Чтобы не пугаться условия \*\*\*задания №3\*\*\*, нужно прежде понять для чего такое приложение будет использоваться, в чем его смысл. Без такого понимания отдельные моменты могут показаться чуть ли не случайными, а это совсем не так.

Итак:

1. Игрок хочет играть \*\*в общего вида нетранзитивную игру с одним ходом\*\*. Не камень-бумага-ножницы, а \*\*\*любую\*\*\*, какую придумает. Главное - нетранзитивность ходов (второй ход выигрывает у первого, третий выигрывает у второго, но третий проигрывает первому при наличии 3 ходов). Соответственно, \*\*в коде не могут быть зашиты любые ходы\*\*, они передаются как аргументы.

2. \*\*Правила определяются исключительно порядком ходов\*\*, а не семантикой. Если игрок хочет поменять правила, он меняет порядок переданных ходов.

3. \*\*\*Число ходов может быть произвольным\*\*\* (нечетным больше 1) — при увеличении числа ходов вероятность ничьей приближается к нулю.

4. Игрок хочет иметь \*\*доказательство того, что компьютер не изменил свой ход\*\* после хода игрока. Но при этом, конечно, игрок хочет играть "с интересом", т.е. игрок не должен знать и \*\*\*не должен мочь вычислить ход компьютера до своего хода\*\*\*.

Если это понятно, всё дальнейшее тривиально и может быть реализовано за час-другой.

\*\*ЗАДАНИЕ №3 (ДЛЯ ВСЕХ НАПРАВЛЕНИЙ)\*\*

Для тех, кто уже прислал №1 и №2.

На языке на выбор (из данного набора — Java/C#/PHP/JavaScript/TypeScript/Ruby/Python) реализовать скрипт, который реализует \*\*обобщенную\*\* игру камень-ножницы-бумага (любое число произвольных комбинаций). Конечно, лучше использовать язык вашей специализации, но это не строго необходимо.

При запуске \*\*\*параметрами командной строки\*\*\* (аргументы метода main или Main в случае Java или C#, sys.argv в Python, process.argv под Node.js и т.д.) передаётся нечётное число ≥ 3 неповторяющихся строк (при неправильно заданных аргументах необходимо вывести аккуратное сообщение об ошибке — что именно неверно, пример как правильно). Все сообщения на английском языке. Эти строки — это ходы (например, Камень Ножницы Бумага или Камень Ножницы Бумага Ящерица Спок или 1 2 3 4 5 6 7 8 9).

Важно: \*\*\*ходы передаются аргументами командной строки\*\*\*, вы их не парсите из потока ввода (например, ход может содержать пробел, но для вашего кода это не должно иметь никакого значения).

Победа определяется так — половина следующих по кругу выигрывает, половина предыдущих по кругу проигрывает (семантика строк не важна, в какой последовательности что пользователь ввел, в такую игру и играет, даже если по его порядку камень проигрывает ножницам — для вас содержимое строк не важно).

Скрипт генерирует \*\*\*криптографически стойкий случайный ключ\*\*\* (SecureRandom, RandomNumberGenerator и т.п. — обязательно!) длиной \*\*\*не менее 256 бит\*\*\*, делает свой (компьюетра) ход, вычисляет HMAC (на базе SHA2 или SHA3) от хода как сообщения со сгенерированным ключом, показывает пользователя HMAC. После этого пользователь получает "меню" 1 - Камень, 2 - Ножницы, ...., 0 - Exit. Пользователь делает свой выбор (при некорректном вводе опять отображается "меню"). Скрипт показывает кто победил, ход компьютера и исходный ключ.

