Игра в 21.

Обратите внимание, правила нашей игры наверняка не соответствуют правилам известных игр (блэк-джек, очко). Поэтому внимательно прочитайте правила, чтобы убедиться, что вы играете в нужную игру.

Правила игры. Играют двое. В начале первый игрок получает верхнюю карту из колоды, затем набирает карты сверху по одной до тех пор, пока не пожелает остановиться. После чего второй получает верхнюю карту из колоды и также набирает нужное количество карт. Далее для каждого из игроков считается сумма очков у него на руках.
Игрок, который набрал более 21 очка автоматически проигрывает, при условии, что второй набрал не более 21. Если оба игрока набрали более 21, объявляется ничья. Если ни один из игроков не «перебрал», выигрывает игрок с наибольшим количеством очков.
После каждого кона все карты собираются вместе и тщательно перемешиваются. Начинается следующий кон.
Мы будем считать, что в колоде 52 игральные карты. Каждая «не картинка» дает столько же очков, сколько на ней написано. Валет — 2 очка, дама — 3, король — 4, туз — 11.

Общие требования. Ваша задача придумать «хорошую» стратегию игры. То есть написать функцию, которая принимает на вход ваши текущие карты и возвращает `True` или `False` в зависимости от того, собирается она брать еще одну карту или нет. Кроме того, функция, реализующая стратегию, получает параметр — число карт, выданных *другому* игроку.
Карты должны обозначаться парой `(номинал, масть)`. Масть обозначается одной заглавной латинской буквой (H — hearts, D — diamonds, C — club, S — spade). Номинал обозначается либо цифрой (от 2 до 10), либо заглавной латинской буквой J, Q, K, A (валет, дама, король, туз).

Задача 1. Вычислите N — за сколько карт в среднем наступает перебор. Напишите следующую стратегию — брать не более N карт. Останавливаться раньше, только если у вас ровно 21.

Задача 2. В следующей стратегии берите карту только если вероятность «перебрать» меньше $\frac{1}{2}$.

Задача 3. Напишите функцию `Game()`, которая проверит, какая из предыдущих двух стратегий более удачная. `Game()` должна принимать на вход две функции (две стратегии) и количество игр, которые должны будут сыграть ваши стратегии. Внутри она должна провести много конов. Во время каждого кона нужно заново создавать колоду, перемешивать ее и вызывать каждую из стратегий, пока та отвечает `True`. Функция `Game()` должна вернуть тройку — количество побед первой стратегии, количество побед второй, количество ничьих.

Задача 4. Попробуйте улучшить стратегию из задачи 2, заменив $\frac{1}{2}$ на другой коэффициент. Подберите наиболее удачный коэффициент.

Задача 5. Для стратегии из задачи 4 посчитайте, сколько раз была набрана каждая из полученных сумм очков (для достаточно большого количества испытаний). На основе этих данных попробуйте придумать стратегию, которая победит стратегию из задачи 4.

Задача 6. Придумайте и реализуйте свою стратегию.

Задача 7. Для каждой из стратегий «брать карту, если сумма очков меньше N» ($N = 14, 15 \dots 19$) оцените вероятность набрать больше 21 очка для первого игрока.

Задача 8. Предположим, ваш соперник использует следующую стратегию: «брать карту, если сумма очков меньше N». Требуется определить число N, если вы сыграли с ним достаточно много игр и знаете суммы очков, которые ваш соперник в них набирал.
Обратите внимание — вы **не знаете** последовательность карт, которые вытягивал ваш соперник до получения той суммы очков, которую он набрал.
Решите задачу при условии, что в случае перебора очков вашим соперником вы не знаете сумму его очков, а знаете лишь о самом факте перебора.

Задача 9. Предположим теперь, что вы играете за второго игрока и знаете, что первый игрок вытащил K карт и играет, используя стратегию предыдущей задачи для некоторого известного вам N.
Для каждого K оцените с какой вероятностью первый игрок набрал $N, N+1, \dots, 21, > 21$ очков.

Соревнование стратегий

Для проверки ваших стратегий вам надо сдать файл, в котором будет функция, реализующая вашу стратегию. Имя файла должно совпадать с вашей фамилией, расширение файла должно быть `.py`. Функция должна называться `<фамилия>_strat_1`.
Фамилия (латиницей, с маленькой буквы) должна совпадать с вашим логином в ejudge без приставки m21.

В эту функцию будут передаваться следующие параметры:

- `current` — ваши текущие карты (массив кортежей, в котором карты заданы в описанном выше формате)
- `stat` — статистика результатов предыдущих игр вашего соперника. Это массив кортежей-троек:
  - номер, которым играл ваш соперник (число 1 или 2)
  - количество карт, которое взял ваш соперник
  - сумма очков, которую набрал ваш соперник (22, если был перебор)
- `another` — количество карт, которое взял ваш соперник (если вы играете вторым номером) и число -1, если вы играете первым номером.

Функция должна вернуть булевское значение:

- `True` — если вы решаете взять ещё ону карту
- `False` — если вы отказываетесь брать карту и готовы «вскрываться»

Вместе с основным файлом, содержащим функцию-стратегию можно сдать файл со вспомогательными функциями. Такой файл должен называться `<фамилия>_lib.py`. В таком случае файл со стратегией должен содержать инструкцию по импорту библиотеки:

```
from <фамилия>_lib import *
```

Если вы хотите сдать больше одной стратегии, назовите остальные файлы `<фамилия>_strat_2`, `<фамилия>_strat_3` и т.д.

Если для вашей стратегии необходимо вычислить некоторую статистику, и это можно сделать один раз (за большое количество запусков), можете сохранить её в файл <фамилия>.txt.