеждает тот, кто выкинет большее число.

Первым шагом игры является определение того, кто сделает первый ход. Вы должны доказать пользователю, что выбор справедлив (недостаточно сгенерировать случайный бит 0 или 1, пользователю нужно доказательство честности).

Когда пользователи делают бросок, они выбирают кости с помощью «меню» CLI и «генерируют» случайное значение с помощью компьютера. Опции состоят из всех доступных кубиков, опции выхода (отмены) и опции помощи.

Когда компьютер делает бросок, он выбирает кости и «генерирует» случайное значение.

Конечно, «случайное» поколение также должно быть доказуемо справедливым.

*Итак, вам необходимо реализовать доказуемую "справедливую" генерацию случайных целых чисел (от O до 1 или от O до 5).*

*Чтобы сгенерировать такое значение, компьютер генерирует одноразовый криптографически защищенный случайный ключ (используя соо тветствующие API, такие как SecureRandom, RandomNumberGenerator и т.д. - это обязательно) длиной не менее 256 бит.*

*Затем компьютер генерирует равномерно распределенное целое число в требуемом диапазоне (с использованием безопасного случайного числа; обратите внимание, что оператора % недостаточно для получения равномерного распределения) и вычисляет HMAC (на основе SHA3) на основе сгенерированных целых чисел в виде сообщения со сгенерированным секретным ключом. Затем компьютер отображает пользователю HMAC.*

*После этого пользователь выбирает целое число в том же диапазоне. Полученное значение вычисляется как сумма номера пользователя и номера компьютера с использованием модульной арифметики. Когда компьютер отображает результат, он также показывает используемые секретные ключи.*

Перечитайте абзац выше; последовательность имеет решающее значение (просто нет смысла делать это по-другому, например, показывать ключ перед выбором номера пользователя или показывать HMAC во второй раз вместо ключа и т.д.).

Таким образом, пользователь может проверить, что компьютер не жульничает (конечно, компьютер все еще может пытаться жульничать, но пользователь может противодействовать этому).

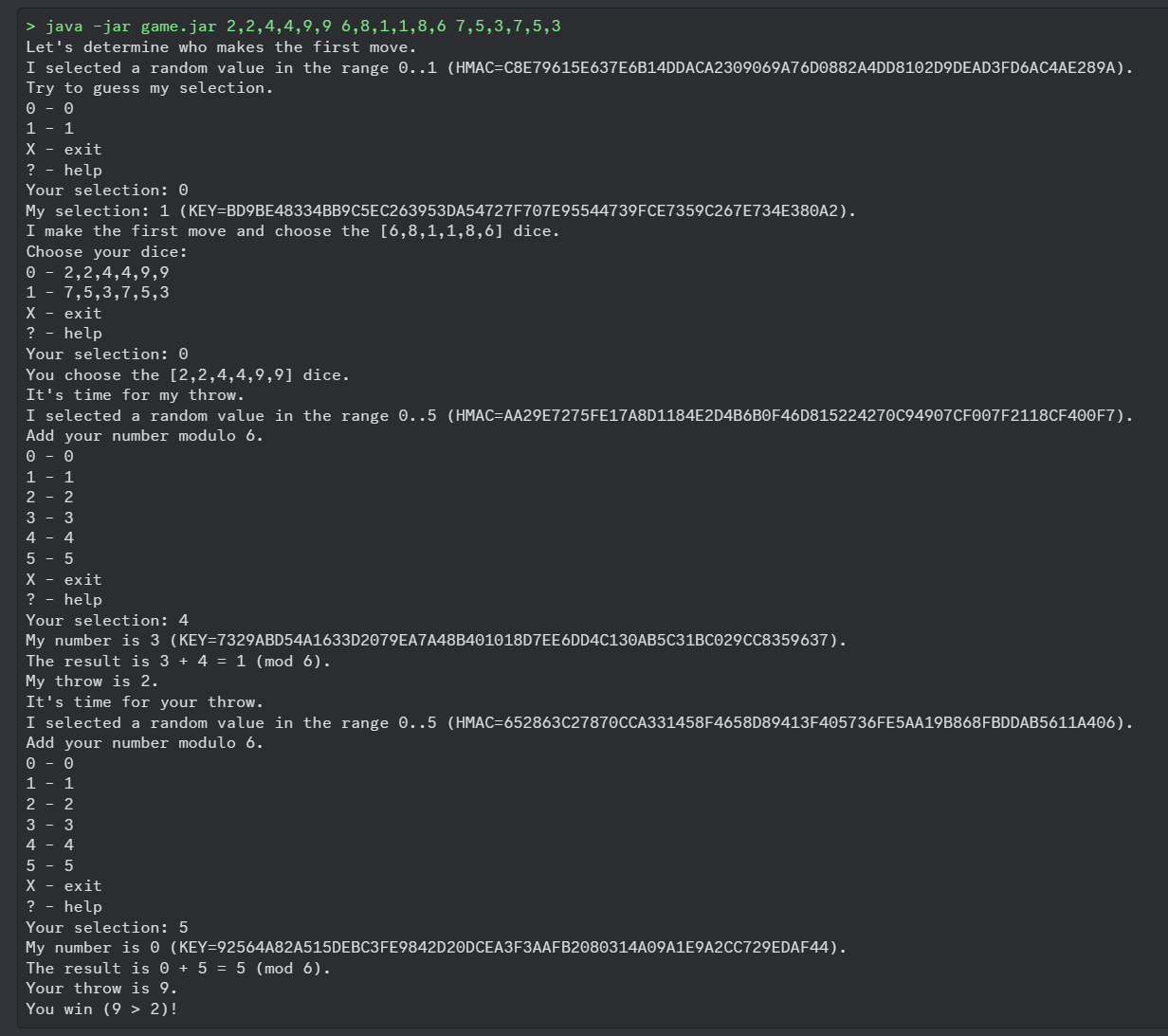
Когда вы выбираете опцию "помощь" в терминале, вам нужно отобразить таблицу (используйте ASCII-графику), которая показывает вероятности выигрыша для каждой пары игральных костей.