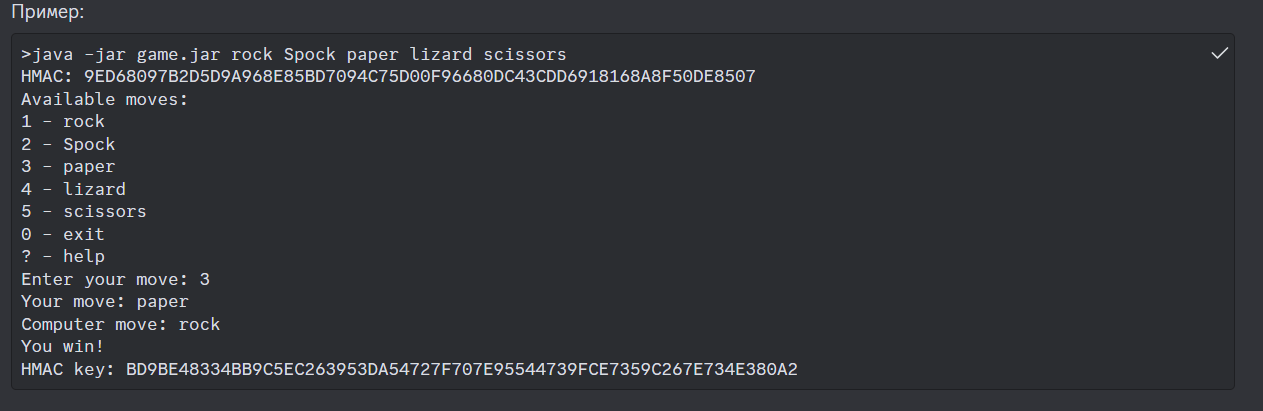
Перечитайте абзац выше, последовательность критически важна (в другой последовательности просто нет смысла, например, показывать ключ до хода пользователя или HMAC вместо ключа).

Таким образом, пользователь может проверить, что компьютер играет честно (не поменял свой ход после хода пользователя).

При выборе опции "help" в терминале нужно отобразить таблицу, определяющую какой ход выигрывает.

Генерация таблицы должна быть вынесена в отдельный класс, определение "правил" кто победил должно быть в отдельном классе, функции генерации ключа и HMAC должны быть в отдельном классе (\*\*как минимум\*\* 4 класса). По максимуму следует использовать базовые библиотеки классов и сторонние библиотеки, а не изобретать велосипед. Помощь нужно оформлить в виде таблицы N + 1 на N + 1, где N - число ходов (определяется числом переданных в скрипт аргументов). +1 потому, чтобы добавить заголовок для строк и заголовок для колонок (содержат название хода). В ячейках может быть Win/Lose/Draw.

ЧИСЛО ХОДОВ МОЖЕТ БЫТЬ ЛЮБЫМ (нечетным > 1, зависит от переданных параметров), не зашито в коде.



Для сдачи нужно прислать e-mail на \*\*p.lebedev@itransition.com\*\* cо следующим:

1. ссылку на видео, демонстрирующее запуск с 3 и 7 параметрами, запуск с неправильными параметрами (повторение или чётное число, один или отсутствие), генерацию таблицы помощи (на 5 параметрах), выбор хода, вывод результатов;

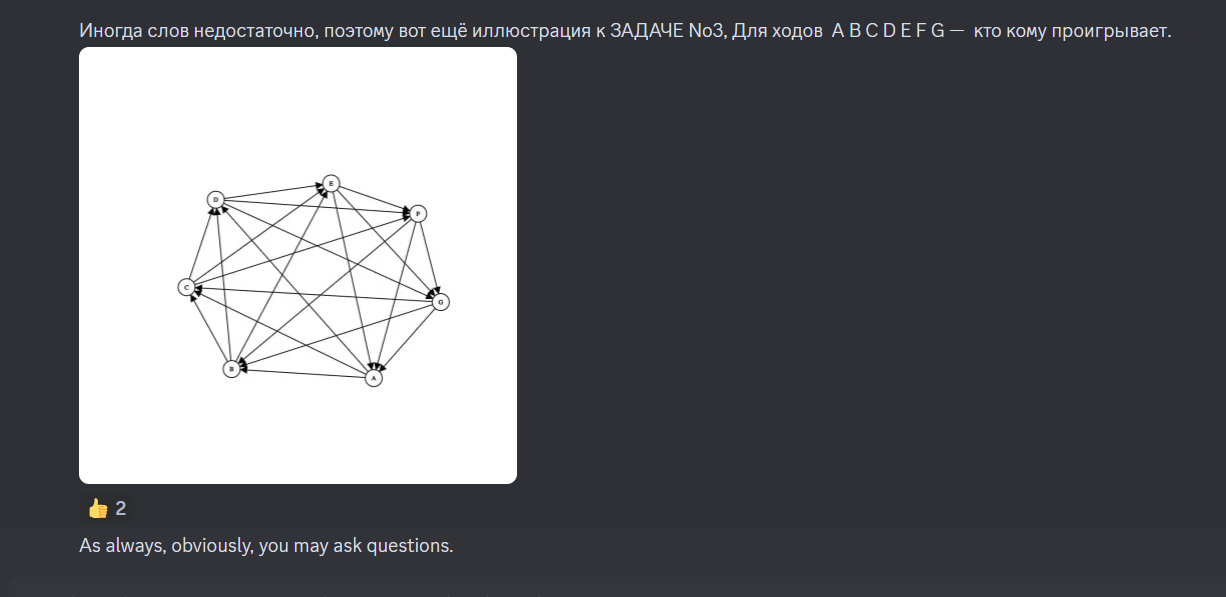
2. исходный код ссылкой на public-репозиторий гитхаб.