Генерация таблицы должна быть в отдельном классе. Вычисление вероятности должно быть в отдельном классе. Реализация "протокола" генерации достоверных чисел должна быть в отдельном классе. Генерация случайного ключа/номера и вычисление HMAC должны быть в отдельном классе. Синтаксический анализ конфигурации dice должен быть в отдельном классе. Абстракция dice должна быть в отдельном классе. Как правило, ваш код должен состоять как минимум из 6-9 классов.

Вы должны по максимуму использовать библиотеки основных классов и сторонние библиотеки, а не изобретать велосипед.

КОЛИЧЕСТВО КУБИКОВ МОЖЕТ БЫТЬ ПРОИЗВОЛЬНЫМ (> 2).

Пример:



Чтобы отправить решение, вам необходимо отправить электронное письмо по адресу p.lebedev@itransition.com со следующим текстом:

1. ссылка на видео, демонстрирующее запуск с разными параметрами (4 одинаковых кубика 1,2,3,4,5,6 1,2,3,4,5,6 1,2,3,4,5,6 1,2,3,4,5,6, а также 3 кубика 2,2,4,4,9,9 1,1,6,6,8,8 3,3,5,5,7,7), запуск с неверными параметрами (нет кубиков; 2 кубика; неверное количество сторон; нецелочисленное значение в конфигурации кубика), справочная таблица с вероятностями (на 3 кубиках из примера), вся игра, сыгранная с выводом результатов (на 3 кубиках из примера). минимум 2 прогона);

2. ссылка на публичный репозиторий Github.

Сделайте видео общедоступным. Не пытайтесь отправить свое видео на мой рабочий e-mail или еще куда-нибудь. Сделайте свое видео общедоступным. Не рассказывайте о нем, не добавляйте звук.

И в качестве пояснения: при вычислении HMAC используется тот же секретный ключ, который вы сгенерировали. И сообщение представляет собой число. После получения ключа пользователь сможет вычислить HMAC и сравнить его с HMAC, который был показан ранее. Это не очень сложно.

Распространенной ошибкой является попытка представить свой "HMAC" как хэш случайного "Key". Это не сработает. Если вы показываете одни и те же строки до выбора пользователя и после выбора пользователя, пользователь не получает новой информации, и, соответственно, вы ему ничего не доказываете. Необходимо сгенерировать ключ (с помощью безопасного генератора), сгенерировать компьютерное значение, вычислить HMAC (по стандартному алгоритму) из компьютерного значения (сообщения) и ключа (key), показать HMAC, получить пользовательское значение, вычислить результат и показать ключ. Перечитайте этот абзац еще раз до полного понимания.

**Для чего нужна эта задача?**

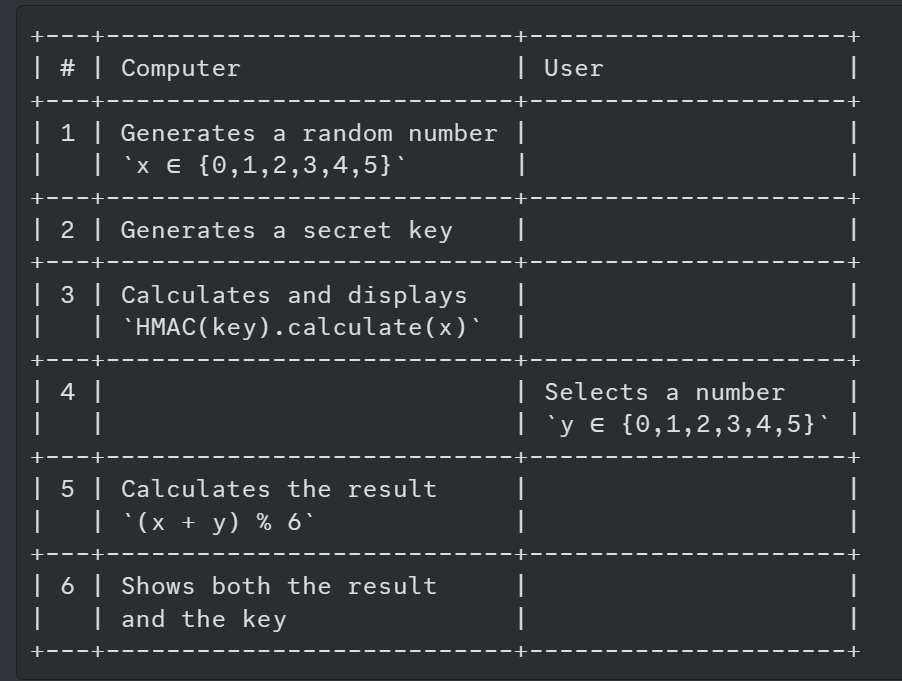
Вам нужно научиться читать и понимать требования, немного глубже разбираться в хэш-функциях, понимать, для чего они нужны, знать концепцию HMAC, научиться думать о том, как точная последовательность шагов может дать вам какое-то доказательство или контракт, работать с внешними библиотеками, затронуть некоторые основы об ООП и некоторых других вещах.

КРАЙНИЙ срок выполнения этой и остальных задач один и тот же - до даты начала проекта (вы можете ознакомиться с расписанием).

Обратите внимание, что процедура "fair" генерирует индекс лицевой стороны кубика, а не "результат броска". Пользователи могут указывать любой порядок значений лицевой стороны при запуске приложения.

Иногда слов недостаточно, поэтому вот схема совместной генерации случайных чисел в диапазоне от 0 до 5 включительно

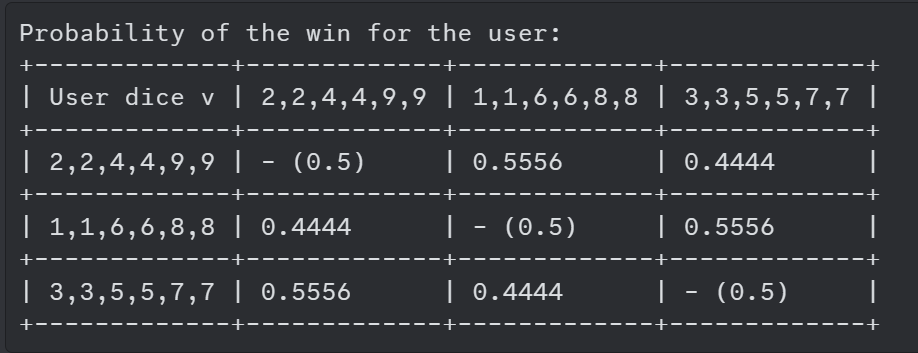
для задачи №3.



Как улучшить удобство использования справочной таблицы консоли (сделать ее более понятной).

Во-первых, вы можете выделить строку заголовка. Например, использовать какой-нибудь цвет.

Во-вторых, вы заполняете заголовки соответствующим содержанием, например, так:



Пожалуйста, помните, что целые числа могут быть больше 9, поэтому вы не можете заранее определить ширину столбцов. В любом случае, для вывода таблицы вам придется использовать стороннюю библиотеку.

И последнее, но не менее важное: вы можете добавить немного текста перед таблицей (например, чтобы помочь понять правила игры).

Конечно, для большого количества кубиков было бы неплохо реализовать какую-то разбивку на страницы, но это выходит за рамки основной задачи

(вы можете рассматривать это как необязательную задачу "для высшего класса").