И как пояснение: при вычислении HMAC, ключ — это тот самый секретный ключ, который вы сгенерировали. А сообщение — это ход (прямо вот строка хода). После хода пользователя у него будет ключ, ход компьютера, пользователь сможет вычислить HMAC и сравнить с HMAC-ом, который был показан до хода пользователя. Это не очень сложно 🙂

Частая ошибка — попытка изобрести свой "HMAC" как хэш от случайного "ключа". Так не пойдёт. Если вы показываете одинаковые строки до хода и после хода, пользователь после хода не получает новой информации и, соответсвенно, вы ничего ему не доказываете. Нужно сгенерировать ключ (безопасным генератором), сделать ход компа, вычислить HMAC (стандартным алгоритмом) от хода компа (сообщение) и ключа (ключ), показать HMAC, получить ход пользователя, показать ключ. Перечитывать этот абзац до полного вкуривания.

"Правильный" порядок (хотя пользователь может использовать другой порядок и играть в игру, в которой ножницы побеждают камень; или играть в игру ХОД1 ХОД2 ХОД3): КАМЕНЬ БУМАГА НОЖНИЦЫ или КАМЕНЬ СПОК БУМАГА ЯЩЕРИЦА НОЖНИЦЫ.

DEADLINE для этой и всех оставшихся задач один и тот же — до начала проекта (можно проверить по расписанию).



Как улучшить usability таблицы помощи в терминале (сделать ее более понятной).

Во-первых, можно отрендерить текст в заголовке каким-нибудь цветом.

Во-вторых, можно заменить заголовок PC Moves чем-нибудь более сложным, например:

```

+-------------+------+-------+----------+

| v PC\User > | Rock | Paper | Scissors |

+-------------+------+-------+----------+

| Rock | Draw | Win | Lose |

+-------------+------+-------+----------+

| Paper | Lose | Draw | Win |

+-------------+------+-------+----------+

| Scissors | Win | Lose | Draw |

+-------------+------+-------+----------+

```

И, конечно, \*можно добавить какой-либо текст перед таблицей \*(например, указать, что результат описан с точки зрения пользователя и добавить, ну, пример).

Вот еще один пример таблицы помощи:

```

+-------------+------+-------+----------+------+------+

| v PC\User > | Rock | Paper | 3rd move | 4th | 5th |

+-------------+------+-------+----------+------+------+

| Rock | Draw | Win | Win | Lose | Lose |

+-------------+------+-------+----------+------+------+

| Paper | Lose | Draw | Win | Win | Lose |

+-------------+------+-------+----------+------+------+

| 3rd move | Lose | Lose | Draw | Win | Win |

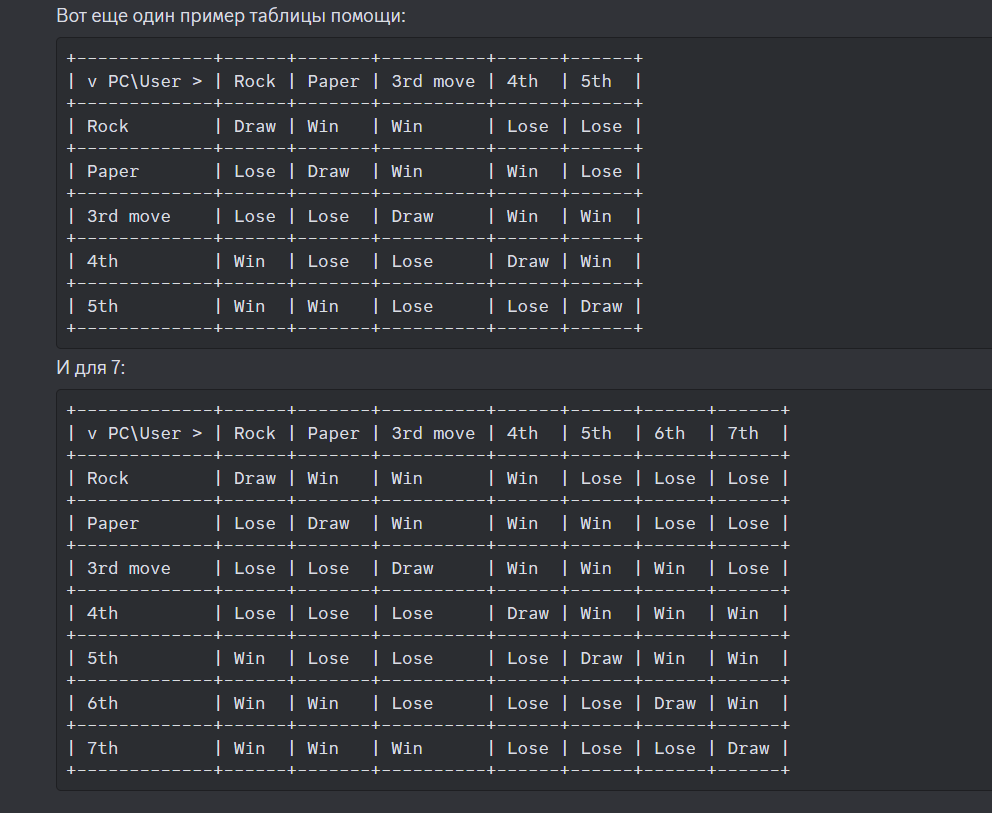
+-------------+------+-------+----------+------+------+

| 4th | Win | Lose | Lose | Draw | Win |

+-------------+------+-------+----------+------+------+

| 5th | Win | Win | Lose | Lose | Draw |

+-------------+------+-------+----------+------+------+

```  И для 7:

```

+-------------+------+-------+----------+------+------+------+------+

| v PC\User > | Rock | Paper | 3rd move | 4th | 5th | 6th | 7th |

+-------------+------+-------+----------+------+------+------+------+

| Rock | Draw | Win | Win | Win | Lose | Lose | Lose |

+-------------+------+-------+----------+------+------+------+------+

| Paper | Lose | Draw | Win | Win | Win | Lose | Lose |

+-------------+------+-------+----------+------+------+------+------+

| 3rd move | Lose | Lose | Draw | Win | Win | Win | Lose |

+-------------+------+-------+----------+------+------+------+------+

| 4th | Lose | Lose | Lose | Draw | Win | Win | Win |

+-------------+------+-------+----------+------+------+------+------+

| 5th | Win | Lose | Lose | Lose | Draw | Win | Win |

+-------------+------+-------+----------+------+------+------+------+

| 6th | Win | Win | Lose | Lose | Lose | Draw | Win |

+-------------+------+-------+----------+------+------+------+------+

| 7th | Win | Win | Win | Lose | Lose | Lose | Draw |

+-------------+------+-------+----------+------+------+------+------+

```

И так далее (нужно поддерживать любое нечетное число ходов). Конечно, для больших значений было бы здорово реализовать какой-то способ постраничного вывода, но это не входит в требования базовой задачи (вы можете рассматривать это как опциольную "на максимальную оценку" проблему).

И ещё несколько ответов:

1. Вы \*\*должны\*\* использовать встроенные и сторонние библиотеки для реализации (особенно для генерации консольной таблицы).

2. Вы (ваш код) генерируете сначала секретный ключ, потом генерируете случайны ход и затем показываете пользователю только HMAC. После этого пользователь делает свой ход. Потом вы показываете результат, сгенерированный ход и секретный ключ. И после этого пользователь может проверить, что ход компьютера вместе с ключом дают такой же HMAC, который был показан раньше.

3. В конце можно сгенерировать ссылку на какой-нибудь онлайн-сервис, который позволяет вычислять HMAC, чтобы упростить проверку для пользователя, но это не обязательно.